2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

RNR-Nav: A Real-World Visual Navigation System Using Renderable Neural Radiance Maps

Minsoo Kim[1], Obin Kwon[2], Howoong Jun[1,3], and Songhwai Oh[1,2,3]

***Abstract*— We propose a novel visual localization and navigation framework for real-world environments directly integrating observed visual information into the bird-eye-view map. While the renderable neural radiance map (RNR-Map) [1] shows considerable promise in simulated settings, its deployment in real-world scenarios poses undiscovered challenges. RNR-Map utilizes projections of multiple vectors into a single latent code, resulting in information loss under suboptimal conditions. To address such issues, our enhanced RNR-Map for real-world robots, RNR-Map++, incorporates strategies to mitigate information loss, such as a weighted map and positional encoding. For robust real-time localization, we integrate a particle filter into the correlation-based localization framework using RNR-Map++ without a rendering procedure. Consequently, we establish a real-world robot system for visual navigation utilizing RNR-Map++, which we call "RNR-Nav." Experimental results demonstrate that the proposed methods significantly enhance rendering quality and localization robustness compared to previous approaches. In real-world navigation tasks, RNR-Nav achieves a success rate of 84.4%, marking a 68.8% enhancement over the methods of the original RNR-Map paper.**

I. Introduction

The development and deployment of visual navigation systems represent a cornerstone in advancing autonomous navigation capabilities. The ability to understand visual information in environments is essential for intelligent agents to perform visual navigation tasks effectively. Numerous studies have attempted to represent the visual information from the observed scenes in various forms, including the adoption of grid-form maps [2]–[4] where each grid cell represents the environmental information of a corresponding region.

Among the myriad of approaches, the concept of renderable neural radiance map (RNR-Map) [1] has been identified as a promising approach for enhancing scene representation with visual information in real-time procedure. RNR-Map is a grid-form map that contains visual information as latent codes at each grid cell, which can be rendered to image signals by neural radiance fields (NeRF) techniques [5], [6]. The original study [1] introduces several applications of RNR-Map for vision-based robotic tasks such as localization and navigation. Though RNR-Map shows significant potential in simulated environments [7], [8], real-world environments impose unexplored challenges. In this paper, we address such challenges encountered in real-world deployment.

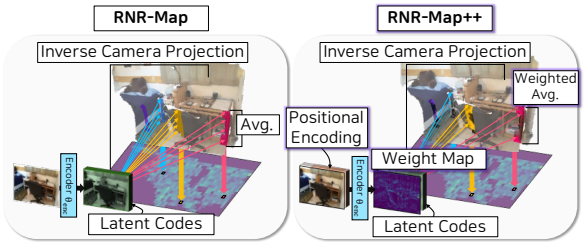


Fig. 1: **Comparison between the proposed RNR-Map++ and the original RNR-Map.**

The original RNR-Map shows poor image reconstruction results for the real-world dataset due to the loss of visual information that occurs during the mapping procedure. The projection simply averages multiple latent codes into a single vector for each grid, which blends distinct visual and spatial details. To achieve environment-agnostic deployment, we initially focus on improving image reconstruction quality by embedding the visual information more effectively. Our advanced version of RNR-Map, RNR-Map++ shown in Figure 1, implicitly learns the importance of each pixel and leverages such values before aggregating to a single vector. RNR-Map++ also adopts positional encoding of 3D positions of each image pixel to contain spatial information.

For the next step, we develop a robust and fast localization process using RNR-Map++. While [1] offers a fast localization method based on cross-correlation operation, inaccuracies in RGB-D images from real-world environments led to unstable predictions that are far from the actual pose. The RNR-Map suggests a supplementary method based on optimizing the photometric loss, utilizing noisy odometry information. However, it requires rendering as well as optimization processes for several iterations, which consumes large computation time. Here, we propose a novel localization method using RNR-Map++ by employing the classical particle filter algorithm [9] to stabilize the pose estimation. Our experiment results demonstrate that the particle filter considerably improves localization accuracy. Moreover, our localization speed appears much faster than the optimization [1] or rendering-based [10] localization approaches.

Building on these advancements, we propose RNR-Nav, a novel visual navigation framework for a real-world robot system utilizing RNR-Map++. With the given RNR-Map++ and occupancy map of the environment, our RNR-Nav system is able to find the target pose of the query image and navigate to the goal point by localizing the current pose and planning a path to the goal. In the image-goal navigation experiment in the real-world environment, the proposed framework achieves a success rate of 84.4%.

[1]Interdisciplinary Program in Artificial Intelligence (IPAI) and ASRI, Seoul, Republic of Korea (e-mail: minsoo.kim@rllab.snu.ac.kr, howoong.jun@rllab.snu.ac.kr, songhwai@snu.ac.kr).

[2]Department of Electrical and Computer Enginnering (ECE) and ASRI, Seoul National University, Seoul, Republic of Korea (e-mail: obin.kwon@rllab.snu.ac.kr).

[3]Sequor Robotics, Inc., Seoul, Republic of Korea.

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.)

Our main contributions can be summarized as follows:

- The proposed method, RNR-Map++, mitigates information loss by estimating the importance of each pixel using a weighted map and positional encoding.
- We propose a novel localization framework using RNR-Map++, alleviating unstable localization in the real world by adopting the particle filter.
- The extensive experiments show the effectiveness of the proposed RNR-Map++ and navigation framework, RNR-Nav, even in noisy real-world environments.

II. Related Work

With the advancement of NeRF [5] techniques, several localization [10], [11] and mapping [12], [13] approaches with NeRF have been developed. Such methods utilize a photometric loss from the rendered image and the observed image information to estimate the camera pose and learn map representation. NeRF itself can be regarded as a scene representation that contains visual information about any specific scene. Loc-NeRF [10] proposed a rendering-based localization method combining particle filter and NeRF. This method calculates the weight of each particle by rendering the image value from the NeRF representation and comparing it with the image observation. However, this dependency on the rendering process introduces considerable time and memory allocation, as well as extensive optimization time for training NeRF representation on a scene-by-scene basis. In contrast, our proposed localization framework does not require rendering to calculate particle weights by utilizing cross-correlation operations. Furthermore, most of the existing methods are limited to simulated environments or localization experiments in small real-world environments. The introduction of RNR-Map++ enables immediate real-world application without a need for fine-tuning, offering flexibility for deployment across diverse environments.

There have been similar real-world navigation approaches with informative scene representations [14]–[17] other than renderable information. They use pretrained features from vision-language models for the semantic scene interpretation. However, those approaches prioritize high-level semantic policy execution, often sidelining robot-centric aspects like localization or path planning. Our proposed method emphasizes the active utilization of embedded information for precise robot localization as well as navigation to the target place specified by images.

III. Preliminary

In this section, we define some notations and review the concept and details of the RNR-Map [1].

A. Map Construction and Rendering

The RNR-Map is made from RGB-D images of the desired environment. In addition, the poses and intrinsic parameters of the camera have to be known. Those images are fed sequentially to the encoder, which consists of convolutional layers. When the t-th RGB-D image $I_t \in \mathbb{R}^{H\times W\times 4}$ is given as an input, the encoder encodes the image into latent codes in identical size, $C_t \in \mathbb{R}^{H\times W\times D}$. Here, H, W, and D refer to the height, width of the image, and the dimension of the latent codes, respectively. For each pixel (h, w), we can calculate 3D position $\mathbf{q}_{(h,w)} = (q_{(h,w),x}, q_{(h,w),y}, q_{(h,w),z})$ in world coordinate system of each pixel by using known camera intrinsic parameters, the depth value, and the pose of the image $p_t = [\mathbf{R}|\mathbf{t}]$. Then, the position is discretized into the grid with position (u, v) in the map as follows:

$$(u, v) = \left(\left\lfloor \frac{q_{(h,w),x}}{s} \right\rfloor, \left\lfloor \frac{q_{(h,w),y}}{s} \right\rfloor \right), \tag{1}$$

where the grid size s denotes the height and length of each grid in RNR-Map. The projection process averages latent codes $c_{(h,w)} \in \mathbb{R}^D$ in C_t belong to the same grid (u, v) into a single latent code. Finally, the map $\mathbf{M} \in \mathbb{R}^{U\times V\times D}$ is constructed with height U and width V. In addition, the RNR-Map keeps track of the numbers of the averaged vectors in each grid as a mask $\mathbf{N} \in \mathbb{R}^{U\times V}$. Then, each component at (u, v) of the map and the mask, $M_{(u,v)} \in \mathbb{R}^D$ in $\mathbf{M}$ and $N_{(u,v)} \in \mathbb{R}$ in $\mathbf{N}$ respectively, can be derived as follows:

$$\begin{aligned} P_{(u,v)} &= \left\{ (h, w) \middle| \left(\left\lfloor \frac{q_{(h,w),x}}{s} \right\rfloor, \left\lfloor \frac{q_{(h,w),y}}{s} \right\rfloor \right) = (u, v) \right\} \\ M_{(u,v)} &= \frac{1}{N_{(u,v)}} \sum_{(h,w)\in P_{(u,v)}} c_{(h,w)}, \quad N_{(u,v)} = |P_{(u,v)}|. \end{aligned} \tag{2}$$

This process so far with a single image input is named as registration process F_{reg}. With RGB-D image I_t, the pose p_t, and the parameter of the encoder θ_{enc}, registration process is written as:

$$\mathbf{M}^l_t, \mathbf{N}^l_t = F_{reg}(I_t, p_t; \theta_{enc}), \tag{3}$$

where l denotes that the map represents only local information from a single image. As images are given sequentially, the global map and the global mask at timestep t, $\mathbf{M}^g_t$ and $\mathbf{N}^g_t$ respectively, should be updated from $\mathbf{M}^g_{t-1}$ and $\mathbf{N}^g_{t-1}$ using $\mathbf{M}^l_t$ and $\mathbf{N}^l_t$. RNR-Map computes $\mathbf{M}^g_t$ by weighted averaging both maps utilizing both masks as weights. For simplicity, $\mathbf{M}$ and $\mathbf{N}$ without any superscript refer to the global map and the global mask in the rest of the paper.

The encoder is trained in an autoencoder manner to learn to encode and decode visual information by minimizing photometric loss. Constructed RNR-Map is used as conditioned latent codes to render the image by NeRF [5] technique similar to the decoder of GSN [6]. The reader may refer to [1], [6] for further information about the decoder.

B. Localization

RNR-Map [1] describes how to use the map for an image-based localization task in two ways: *correlation-based* method and *optimization-based* method. With given key map $\mathbf{M}_k$ and a query image I_q, the localizer figures out which position, *i.e.*, grid cell, on the map is the closest location where the query image is taken. The correlation-based localizer consists of three neural networks, F_k, F_q, and F_E. First, the query map $\mathbf{M}_q$ is constructed by (3) using trained encoder assuming $p = [\mathbf{I}|\mathbf{0}]$. Then, F_k and F_q having U-Net [18] architecture convert $\mathbf{M}_k$ and $\mathbf{M}_q$ into $\mathbf{M}'_k$ and $\mathbf{M}'_q$, respectively. Since the orientation of the query map is unknown, the localizer expands the query map to 36 maps, $\{\mathbf{M}'_{q,r}\}^{36}_{r=1}$, by rotating the map 36 discrete angles $\{0°, 10°, \cdots, 350°\}$. Utilizing $\mathbf{M}'_{q,r}$ as a filter, the localizer conducts convolution with $\mathbf{M}'_k$ to gain cross-correlations between the key and the query. Finally, F_E transforms the cross-correlations into predicted heatmap $\tilde{\mathbf{H}} \in \mathbb{R}^{U\times V\times R}$,

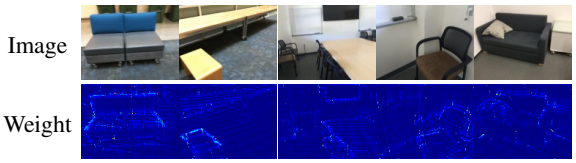


Fig. 2: **Weights computed by the encoder.** Important regions such as corners or edges have high weight values. Shown real-world images are from ScanNet [19] dataset.

which represents the estimated probability of the pose of the agent on the map. The localizer is trained to minimize the cross-entropy loss between $\tilde{\mathbf{H}}$ and ground truth $\tilde{\mathbf{H}}_{gt}$. The encoder is frozen while training the localizer.

The optimization-based localization method assumes noisy odometry information is given. The inaccurate pose is refined by optimizing the pose to minimize the photometric loss between the observation and the reconstructed image.

IV. Proposed Method

In this section, we elaborate on the motives and specifications of RNR-Map++, our advanced version of RNR-Map, which handles the problems of real-world settings. Moreover, we describe how we implement the robust localization and navigation system, RNR-Nav, using RNR-Map++ for the real-robot system.

A. Map Construction and Rendering

1) Weighted Map: Each latent code of the grid cell in the RNR-Map is computed by averaging corresponding latent codes along the height. However, this approach treats all pixels equally, disregarding the amount of visual information in each region. Generally, there are relatively less amount of valuable parts with abundant features (*e.g.*, corners and edges) than regions with few features (*e.g.*, flat walls). Therefore, the grid cell of the RNR-Map originates from a high proportion of uninformative regions. To rectify this, we adopt a weighted average for the projection.

We modify the encoder to return additional output $B_t \in \mathbb{R}^{H\times W}$ along with C_t. The weights $\beta_{(h,w)}$ are calculated by $\beta_{(h,w)} = \{softmax(B_t)\}_{(h,w)}$. Each $c_{(h,w)} \in C_t$ is multiplied by $\beta_{(h,w)}$ to calculate $M_{(u,v)}$, and $N_{(u,v)}$ keeps the sum of the weights rather than the number of the latent codes. Thus, (2) can be rewritten as follows:

$$\begin{aligned} M_{(u,v)} &= \frac{1}{N_{(u,v)}} \sum_{(h,w)\in P_{(u,v)}} \beta_{(h,w)} c_{(h,w)} \\ N_{(u,v)} &= \sum_{(h,w)\in P_{(u,v)}} \beta_{(h,w)}. \end{aligned} \tag{4}$$

Figure 2 shows weights inferred by the trained encoder. We can confirm that the high-weight values appear at the corners and edges of each image.

2) Positional Encoding: In an ideal situation such as a simulated setting, the roll and pitch values are fixed to 0, and the movement along the z-axis, *i.e.*, vertical axis, is also 0. It enables 3D positions of pixels with identical image coordinate values to be determined solely by distances. Consequently, the encoder can capture the 3D positions of pixels without requiring additional spatial information. However, the real-world operation may have rolls, pitches, and z-axis displacement. When encoded latent codes are projected on the 2D RNR-Map, information on the z-coordinate can be lost. Therefore, we decide to utilize positional encoding of the world coordinate values of each pixel. As original NeRF, cos and sin functions are applied to 3D positions $\mathbf{q}$ after multiplying several frequency values to get positional encoding as follows:

$$\gamma(\mathbf{q}) = \left[\sin(\mathbf{q}), \cos(\mathbf{q}), \cdots, \sin(2^{L-1}\mathbf{q}), \cos(2^{L-1}\mathbf{q})\right]. \tag{5}$$

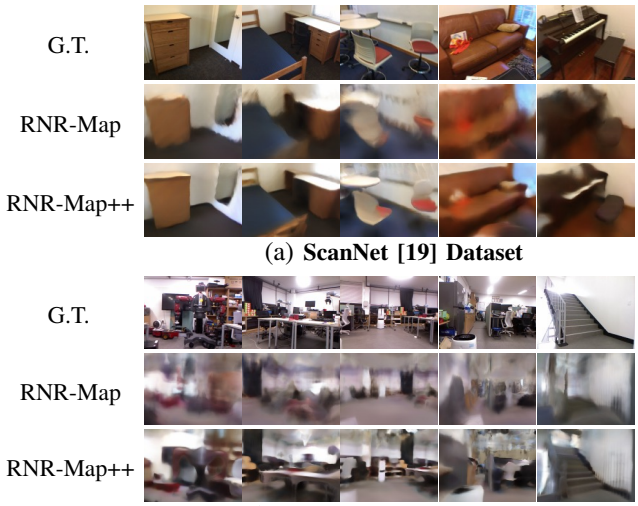
We concatenate positional encoding vectors to the RGB-D images and hand over them to the encoder as input. Then, (3) can be rewritten as follows:

$$\mathbf{M}_t^l, \mathbf{N}_t^l = F_{reg}(I_t \oplus \gamma(\mathbf{q}), p_t; \theta_{enc}), \tag{6}$$

where $\oplus$ denotes the concatenation.

3) Generalizable Input Image Size: Applying the trained encoder to diverse inputs, especially observations obtained by real-world robots, requires compatibility with varying image scales. To ensure each kernel of convolutional layers gets an identical range of spatial information as input, we standardize the height and width of each pixel on the image plane, *i.e.*, consistent focal length, f_x and f_y. Here, we opt for 128 as the f_x and f_y values. It can be satisfied by simply resizing input images utilizing given camera intrinsic parameters.

Figure 3 depicts the rendering qualities of the original RNR-Map and the advanced RNR-Map++. The proposed RNR-Map++ demonstrates an enhanced capacity to preserve and render finer details within scenes. Furthermore, we can successfully apply the trained encoder and decoder directly to images with different sizes and camera parameters, as Figure 3a and 3b. One more notable point is that, in the case of the original RNR-Map, rendering at the same 2D point often appears as a single large chunk. On the contrary, in RNR-Map++, we can observe detailed variations in height at the same location, indicating finer details in rendering. It suggests that with the addition of positional encoding within the same grid, spatial information has been augmented, enabling more precise distinctions during rendering.



(a) **ScanNet [19] Dataset**

(b) **Real-World Observation**

Fig. 3: **Comparisons of reconstruction qualities of the original RNR-Map and the RNR-Map++.** (a) Images from the ScanNet dataset, which is used for training and validation. (b) Real-world images observed by our robot system, which has different image sizes and camera intrinsics from ScanNet. Note that the same model trained by ScanNet is used for rendering in both datasets without additional training.

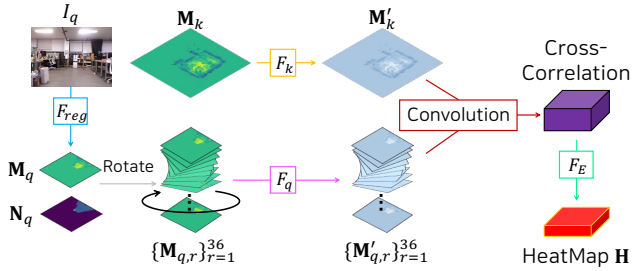


Fig. 4: **Workflow of the correlation-based localization.** Note that weight mask $\mathbf{N}_q$ is used for rotating query map $\mathbf{M}_q$.

B. Localization

1) Correlation-Based Localization: Since we adopt the weighted map for constructing RNR-Map++, we revise the way to compute rotated query maps. To maintain the importance of latent codes during rotation, we use the weight mask $\mathbf{N}_q$ as well as $\mathbf{M}_q$ from (3). We rotate $\mathbf{M}_q$ in 36 discrete angles by interpolation using weight mask $\mathbf{N}_q$ as follows:

$$\mathbf{M}_{q,r} = \frac{(\mathbf{M_q} \odot \mathbf{N_q})_r}{\mathbf{N}_{q,r}}, \quad r \in \{1, \cdots, 36\}. \tag{7}$$

where $\odot$ denotes the element-wise product and division is also an element-wise operation. Then, we convert $\mathbf{M}_{q,r}$ to $\mathbf{M}'_{q,r}$ with F_q. The overall framework is described in Figure 4. The estimated heatmap is handed over to a particle filter.

2) Particle Filter: We propose a robust and real-time applicable localization method with RNR-Map++ adopting particle filter [9], assuming noisy odometry information is provided. For N_p particles, each particle has 3-DoF pose value $\{(u_i, v_i, \alpha_i)\}_{i=1}^{N_p}$: position (u_i, v_i) on the grid-map and the orientation α_i. We use heatmap $\mathbf{H} \in \mathbb{R}^{U \times V \times R}$ of 3-DoF poses as estimated weights of particles after normalizing by $softmax$ function. At each step, predicted position $\bar{\mathbf{u}} = (\bar{u}, \bar{v})$ and orientation $\bar{\alpha}$ of the agent are computed by weighted averages using weights $\{w_{p,i}\}_{i=1}^{N_p}$ as follows:

$$\begin{aligned} \bar{\mathbf{u}} &= \frac{1}{N_p} \sum_{i=1}^{N_p} w_{p,i} \mathbf{u}_i \\ \bar{\alpha} &= \operatorname{atan2}\left(\frac{1}{N_p} \sum_{i=1}^{N_p} w_{p,i} \sin \alpha_i, \frac{1}{N_p} \sum_{i=1}^{N_p} w_{p,i} \cos \alpha_i \right). \end{aligned} \tag{8}$$

C. Navigation Using Jackal Robot

We establish a real-world robot system using RNR-Map++ for image-goal navigation tasks, RNR-Nav. We opt for the Jackal UGV as our robot platform and Kinect v1 camera to capture RGB-D images. Note that Jackal UGV provides estimated odometry values through its built-in sensors, which are suitable for the particle filter. Our RNR-Nav system comprises a localization module and a navigation module, as illustrated in Figure 5. The workflow of the system is as follows: Prior to the navigation task, both an RNR-Map++ and an occupancy map of the environment should be prepared. At the beginning, the target pose p_{trg} is determined based on the given target RGB-D image I_{trg} utilizing the RNR-Map++ and correlation-based localizer. The localization module, consisting of a correlation-based localizer and a particle filter, keeps track of the pose of the robot. When it receives a new RGB-D observation I_t, the localizer predicts a heatmap and subsequently passes it to the particle filter. Simultaneously, the particle filter receives pose discrepancy information δp_t from the odometry values of the Jackal robot. Consequently, the particle filter provides a prediction regarding the current pose p_t of the agent.

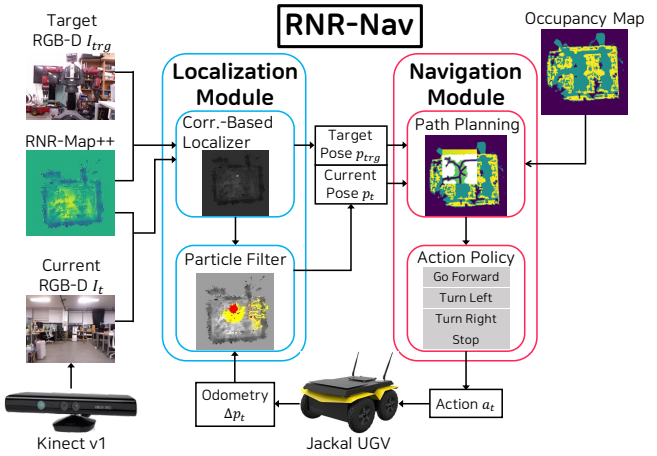


Fig. 5: **Overview of the proposed RNR-Nav navigation framework using RNR-Map++.** The localization module predicts the current pose from the current RGB-D image. The navigation module plans the path and gives the next action to the agent. An RNR-Map++ and an occupancy map need to be prepared.

The navigation module determines the next action a_t to be taken from the current pose. For safety, we employ a conservative approach by filtering out areas close to the obstacles based on the occupancy map. As the localization module provides the current pose, the planner sets a local goal and local path while ensuring adherence to the safe regions. When the robot reaches the local goal, or there is a significant change in the estimated pose compared to the previous one, the planner replans the local path. Subsequently, a suitable action a_t is determined to facilitate the path following from the current position. The decision to stop is based on the relative distance between the estimated current position and the target position. We test our robot system in real-world environments, and the results are detailed in the following section.

V. Experiments

A. Training Preparation

For training, we use the ScanNet [19] dataset, which consists of more than 1500 indoor sequences for training and 100 test sequences containing RGB-D observations and ground truth camera poses. We find that training with 500 of the 1500 sequences is sufficient for the generalizability.

We set the map size of RNR-Map++ to $(U, V) = (128, 128)$ with grid size $s = 0.25m$, which makes the map cover $32m \times 32m$ regions of the environment. To fulfill our constraint of camera parameters, $f_x = f_y = 128$, we use resized images with the shape of $(H, W) = (102, 134)$ for ScanNet images. For our robot system with Kinect v1 camera, we use $(H, W) = (113, 152)$ for input image size. We measure and report the running times of various processes using a laptop PC with an Intel i7-10875H CPU @ 2.30GHz and an NVIDIA GeForce RTX 2070 Super GPU, which is also used for our real-world RNR-Nav experiment.

TABLE I: **Rendering quality comparisons of variants of RNR-Map on ScanNet [19] test scenes.** The terms Weight and PE refer to whether using a weighted map and positional encoding for the encoder.

		Seen			Full		
Weight	PE	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
✗	✗	16.318	0.465	0.613	16.315	0.466	0.613
✓	✗	16.351	0.468	0.611	16.607	0.470	0.610
✗	✓	18.972	0.566	0.536	18.965	0.566	0.536
✓	✓	**19.113**	**0.571**	**0.533**	**19.106**	**0.571**	**0.532**

TABLE II: **Localization results on ScanNet [19] test scenes.** The term PF refers to a particle filter, and the term RR refers to a robustness ratio.

Method	$e_{dist}(m)\downarrow$	$e_{ori}(^\circ)\downarrow$	**RR(%)** ↑
Loc-NeRF [10]	1.472	44.738	44.3
RNR-Map [1]	0.375	14.671	90.1
RNR-Map w/ PF	0.156	5.804	95.7
RNR-Map++	0.293	10.552	94.5
RNR-Map++ w/ PF	**0.119**	**4.117**	**97.1**

B. Results

1) Reconstruction: We evaluate the rendering performance on the ScanNet test dataset, which consists of 100 scenes and more than 200k RGB-D images with known poses. Note that they are unseen frames during training. The entire map is constructed beforehand and subsequently used to render images at specified poses. The mapping process F_{reg} processes each image of the ScanNet dataset in just $12.1ms$, enabling operation at a speed of 82.6Hz. We only use uniformly sampled 10% of the total frames to make the maps. For rendering, we test with two subsets: *Seen* - about 20k frames that are used to construct maps, and *Full* - about 200k full frames that include seen and unseen images.

We measure and report the three widely used metrics representing image quality: PSNR, SSIM, and LPIPS. The results shown in Table I confirm that the reconstruction performance remains consistent regardless of whether the frame is observed before. It verifies that the RNR-Map++ is robust in terms of the number of images for construction. The results indicate that incorporating positional encoding (PE) has a notable impact on improving rendering performance. It suggests that maintaining vertical positional information is essential for creating visually informative maps. Additionally, the weighted map demonstrates quantitative improvements, suggesting that capturing salient features from images onto the map yields more effective results.

As we can see from the quantitative result and qualitative result from Figure 3, the RNR-Map++ shows robustness for unseen environments, such as the ScanNet test set or our own real-world observations. Unlike several NeRF-based methods that are limited to use in a single scene, our approach is applicable to any environment after training once.

2) Localization: With the trained encoder, we could make an egocentric query RNR-Map++ from a query RGB-D image. Constructed RNR-Map++ of test scenes are used as key maps. We evaluate localization performance by estimating the poses of the 200k *Full* subset for $R = 18$ angle bins. From the ground truth poses, we compute the relative pose differences for the particle filter with additional noises.

We compare our method to the localization method of Loc-NeRF [10], which uses rendering processes and a particle filter. Since the scales of ScanNet scenes are more extensive than [10] settings, and the performance inevitably depends on the number of particles, we adjust the initial number of particles proportional to the volume of each scene with a maximum limit of 500. In addition, we measure the performance of the localization process using the original version of RNR-Map and its correlation-based localizer.

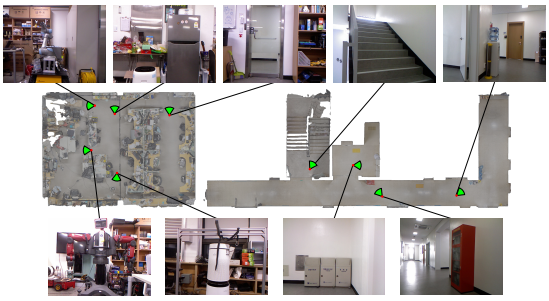


Fig. 6: **Real-world environments and various landmarks.** We set two environments: *Laboratory* and *Corridor*. Landmarks with distinct views are selected for starting points and target points.

We use the following metrics for the localization task: 1) average distance error e_{dist}, 2) average absolute value of orientation error e_{ori}, and 3) robustness rate RR, which denotes the ratio of when e_{dist} is smaller than $0.5m$. Table II provides a comprehensive overview of the results. Notably, the particle filter shows significant improvements in both accuracy and robustness. Specifically, in the case of RNR-Map++, we observe a substantial 59.4% reduction in the e_{dist} value. The high robustness rate of 97.1% indicates the applicability of the proposed localization process in diverse tasks such as navigation. Moreover, the results reveal that effectively integrating visual information into RNR-Map++ is advantageous for correlation-based localization compared to RNR-Map. In contrast, the Loc-NeRF exhibits less favorable results with a robustness rate of only 44.3%. We find that its performance is susceptible to the convergence of particles in the early stages, often displaying high levels of randomness. The proposed localization method using RNR-Map++ with particle filter shows an average inference time of $33.8ms$ (29.6Hz), confirming it is reasonable to consider as a real-time process. On the contrary, Loc-NeRF requires an average of $2340ms$ (0.43Hz) per frame, which makes it challenging to serve as a real-time process.

C. Jackal Demonstration

The visual navigation experiment is conducted in two real-world environments: *Laboratory* and *Corridor*. We select characteristic landmarks, five for the laboratory and four for the corridor, as shown in Figure 6. They pair up with each other as starting points and goal points, which gives 20 pairs of trajectories for the laboratory and 12 for the corridor. Before navigation, RNR-Map++ and the occupancy map are required. So, we first drive the Jackal robot around two environments thoroughly and record RGB-D images. After calibrating the intrinsic parameters of the Kinect v1 RGB-D camera, we gain estimated poses of the images using ORB-SLAM3 [20], which we treat as ground truth poses. With ground truth poses, we construct RNR-Map++ and the occupancy map.

TABLE III: **Navigation results in real-world scenarios.** The term Optim. refers to optimization-based localization.

Method	Environment	Success Rate (%) ↑	Loc. Time(ms) ↓
RNR-Map [1] w/ Optim.	Laboratory	20.0 (4/20)	248 (4.03Hz)
	Corridor	8.3 (1/12)	
	Total	15.6 (5/32)	
RNR-Nav (Ours)	Laboratory	95.0 (19/20)	**39.4 (25.4Hz)**
	Corridor	66.7 (8/12)	
	Total	**84.4 (27/32)**	

We test navigation performance on 32 trajectories of the proposed RNR-Nav and the original RNR-Map navigation system. For both cases, the correlation-based localizer estimates the starting point and the goal point. For RNR-Map, rendering-based localization is used to refine the estimated pose. As an output of the navigation module, we implement to return one of the four discrete actions: to move forward for $25cm$, to turn left for $10°$, to turn right for $10°$, and to stop when the distance to the goal is lower than $30cm$.

The results are shown in Table III. RNR-Nav demonstrates a high success rate in navigation within the laboratory environment, which has sufficient visually distinct objects. It facilitates the correlation-based localizer to predict the pose of the robot without confusion. The sole failure has arisen from a collision in a narrow pathway. Navigation through the corridor appears more challenging due to the prevalence of similar monotonous observations like empty walls. Nonetheless, during successful episodes, the particle filter prove invaluable, effectively refining the pose whenever the visual localizer faced uncertainties.

In contrast, the navigation framework employing the original RNR-Map and optimization-based localization exhibits a notably low success rate compared to RNR-Nav. Since the performance of the correlation-based localizer falls short of the proposed method, it often mislocates the starting points or the goal points. An incorrect prediction of the goal pose inevitably leads to a completely different trajectory. Rectifying an erroneous starting point is only possible through an optimization process, which is exceedingly challenging with limited iterations. Even when the initial and final points of the trajectory are accurately predicted, the optimizer demonstrates inaccurate outcomes since the performance depends on the rendering quality.

We also measure the speed of the localization process during navigation. The proposed localization method with the particle filter demonstrates an average execution time of $39.4ms$ (25.4Hz), which is slightly longer than the ScanNet case because of the bigger image size. It is more than six times faster than the running time of $248ms$ (4.03Hz) using rendering and optimization.

VI. Conclusion

In this paper, we have proposed a novel real-world image-goal navigation framework, RNR-Nav, based on RNR-Map++. RNR-Map++ adopts weighted map and positional encoding to reduce information loss during map construction. In addition, we have introduced a robust localization method utilizing cross-correlation of RNR-Map++ and particle filter. The experimental results show enhanced performance of RNR-Map++ and the localization method. Moreover, RNR-Nav demonstration in real-world environments achieves a high success rate.

References

[1] O. Kwon, J. Park, and S. Oh, "Renderable Neural Radiance Map for Visual Navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[2] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object Goal Navigation using Goal-Oriented Semantic Exploration," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[3] V. Blukis, C. Paxton, D. Fox, A. Garg, and Y. Artzi, "A Persistent Spatial Semantic Representation for High-level Natural Language Instruction Execution," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2022.

[4] J. F. Henriques and A. Vedaldi, "MapNet: An Allocentric Spatial Memory for Mapping Environments," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[5] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[6] T. DeVries, M. A. Bautista, N. Srivastava, G. W. Taylor, and J. M. Susskind, "Unconstrained Scene Generation with Locally Conditioned Radiance Fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[7] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson Env: Real-World Perception for Embodied Agents," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[8] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from RGB-D Data in Indoor Environments," *International Conference on 3D Vision (3DV)*, 2017.

[9] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *IEE proceedings F (radar and signal processing)*, 1993.

[10] D. Maggio, M. Abate, J. Shi, C. Mario, and L. Carlone, "Loc-NeRF: Monte Carlo Localization using Neural Radiance Fields," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[11] J. Liu, Q. Nie, Y. Liu, and C. Wang, "NeRF-Loc: Visual Localization with Conditional Neural Radiance Field," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[12] H. Wang, J. Wang, and L. Agapito, "Co-SLAM: Joint Coordinate and Sparse Parametric Encodings for Neural Real-Time SLAM," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[13] A. Rosinol, J. J. Leonard, and L. Carlone, "NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[14] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler, "Open-vocabulary Queryable Scene Representations for Real World Planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[15] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual Language Maps for Robot Navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[16] B. Bolte, A. Wang, J. Yang, M. Mukadam, M. Kalakrishnan, and C. Paxton, "USA-Net: Unified Semantic and Affordance Representations for Robot Memory," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[17] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. Szlam, "CLIP-Fields: Weakly Supervised Semantic Fields for Robotic Memory," in *ICRA2023 Workshop on Pretraining for Robotics (PT4R)*, 2023.

[18] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[19] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[20] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.