Scalable Spatial Memory for Scene Rendering and Navigation

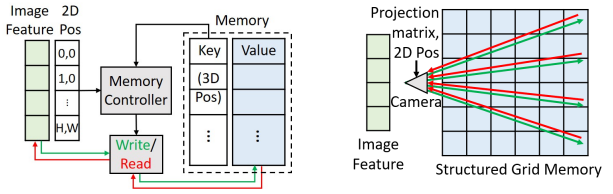
Wen-Cheng Chen[1]*, Chu-Song Chen[2], Wei-Chen Chiu[3], Min-Chun Hu[4]†
[1]National Cheng Kung University
[2]National Taiwan University
[3]National Yang Ming Chiao Tung University
[4]National Tsing Hua University
wenchengchen123@gmail.com, chusong@csie.ntu.edu.tw
walon@cs.nctu.edu.tw, anitahu@cs.nthu.edu.tw

Abstract

Neural scene representation and rendering methods have shown promise in learning the implicit form of scene structure without supervision. However, the implicit representation learned in most existing methods is non-expandable and cannot be inferred online for novel scenes, which makes the learned representation difficult to be applied across different reinforcement learning (RL) tasks. In this work, we introduce Scene Memory Network (SMN) to achieve online spatial memory construction and expansion for view rendering in novel scenes. SMN models the camera projection and back-projection as spatially aware memory control processes, where the memory values store the information of the partial 3D area, and the memory keys indicate the position of that area. The memory controller can learn the geometry property from observations without the camera's intrinsic parameters and depth supervision. We further apply the memory constructed by SMN to exploration and navigation tasks. The experimental results reveal the generalization ability of our proposed SMN in large-scale scene synthesis and its potential to improve the performance of spatial RL tasks.

Introduction

Inferring structure from images and reconstructing scenes for 3D inference are basic abilities of human visual perception. These abilities play important roles in 3D spatial tasks such as environment exploration and navigation. In recent years, neural scene representation and rendering techniques (Eslami et al. 2018; Tobin, Zaremba, and Abbeel 2019; Sitzmann et al. 2019; Sitzmann, Zollhöfer, and Wetzstein 2019; Mildenhall et al. 2020; DeVries et al. 2021; Chen, Hu, and Chen 2021) have shown promising results in learning implicit forms of scene structure from observation without explicit supervision. However, for large-scale scenes, current methods construct neural scene representations which are limited in ways that make them imperfect replacements for explicit scene structure information. For example, Generative Query Network (GQN) (Eslami et al. 2018) uses a CNN to extract 1D scene representation, but it has limited generalization ability for complex scenes. Some extended methods of GQN (Tobin, Zaremba, and Abbeel 2019; Chen, Hu, and Chen 2021) solve the generalization problem but its scene representation is non-expandable, and therefore can only be applied to the scenes of small rooms. On the other hand, Neural Radiance Field (NeRF)-based methods (Mildenhall et al. 2020; DeVries et al. 2021) store the information of scenes in the weights of neural network, which is hard to be taken as the input state in RL tasks. Furthermore, the scene representation of NeRF can be updated only via offline fine-tuning for novel scenes, making real-time scene construction demanding to achieve.



(a) Our proposed spatially aware memory control.

(b) Camera projection and back-projection.

Figure 1: Comparison between our proposed memory control process with camera projection and back-projection.

In this work, we aim to apply implicit scene representation to spatial RL tasks in large scenes. The abilities of online inference and memory expansion are important to this goal. Some researches combine memory augmented network with neural scene representation to achieve these two abilities but still have their own limitations. For example, GTM-SM (Fraccaro et al. 2018) stores each 1D image feature to a separate memory block and does not fuse the spatial information of different observations. This design makes it difficult for RL models to recognize the surrounding scene structure based on the memory. Incremental Scene Synthesis (ISS) (Planche et al. 2019) adopts the recurrent network to fuse the spatial information of different observations to a 2D structured memory, but it requires additional depth information. To solve the above shortcomings, we introduce *Scene Memory Network (SMN)* with a spatially aware memory controller. The concept of SMN is shown in Fig. 1a. It constructs memory blocks that comprise memory values representing features of partial 3D areas and memory keys

*corresponding author.
†corresponding author.

representing the 3D positions. The memory controller in SMN determines the information passing between feature map and memory blocks based on memory keys and pixel coordinates. For the propose of map expansion, we allocate new memory blocks with the keys sampled in new 3D area.

Our proposed SMN has the following advantages. First, the design of spatially aware memory control fuses information of the same local 3D area based on multiple observations and allows the expansion of scene representation for large-scale scenes. Second, the message passing process is learned from observations instead of utilizing the camera projection model (as shown in Fig. 1b). Therefore, the proposed SMN can be trained and inferred without knowing depth information and camera intrinsic parameters. Third, the memory constructed by SMN can be applied to spatial reinforcement learning models by taking the memory as a local map to supplement the insufficient part of the current observation. Forth, the entropy of memory can be used as a measure of intrinsic curiosity, which encourages the agent to explore the environment. The experimental results prove that the proposed SMN has good generalization ability for view rendering in large scenes and the memory of SMN is beneficial for improving the performance of navigation tasks, which validates the practicality of using implicit scene representation in spatial RL tasks.

Related Works

Neural Scene Representation and Rendering

The concept of neural scene representation and rendering was first proposed in GQN (Eslami et al. 2018). Later works combine GQN with spatial properties. For example, GRNN (Tung, Cheng, and Fragkiadaki 2019) constructs the voxel-based scene representation via a back-projection process and E-GQN (Tobin, Zaremba, and Abbeel 2019) constructs the feature map of query view on the epipolar line of the observation feature map. STR-GQN (Chen, Hu, and Chen 2021) models the geometry transformation as the routing process between the feature map and the world cells. The design of routing process enables STR-GQN to be trained without intrinsic camera parameters. The GQN extensions can adapt to more complex scenes, but the size of their scene representations are fixed and therefore cannot expand the map. Some researches combine external memory with GQN to achieve expandable scene representation. GTM-SM (Fraccaro et al. 2018) adopts a differentiable neural dictionary (DND) (Pritzel et al. 2017) architecture to record the features of each observation with the camera pose and retrieve the closest k neighbors of the query pose in the memory for rendering. Another work is incremental scene synthesis (ISS) (Planche et al. 2019), which incrementally constructs a 2D grid memory and adopts the camera projection/back-projection process to transform the information between feature map and memory.

In addition to GQN-based methods, Neural Radiance Field (NeRF) (Mildenhall et al. 2020) is another neural scene representation framework which records the color and density of each position via a neural network. Although the rendering results of NeRF are realistic, it can only be applied to a constrained camera pose setting such as limiting the camera to lie on the surface of a sphere, looking inwards. Generative Scene Network (GSN) (DeVries et al. 2021) learns a 2D feature as the scene prior and takes it as the input of an conditional radiance field model, which makes GSN successfully render the indoor scene for free-movement camera. However, NeRF-like models store the scene information by the weights of the neural network and can only be updated via fine-tuning, making it hard to achieve real-time computation for online scene construction. In this work, we aim to design a model to address the problems of non-expandable memory, offline inference, and the constrained viewpoint in the current neural scene representation methods.

Memory Augmented Neural Network

Neural Turing Machine (NTM) (Graves, Wayne, and Danihelka 2014) first introduced the concept of differentiable memory control and demonstrated its ability to learn the operations such as copy and reverse. Differentiable Neural Computer (DNC) is an extension of NTM, but it is more efficient in terms of memory usage. Memory Networks (Sukhbaatar et al. 2015; Miller et al. 2016) further apply differentiable memory control to QA problems in natural language processing. In the RL field, RL-NTM (Zaremba and Sutskever 2015) combines NTM with policy gradient to learn computation tasks. Neural Episodic Control (NEC) (Pritzel et al. 2017) constructs a differentiable neural dictionary (DND) with writing and lookup operations. Given a key-value pair, the writing operation appends the pair on the memory, and the lookup operation returns the weighted sum of values in the memory, whose weights are given by normalized kernel between the lookup key and the keys in the memory. FRMQN (Oh et al. 2016) combines Q-network with DND memory to learn the navigation task. In this work, we introduce the concept of spatially aware memory control in the proposed SMN model, which can achieve flexible memory expansion like DND and spatially aware information fusion similar to geometry-based methods with structured memory.

Deep Learning-based Map Reconstruction

Neural SLAM methods (Zhang et al. 2017; Chaplot et al. 2019) learn to infer the 2D occupancy grid map from RGB images, but these methods rely on the ground truth data of range sensor. SfMLearner (Zhou et al. 2017) and SfM-Net (Vijayanarasimhan et al. 2017) train the depth estimation network by the re-projection photometric error without external structure supervision. D3VO (Yang et al. 2020) is the state-of-the-art learning-based SLAM method, which combines self-supervised depth estimation with back-end non-linear optimization to achieve better performance. In addition to the above methods which construct the map of external structure, some studies focus on constructing the map in the form of implicit features. Neural Map (Parisotto and Salakhutdinov 2018) adopts a 2D structured memory to store the input feature in a memory block whose index represents the discretized 2D position of the agent. ISS (Planche

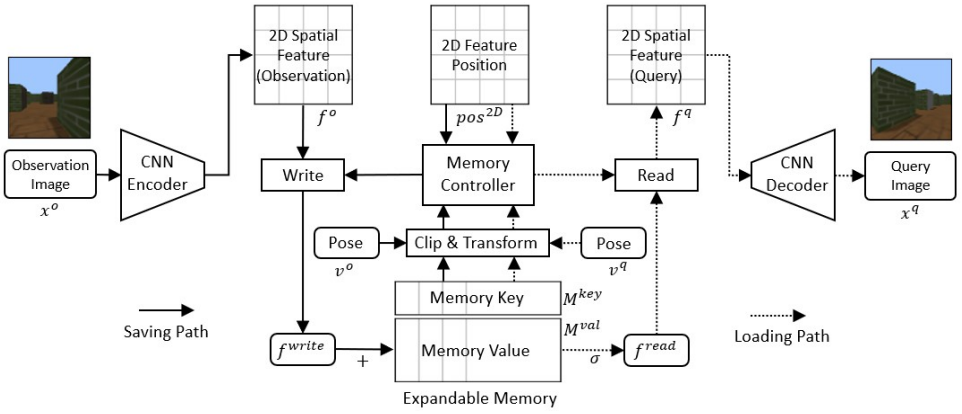


Figure 2: The architecture of the proposed scene memory network.

et al. 2019) projects the feature map on floor plane to construct the 2D structured memory. iMAP (Sucar et al. 2021) takes RGB-D sequences as input to update the neural radiance field online. Neural topological SLAM (Chaplot et al. 2020) constructs a graph-based map for the image-goal navigation. Each node in the map stores the image representation and is used for the prediction of explorable area and semantic score. The above mentioned methods requires the intrinsic camera parameters (Zhou et al. 2017; Vijayanarasimhan et al. 2017; Planche et al. 2019; Sucar et al. 2021), structure information (Zhang et al. 2017; Chaplot et al. 2019; Planche et al. 2019; Sucar et al. 2021) and semantic label (Chaplot et al. 2020) for training. In contrast, we propose SMN to construct the implicit map and the model can be trained without any of the above information.

Scene Memory Network

Basic Architecture

The architecture of the proposed SMN is illustrated in Fig. 2, which comprises two memory control processes, the *memory saving process* (denoted by solid arrows) and the *memory loading process* (denoted by dashed arrows). We adopt a CNN encoder for feature extraction and a convolutional DRAW (Gregor et al. 2016) decoder for view rendering. Let x^o denote the observation image of pose v^o, and f^o denote the 2D features extracted by the encoder given the observation image. Each memory block is composed of the value M^{val} to store partial scene representation and the key M^{key} indicating the 3D position of the memory block. The memory value M^{val} is an $N \times C$ tensor and the memory key M^{key} is an $N \times 3$ tensor, where C is a pre-defined feature dimension and N is the number of memory blocks that can be increased dynamically during inference. The *memory saving process* passes the information from the observation to the memory, which can be formulated as Eq. 1.

$$\begin{aligned} f^o &= Encoder(x^o), \\ [R, m] &= Controller(\mathcal{T}_{v^o}(M^{key})), \\ f^{write} &= Write(f^o, R, m), \\ M^{val}_{(new)} &= M^{val}_{(old)} + f^{write}, \end{aligned} \tag{1}$$

where R is the relation matrix between the pixel positions and the memory blocks, and m denotes the memory mask that decides which memory blocks are related to the control process. The details of R and m for memory control will be described in Sec. . $\mathcal{T}_{v^o}$ is the transformation that transforms the memory keys to the camera space based on the observation pose v^o.

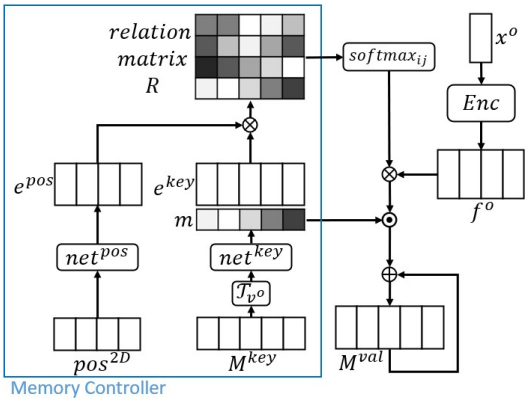
Similar to the *memory saving process*, let x^q denote the query image of pose v^q generated by the model and f^q denote the feature map of the query view. The *memory loading process* passes the information in memory to the query view, which can be formulated as Eq. 2.

$$\begin{aligned} f^{read} &= \sigma(M^{val}), \\ [R, m] &= Controller(\mathcal{T}_{v^q}(M^{key})), \\ f^q &= Read(f^{read}, R, m), \\ x^q &= Decoder(f^q). \end{aligned} \tag{2}$$

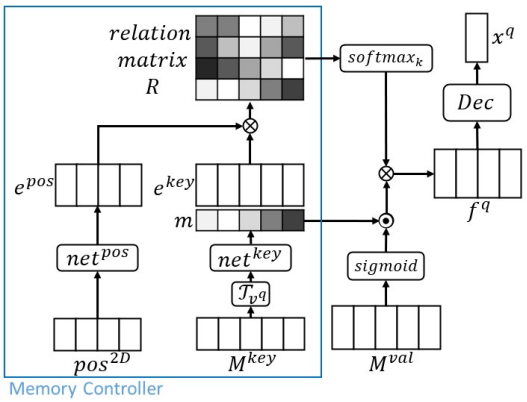
Note that we apply sigmoid operation σ to the memory value for preserving the scale of the scene representation. This operation can also be treated as transforming the posterior in the form of log-odds to the probabilistic form, as demonstrated in STR-GQN (Chen, Hu, and Chen 2021). As for the objective function, we adopt the ELBO loss proposed in GQN (Eslami et al. 2018).

Spatially Aware Memory Control

In this subsection, we introduce the concept of spatially aware memory control, where each memory block stores partial 3D area information composed of the features on 2D feature map. To represent the relation between feature map and memory blocks, we adopt a relation matrix (denoted as R) with the size $(HW) \times N$, where (H, W) are the height and width of the feature map, respectively. The image only contains the information in the field of view; thus we apply a memory mask m to represent whether each memory block is in the viewing frustum. We design a spatially aware memory controller that takes the 2D positions pos^{2D}, memory keys M^{key}, and the transformation $\mathcal{T}_v$ as input to generate the relation matrix R and memory mask m. The formulation



(a) Save.



(b) Load.

Figure 3: The details of the memory saving process (a) and memory loading process (b).

of memory controller is shown in Eq. 3 and illustrated in Fig. 3. Each memory key is first transformed to the camera space via transformation $\mathcal{T}_v$ and is then taken as the input of the key embedding network net^{key} to generate the embedding e^{key} and the mask m of each memory block. The position embedding network net^{pos} takes the 2D position pos^{2D} as the input to generate the embedding e^{pos}. The relation matrix is constructed by the inner-product of the key embedding e^{key} and position embedding e^{pos}.

$$\begin{aligned} [e_k^{key}, m_k] &= net^{key}(\mathcal{T}_v(M_k^{key})), \\ e_{ij}^{pos} &= net^{pos}(pos_{ij}^{2D}), \\ R_{ij,k} &= (e_k^{key})^T(e_{ij}^{pos}), \end{aligned} \tag{3}$$

where k in the subscript denotes the index corresponding to the kth memory block and i, j in the subscript denotes the 2D index corresponding to the 2D pixel position. Due to the fact that the processes of camera back-projection and projection are symmetric, the *memory loading process* and *memory saving process* share the same networks in memory controller (i.e., net^{key} and net^{pos}). The attention computation have a similar form of spatial transformation routing proposed in STR-GQN (Chen, Hu, and Chen 2021). Given the relation matrix R and mask m, the writing operation is defined in Eq. 4.

$$\begin{aligned} att_k(i,j) &= \frac{\exp(R_{ij,k})}{\sum_{i',j'} \exp(R_{i'j',k})}, \\ f_k^{write} &= m_k * \sum_{i,j} att_k(i,j) * f_{ij}^o. \end{aligned} \tag{4}$$

The attention map att is generated by performing the softmax operation on the relation matrix along the dimension of 2D positions. This step keeps the scale of features the same after the saving process , which enables the memory controller to be applicable to different image resolutions. After fusing the features by the attention map, an element-wise product is performed on the fused feature and the mask to construct the writing feature f^{write}. As for the reading operation, the attention map is constructed by performing the softmax operation on the relation matrix R along the dimension of the key embedding. An element-wise product is performed on the reading feature f^{read} and the mask. The product result is then weighted by the attention map to construct the feature map of the query view as in Eq. 5.

$$\begin{aligned} att_{ij}(k) &= \frac{\exp(R_{ij,k})}{\sum_{k'} \exp(R_{ij,k'})}, \\ f_{ij}^q &= \sum_k att_{ij}(k) * m_k * f_k^{read}. \end{aligned} \tag{5}$$

Memory Clipping and Efficient Training

The computational cost of the memory control process is proportional to the size of the memory blocks, and the cost become impractical as the scene size increases. A solution is only handling the memory blocks whose memory keys are close to the position of camera (i.e. memory keys within a clipping range) in the memory control process. We further propose a training trick to save the space consumption. Considering that the learning is mainly based on the prediction error of the query image and only the memory blocks within the clipping range are related to the prediction error, we can only use a small part of the memory blocks to train a query sample. In practice, we construct a training sample by composing several observation views with only one query view and then transform each observation pose to the coordinate system of the query pose. In this way, every query pose in a batch is at the origin, and hence we can use the same memory blocks in the clipping range for different training samples. Based on this training trick, the space complexity of the memory control process in training phase becomes independent of the scene size.

Agent with Scene Memory

To evaluate whether the proposed SMN can improve the performance of RL tasks, we design a model (named as SMN-DQN) that takes the memory of the proposed SMN and the observation images as the input of Q-network. We adopt Deep Q-Network (Mnih et al. 2015) as the core RL algorithm with double Q-learning loss (Van Hasselt, Guez, and Silver 2016) and dueling network structure (Wang et al. 2016). As illustrated in Fig. 4, the embedding of the memory

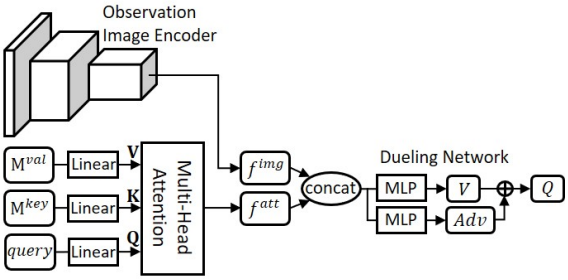


Figure 4: The architecture of the SMN-DQN agent.

(a) Scene Exploration (b) Item-Collecting

Figure 5: Rendering examples of the RL tasks in the *Orario3d* platform with the maze size of 11×11.

keys, memory values, and a learnable query vector are taken as the input of the multi-head attention module (Vaswani et al. 2017) to generate memory attention features f^{att}. In addition, an observation image encoder is adopted to extract the image features f^{img}. The memory attention features f^{att} and the image features f^{img} are concatenated and taken as the input of the dueling network to generate the Q value. Moreover, we design an intrinsic curiosity reward based on the entropy gain of the memory to make the agent explore sparse reward environments. For every step, we save the information of the observation to the memory and compute the intrinsic reward via Eq. 6.

$$
\begin{aligned}
&p_k(t) = \sigma(M_k^{val}(t)),\\
&H(t) = -p_k(t)\log p_k(t) - (1-p_k(t))\log\left(1-p_k(t)\right),\\
&r^{int}(t) = \sum_k [H(t) - H(t+1)],
\end{aligned}
\tag{6}
$$

where k denotes the index of each memory block and t denotes the time step. The weights of SMN are fixed during the training of SMN-DQN.

Experiments

Experimental Setup

Dataset for Scene Rendering. We evaluate the rendering performance of the proposed model in both small and large scenes. For small scenes, we adopted Rooms-Free-Camera (RFC) dataset proposed in GQN (Eslami et al. 2018). As for large scenes, we design and develop a 3D platform named *Orario3d* based on pyrender (Matl 2019) to procedurally generate 3D mazes with different structures and textures (as shown in Fig. 5a). A 3-DOF camera is placed on a plane parallel to xy-plane to render the RGB images. The side length of one grid in the maze is 1 meter. The size of each maze is set to 11×11 grids and the FOV of the camera is set to $80°$. Each observation/query image has 64×64 pixels.

Training Setting for Scene Rendering. For training on the RFC dataset, we sampled 2048 memory blocks with the keys in the range of $-1 \sim +1$. For training on scenes from *Orario3d*, the density of the memory blocks is set to $(10\ blocks)/m^3$, and the clipping space is set to a $(6m)^3$ cube with the agent at the center of the cube (averagely 2160 memory blocks in the clipping range). The batch size and the number of training steps are set as 32 and 1.6M, respectively. We use the Adam optimizer to train the network with a learning rate of 5e-5. All the experiments are conducted on a PC with an Intel Core i7-9700K CPU and NVIDIA Geforce GTX1080Ti GPU.

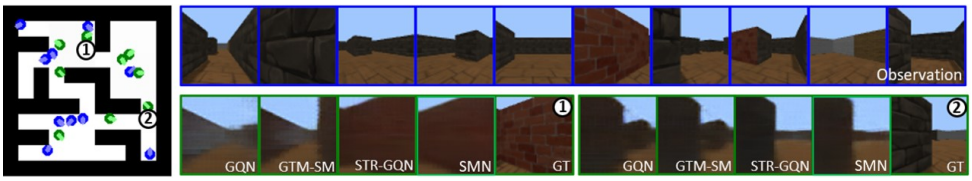
Reinforcement Learning Environments. We construct an environment with two different tasks for evaluating the RL models based on *Orario3d* platform. The agent can perform 5 discrete actions, which are {*Move Forward*, *Clockwise Rotation*, *Anticlockwise Rotation*, *Clockwise Rotation + Move Forward*, and *Anticlockwise Rotation + Move Forward*}. The two tasks are described as follows and Fig. 5 illustrates the examples of the RL environments.

- **Item-Collecting:** 15 red balls are randomly placed in the maze, and the distance between any two red balls is larger then $1m$. The agent aims to collect as many red balls as it can in 1000 steps. The agent receives one point when it collects a red ball.
- **Scene Exploration:** Only the observation images/poses are provided, and no extrinsic reward signal is considered. The agent aims to explore as much areas in 1000 steps.

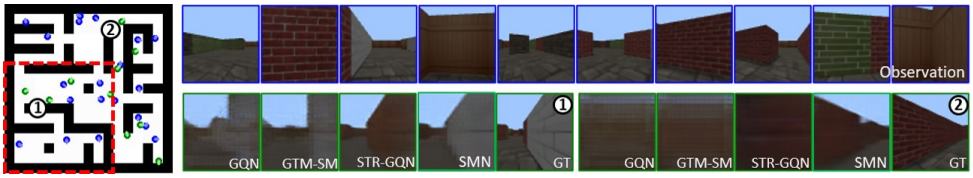
Training Setting for Reinforcement Learning. We stack the 5 previous frames to construct the input state of the image encoder in the DQN model. 1000 memory blocks are randomly sampled in the clipping range and taken as the local map of the SMN-DQN model. The discount factor is set to 0.95 and the batch size for training is set to 32. We train the RL models by using RMSprop optimizer with a learning rate of 2e-4 and 1M training steps.

Evaluation of View Synthesis

We use root mean square error between the rendered view and the ground truth as the metric of quantitative evaluation for view synthesis. We compare the proposed SMN with GQN (Eslami et al. 2018), GTM-SM (Fraccaro et al. 2018) and STR-GQN (Chen, Hu, and Chen 2021). The same as our proposed SMN, these three methods also use the prior-free setting, i.e., the view is rendered without the depth map and camera parameters. Tab. 1 shows the results of different methods on four evaluation sets. The evaluation sets "*Local*", "*Base*" and "*Large*" are collected based on the proposed *Orario3d* platform with different observation sampling strategies for evaluating different degrees of generalization ability. Each evaluation set contains 100 mazes. For "*Local*", we randomly sample 5 observation poses and 10 query poses in the 6×6 grids area of each 11×11 grids maze, which is similar to the sampling strategy of the train-



(a) Maze-Base



(b) Maze-Large

Figure 6: Comparison of the rendering results for different methods based on the Maze-Base and Maze-Large setting.

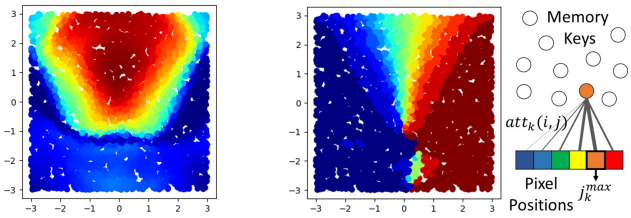
	GQN	GTM-SM	STR-GQN	SMN (Ours)
Rooms-Free-Camera	19.1±13.9	19.1±13.9	**15.9±12.8**	16.1±13.8
Orario3d (Local)	30.5±24.1	31.1±25.4	22.5±22.0	**21.3±20.7**
Orario3d (Base)	34.2±25.6	32.8±25.3	22.8±21.5	**21.7±21.6**
Orario3d (Large)	56.5±28.8	53.8±29.9	52.1±30.8	**25.8±22.5**

Table 1: The mean and standard deviation of the root mean square error for view synthesis.

ing data. For "*Base*", we randomly sample 10 observation poses and 10 query poses in overall 11×11 grids maze. In "*Large*" evaluation set, we randomly sample 20 observation poses and 10 query poses in the overall area of each 17×17 grids maze.

We can observe that GQN has the worst generalization ability. GTM-SM is better than GQN in *"Base"* due to the DND memory. Benefit from the design of spatially aware mechanism, STR-GQN has better performance on both *"Local"* and *"Base"* than GTM-SM, but both of STR-GQN and GTM-SM cannot handle the rendering of the mazes larger than the training scene. The proposed SMN has the lowest rendering error on most of the evaluation sets and is the only one that can adapt to the *"Large"* evaluation set. The reason might be that GQN, GTM-SM and STR-GQN cannot handle the data whose camera locations are outside the range of the training data. In SMN, we transform the camera poses and the memory keys together to the egocentric form and clip the memory based on the keys, which makes the input of the network in memory controller always in the same range and therefore enables the model to adapt to large scenes.

Fig. 6 illustrates the rendering results of *"Base"* and *"Large"* for different methods. The images with blue and green outer frames are the observation images and the query images, respectively. The blue and green circles on the top view of maze are the poses of observation views and query views, respectively. We visualize the rendering results of two query poses (indicated by ① and ②) for each scene to compare different methods. In the results of *"Base"*, we can observe that the rendered images of STR-GQN and SMN are reasonable but the ones of GQN and GTM-SM have wrong scene structures. As for the results of *"Large"*, we visualize



(a) Memory Mask m (b) Max Attention Index j_k^{max}

Figure 7: Visualization of the memory mask m and max attention index j_k^{max} for the corresponding memory keys.

the rendering result of query pose ① that is in the range of the maze size for training samples (11×11 grids area denoted as the red dotted square) and query pose ② that is out of that range. We can find that the result of STR-GQN is reasonable for the query pose ① but does not perform well for the query pose ②.

Visualization of the Memory Control Process

We visualize the memory mask m (Eq. 3) and the attention att (Eq. 5) generated by the memory controller to evaluate the geometric rationality of the learned memory control process. We randomly sample 4096 memory keys on the xy-plane of 3D space and place the camera at the origin facing to the positive y-axis. Fig. 7a shows the value of memory mask for each corresponding memory key. As the value increases, the color changes from blue to red. We can observe that the memory keys with high mask values correspond

	DQN	DQN+Epi.	DQN+Ent.	SMN-DQN	SMN-DQN+Ent.
Item-Collecting	8.85±4.05	9.15±3.42	9.30±3.04	9.75±3.69	**11.0±3.29**

Table 2: Quantitative evaluation of the item-collecting tasks.

(a) Examples of exploration in an 11×11 grids maze.

(b) Examples of exploration in a 17×17 grids maze.

Figure 8: Experimental results of scene exploration based on the proposed intrinsic reward.

to the front viewing area of the camera, indicating that the memory controller successfully learns the property of viewing frustum. As for the visualization of attention, we collect the pixel position having the max attention value for each memory key (i.e. $[i_k^{max}, j_k^{max}] = \arg\max_{i,j} [att_k(i,j)]$), where i and j denote the index of height and width on the observation image, respectively. As shown in Fig. 7b, the color represents the index j_k^{max} of the corresponding memory keys. As the value of j_k^{max} increases (from left to right on the observation image), the color changes from blue to red. We can observe that the left/right area in front of the camera corresponds to the pixels in left/right area on the image, respectively. Furthermore, the memory keys with the same color correspond to points on the same ray tracing line, which proves that the proposed memory controller successfully learns the concept of camera back-projection.

Evaluation of Reinforcement Learning

We conduct experiments of item-collecting tasks to quantitatively evaluate whether the proposed SMN can improve the performance of the RL model. We randomly generate 20 mazes with different initial poses and item positions for evaluating each method. Tab. 2 shows the mean and standard deviation of the scores for different methods in item-collecting task. "*DQN+Epi.*" is the method that uses DQN with one of the state-of-the-art episodic intrinsic reward proposed in NGU (Badia et al. 2020). "*DQN+Ent.*" is the method that uses DQN with the proposed entropy-gain intrinsic reward based on the learned memory of SMN as described in Eq. 6. "*SMN-DQN*" is the method that adopts the network architecture which takes memory blocks as the additional state. "*SMN-DQN+Ent.*" is the method that utilizes the network architecture of "*SMN-DQN*" with the proposed entropy-gain intrinsic reward. The experimental results show that the performance of RL model improves after adding information of memory blocks to the DQN model, which reveals the advantage of additional scene information constructed by the proposed SMN in the partial observed environment. The RL model that combines the memory blocks and the entropy-gain intrinsic reward achieves the best performance. Fig. 8 demonstrates the trajectories and the rendered images of the agent trained by the proposed intrinsic reward in the scene exploration task. We train the SMN-DQN in the 11×11 grids maze and test the agent on both 11×11 grids and 17×17 grids maze. The agent successfully generalizes the exploration strategy learned from small scenes to large scenes. The color on the local trajectory indicates the entropy-gain of that time step. We can observe that the entropy-gain is large (colored as red) when the area is unexplored and the entropy-gain is small (colored as blue) when the agent returns to a place it has observed.

Conclusion and Future Works

In this work, we propose a neural scene representation method named *Scene Memory Network (SMN)*, which can construct the scene representation online for view rendering without the depth map and camera parameters. In SMN, a spatially aware memory control mechanism is designed to capture the property of geometry transformation. The proposed memory controller allows expandable memory, which scales linearly with the size of the map rather than the number of observations. The memory constructed by the SMN can further assist navigation and exploration tasks. A limitation of our work is that our model can only be applied on synthetic environments with simple scene structures. Our future work is to improve the proposed SMN to adapt to realistic scenes with complex textures and lighting conditions.

Acknowledgments

This work was supported in part by the National Science and Technology Council, Taiwan under Grants NSTC 108-2221-E-007-106-MY3, NSTC 111-2634-F-002-023, MOST 110-2221-E-002-185-MY2, and in part by an unrestricted gift from Google.

References

Badia, A. P.; Sprechmann, P.; Vitvitskyi, A.; Guo, D.; Piot, B.; Kapturowski, S.; Tieleman, O.; Arjovsky, M.; Pritzel, A.; Bolt, A.; et al. 2020. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*.

Chaplot, D. S.; Gandhi, D.; Gupta, S.; Gupta, A.; and Salakhutdinov, R. 2019. Learning To Explore Using Active Neural SLAM. In *International Conference on Learning Representations*.

Chaplot, D. S.; Salakhutdinov, R.; Gupta, A.; and Gupta, S. 2020. Neural topological slam for visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12875–12884.

Chen, W.-C.; Hu, M.-C.; and Chen, C.-S. 2021. STR-GQN: Scene Representation and Rendering for Unknown Cameras Based on Spatial Transformation Routing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5966–5975.

DeVries, T.; Bautista, M. A.; Srivastava, N.; Taylor, G. W.; and Susskind, J. M. 2021. Unconstrained Scene Generation with Locally Conditioned Radiance Fields. *arXiv preprint arXiv:2104.00670*.

Eslami, S. A.; Rezende, D. J.; Besse, F.; Viola, F.; Morcos, A. S.; Garnelo, M.; Ruderman, A.; Rusu, A. A.; Danihelka, I.; Gregor, K.; et al. 2018. Neural scene representation and rendering. *Science*, 360(6394): 1204–1210.

Fraccaro, M.; Rezende, D.; Zwols, Y.; Pritzel, A.; Eslami, S. A.; and Viola, F. 2018. Generative temporal models with spatial memory for partially observed environments. In *International Conference on Machine Learning*, 1549–1558. PMLR.

Graves, A.; Wayne, G.; and Danihelka, I. 2014. Neural Turing Machines. *arXiv preprint arXiv:1410.5401*.

Gregor, K.; Besse, F.; Jimenez Rezende, D.; Danihelka, I.; and Wierstra, D. 2016. Towards conceptual compression. *Advances In Neural Information Processing Systems*, 29: 3549–3557.

Matl, M. 2019. Pyrender. https://github.com/mmatl/pyrender. Accessed: 2021-04-20.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, 405–421. Springer.

Miller, A. H.; Fisch, A.; Dodge, J.; Karimi, A.-H.; Bordes, A.; and Weston, J. 2016. Key-Value Memory Networks for Directly Reading Documents. In *EMNLP*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Oh, J.; Chockalingam, V.; Lee, H.; et al. 2016. Control of memory, active perception, and action in minecraft. In *International Conference on Machine Learning*, 2790–2799. PMLR.

Parisotto, E.; and Salakhutdinov, R. 2018. Neural Map: Structured Memory for Deep Reinforcement Learning. In *International Conference on Learning Representations*.

Planche, B.; Rong, X.; Wu, Z.; Karanam, S.; Kosch, H.; Tian, Y.; Ernst, J.; and HUTTER, A. 2019. Incremental Scene Synthesis. *Advances in Neural Information Processing Systems*, 32: 1668–1678.

Pritzel, A.; Uria, B.; Srinivasan, S.; Badia, A. P.; Vinyals, O.; Hassabis, D.; Wierstra, D.; and Blundell, C. 2017. Neural episodic control. In *International Conference on Machine Learning*, 2827–2836. PMLR.

Sitzmann, V.; Thies, J.; Heide, F.; Nießner, M.; Wetzstein, G.; and Zollhofer, M. 2019. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2437–2446.

Sitzmann, V.; Zollhöfer, M.; and Wetzstein, G. 2019. Scene representation networks: continuous 3D-structure-aware neural scene representations. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 1121–1132.

Sucar, E.; Liu, S.; Ortiz, J.; and Davison, A. J. 2021. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6229–6238.

Sukhbaatar, S.; Weston, J.; Fergus, R.; et al. 2015. End-To-End Memory Networks. *Advances in Neural Information Processing Systems*, 28: 2440–2448.

Tobin, J.; Zaremba, W.; and Abbeel, P. 2019. Geometry-aware neural rendering. *Advances in Neural Information Processing Systems*, 32: 11559–11569.

Tung, H.-Y. F.; Cheng, R.; and Fragkiadaki, K. 2019. Learning spatial common sense with geometry-aware recurrent networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2595–2603.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Vijayanarasimhan, S.; Ricco, S.; Schmid, C.; Sukthankar, R.; and Fragkiadaki, K. 2017. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*.

Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, 1995–2003. PMLR.

Yang, N.; Stumberg, L. v.; Wang, R.; and Cremers, D. 2020. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1281–1292.

Zaremba, W.; and Sutskever, I. 2015. Reinforcement Learning Neural Turing Machines-Revised. *arXiv e-prints*, arXiv–1505.

Zhang, J.; Tai, L.; Liu, M.; Boedecker, J.; and Burgard, W. 2017. Neural slam: Learning to explore with external memory. *arXiv preprint arXiv:1706.09520*.

Zhou, T.; Brown, M.; Snavely, N.; and Lowe, D. G. 2017. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1851–1858.