Learning Temporally Consistent 3D Robot Mapping

Weining Ren[*]
ETH Zürich
`weiren@student.ethz.ch`

Haoran Chen[*]
ETH Zürich
`chenhao@student.ethz.ch`

Abstract: In multi-agent tasks, shared environments are often subject to long-term scene changes. However, current neural implicit representation methods for 3D reconstruction mainly focus on static environments and cannot capture temporal changes. In this work, we propose a encoder-decoder structure network with 3D GRU fusion layer to capture such temporal changes. We introduce free-space encoding to distinguish unobserved space and free space. A novel fusion training strategy to proposed to train the network on dynamic datasets. Compared to previous methods designed for static scenes, our scene representation is more robust in temporal changes and have the potential to achieve real time performance on a GPU. We demonstrate our approach on real-world dataset with different settings.

Keywords: Neural implicit representation, long-term scene change, free-space encoding

1 Introduction

As robot becomes increasingly common in people's daily life, it's essential for robots to have a consistent understanding of surroundings in dynamic scenes. In many applications like long-time autonomous exploration or multi-agent tasks, the environment is inevitably subject to long-term scene changes. Accounting for such changes is important for the robot tasks like path planning and object avoidance.

In the field of 3D reconstruction, explicit represent the environment with voxel or pointcloud has achieved a lot of success, such as occupancy based method [1] or Truncated Singed Distance Fields(TSDF) based method [2]. However, such explicit representation method is generally based on a fixed grid division, which makes it memory expensive and inflexible for scenes changes. Recently, implicit representation has gained popularity due to its flexibility and expressiveness. Neural radiance fields(NeRF) [3] achieves impressive performance in novel view synthesis by encoding the space into radiance fields and then decode it with volume rendering techniques. ONet [4], DeepSDF [5] represent the scene with implicit function and then decode it to occupancy probabilities or SDF values.

However, most existing neural implicit representation methods mainly focus on static scenes and work poorly on dynamic ones. As shown in Figure 1, they tend to merge irrelevant objects and keep removed objects. In this work, we aim to present a 3D robot mapping method that can handle such long-term scene changes and perform efficient 3D mapping through RGB-D image series.

This work is based on NeuralBlox [6], which utilizes ConvONet [7] as backbone and averages all encoded history observations in a latent space. A 3D convolutional layer is introduced to process the averaged latent. We find that the free-space information is crucial for 3D reconstruction under dynamic scenes, without which the network cannot tell the difference between free-space and unobserved space. Noticing the free space shares the same representation with occupancy space, we design a dual encoder structure that encode occupancy space and free space into latent space separately. In contrast to the average pattern in NerualBlox, we propose a 3D GRU (Gated recurrent

[*]These authors contributed equally to this work.

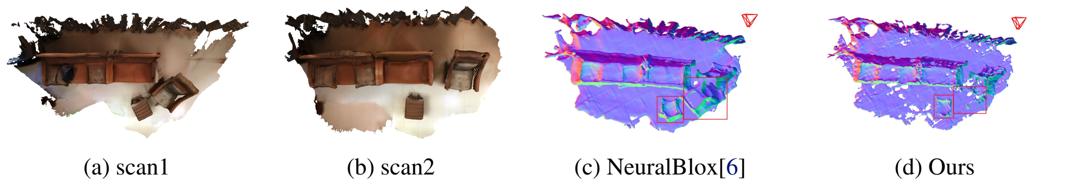


Figure 1: Reconstruction result directly running scan 2 after scan 1. Our method could capture the movement of stool and chair, but NeuralBlox keep all history observation

unit) [8] fusion layer to fuse current observation with history latent. We expect the update gate and reset gate structure can strike a balance between incrementally building the surroundings and deleting removed objects. However, such balance is hard to find. To solve this problem, we provide a new training strategy that utilizes training sequences on voxel level. We also introduce a observation memory function to stress more on current observation.

In summary, our contributions are as follows:

- We propose a new network structure with a a dual encoder for free space and occupancy space. A 3D GRU layer is also introduced to capture temporal changes.
- We provide a new training pipeline for dynamic scene reconstruction, which randomly samples from different scans of the same scene on a voxel level to accelerate the training process.
- Our method is demonstrated on real world dataset.

2 Related Work

Neural Implicit Representation: Recently, many works show that learning-based continuous implicit functions have superior performance than explicit representations [2, 9, 10]. The key idea of implicit function is that we can encode the input data into a latent code, which can be used to predict a continuous occupancy probability space [4] or SDF value space [5]. However, due to the limitation of simple MLPs, these methods are limited to object-level reconstruction and cannot scale to large scenes. Peng et al. [7] introduce convolution layers to incorporate inductive biases such as translational equivariance and thus is scalable to large scene completion. Similarly, Jiang et al. [11] achieves great scalability by leveraging geometry priors that the most 3D surfaces share geometric details at some scale. Although these method achieve great achievement in reconstruct large scale scenes, they take the input as a whole and cannot incrementally update the implicit function. Lionar et al. [6] adopt an average pattern for the history observation and use a translator network to deal with the pose uncertainty. [12] shares similar idea with [6] but uses a fusion module instead of the average pattern. Yan et al. [13] understand the stream video input from a continual learning perspective, and propose an experience replay approach to tackle the catastrophic forgetting problem. However, all of these methods only focus on static scenes and cannot deal with dynamic changes.

Recently, NeRF [3] has gained a lot of success in novel view synthesis, and its proposed volume rendering technique is widely used in neural reconstruction. Azinović et al. [14] propose a hybrid scene representation by introducing TSDF information. However, traditional NeRF-like method cannot be trained on real time and thus not applicable in robot application. iMap [15] firstly shows that with such technique the implicit representation can be trained in a real-time setting for SLAM system. NICE-SLAM [16] builds on iMAP by introducing a voxel grid of latent instead of a single global model. Although Nice-SLAM considered the influence of dynamic objects, but it just neglects all changes sample and only keep the static part. In contrast, our system can capture the long-term scene changes and can almost run at real time.

Reconstruction under Dynamic Scenes: Explicit representation method has gain some success in capturing long-term scene changes. Finman et al. [17] construct multiple dense maps and detect changes by performing a 3D difference of the scenes. Similarly, [18] use a mutli-layer TSDF representation and also perform a post checking of existing scenes. Although such post-hoc processing method can successfully detect scene changes, but building such multiple maps is time consuming and not applicable in real time robot task.

For the real time long-term robot mapping method, Tang et al. [19] proposes a topological local-metric framework, which encodes relative relationship of submap into a scene graph. Macenski et al. [20] add a spacial decay to each voxel to capture and give higher weight for current observations. Schmid et al. [21] introduce panoptic segmentation into the multi-layer TSDF submap. It could preserve the semantic consistency in long-term scene changes. But these methods are based on explicit representation and thus have inferior expressiveness and flexibility compared with implicit representation.

Recent works [22, 23] show the potential that neural implicit representations can be applied for dynamic scenes. They reconstruct a canonical pose of the object and use a network to predict the deformation of canonical pose. NeRF-W [24] deletes the transient objects by predicting of photometric uncertainty of each pixel and use such uncertainty as a weight to train the network. Niemeyer et al. [25] proposes to decouple shape and motion of objects and predict the motion of the object separately. However, these methods is mainly designed for the short-term changes and the local movement assumption is not applicable for long-term changes. To the best of our knowledge, our method is the first work to use neural implicit representation to capture long-term scene changes.

3 Method

An overview of the network pipeline is shown in Figure 2a. Given the input sequence from a RGB-D camera with known poses, the network aims to generate a consistent occupancy map under temporal changes. With such map, robots are enabled to perform tasks based on the accurate understanding of environments.

We divide the map of 3D space into voxels and represent the local scene in every voxel by a latent code. Our network dynamically updates the latent codes when new surface points p^t_{occ} are observed in the corresponding voxels. In contrast to [6], we also consider points in the free space p^t_{fs} between the observed surface and the camera to distinguish unobserved areas and free space .

When a new observation is given, we firstly encode p^t_{occ}, p^t_{fs} separately with a same encoder into z^t_{occ}, z^t_{fs}. With the prior knowledge h^{t-1} of the voxel, we then fuse the current observation of the scene information using the 3D GRU fusion layer and update the knowledge of the scene h^t, which is also the current estimation of the environment $\hat{z}^t_{occ}$.

Finally, we decode the occupancy of 3D points via a decoder conditioned on the latent code $\hat{z}^t$. With the occupancy map, we can reconstruct a high-resolution mesh using marching cubes algorithm [26].

The rest of this section describes our network architecture and a novel fusion training strategy to capture changes in dynamic scenes. We introduce free space encoding and a 3D GRU fusion layer in our network architecture, and elaborate the observation memory function in the fusion training strategy.

3.1 Network Architecture

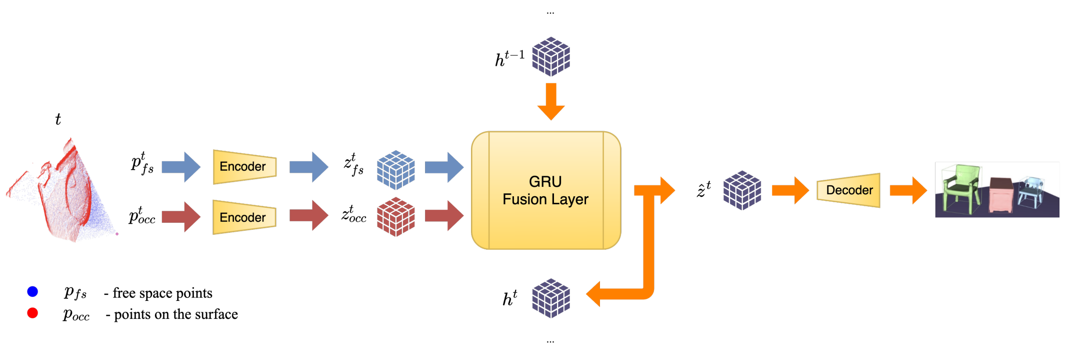
Our network uses the same backbone as [7] with free space encoding and a 3D GRU fusion layer.

We denote the encoder with weights θ_e as F_{θ_e}. Given the input point clouds p^t, the latent code is generated as:

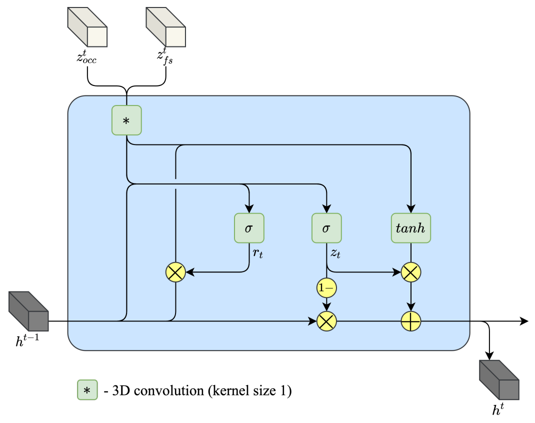
$$z^t_{occ} = F_{\theta_e}(p^t_{occ}) \tag{1}$$

We denote the decoder with θ_d as G_{θ_d}. With a latent code z^t_{occ}, the decoder outputs the occupancy map of the voxel V_{Occ} of the query points q:

$$V_{occ}(q|z^t_{occ}) = G_{\theta_d}(q, z^t_{occ}) \tag{2}$$



(a) Network pipeline. For every time step, we extract surface points p^t_{occ} and free space points p^t_{fs}, which are sampled between the surface and the camera. Then, we encode p^t_{occ}, p^t_{fs} separately and fuse them with a 3D GRU fusion layer to build temporarily consistent latent codes for the scene. This latent code can be decoded into a occupancy map at any time step.



(b) GRU fusion layer. The latent codes z^t_{occ}, z^t_{fs} are firstly combined with a 3D convolution network with kernel size 1, then followed by a 3D convolutional GRU network to output the prediction h^t ($\hat{z}^t$) of current scene.

Figure 2: Overview of our method.

Free Space Encoding: Our network takes surface points together with points in the free space as input. Free space contains information for distinguishing unobserved areas and free space. Without free space encoding, cleared voxels cannot be detected and included in the fusion step.

The latent code for free space is obtained using the same encoder for surface points since representing free space has the same math formulation as representing occupancy:

$$z^t_{fs} = F_{\theta_e}(p^t_{fs}), \quad V_{fs}(q|z^t_{fs}) = G_{\theta_d}(q, z^t_{fs})$$

3D GRU Fusion Layer: Figure 2b presents our 3D GRU fusion layer. The goal of this layer is to output the current estimation of the scene given new partial observations. The fusion layer T_{θ_f} takes the current observations z^t_{occ}, z^t_{fs} as input, which are encoded by a same encoder F_{θ_e}, and outputs the fused latent codes z^t_{occ} based on the previous understanding of the scene h^{t-1}.

$$h^t = T_{\theta_f}(z^t_{occ}, z^t_{fs}, h^{t-1}) \tag{3}$$

$$\hat{z}^t_{occ} = h^t \tag{4}$$

We use a proven structure, Gate Recurrent Unit (GRU) [27], to capture the hidden logic within a sequence. In our case, we utilize GRU to understand the long-term change in an environment. Since

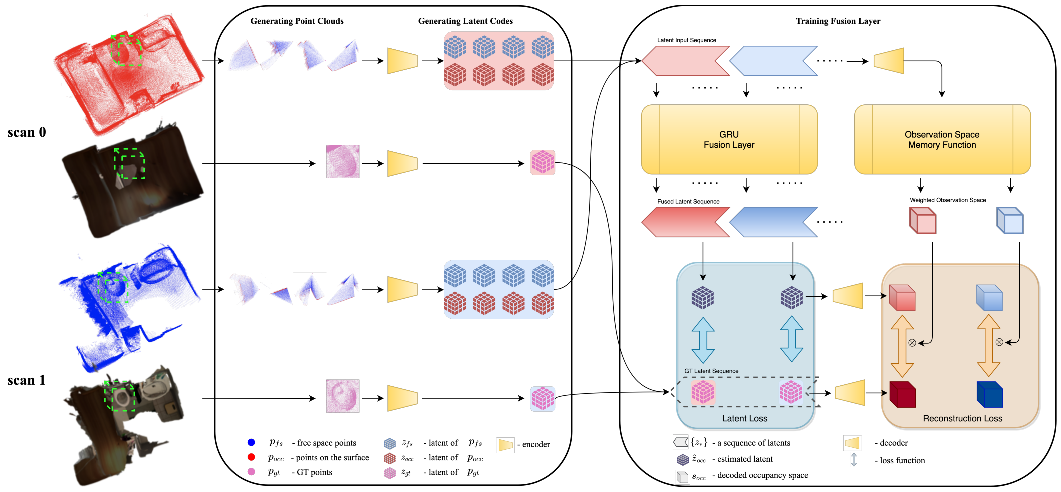

scan 0
scan 1
Generating Point Clouds
Generating Latent Codes
Training Fusion Layer
Latent Input Sequence
GRU Fusion Layer
Observation Space Memory Function
Fused Latent Sequence
Weighted Observation Space
GT Latent Sequence
Latent Loss
Reconstruction Loss
p_{fs} - free space points
p_{occ} - points on the surface
p_{gt} - GT points
z_{fs} - latent of p_{fs}
z_{occ} - latent of p_{occ}
z_{gt} - latent of p_{gt}
- encoder
$\{z_*\}$ - a sequence of latents
$\hat{z}_{occ}$ - estimated latent
s_{occ} - decoded occupancy space
- decoder
- loss function

Figure 3: Fusion Training Strategy.

the latent code represents the geometric information in 3D world, we replace every forward pass in GRU with a 3D convolutional network to fully perceive the environments.

3.2 Fusion Training Strategy

Our training strategy of the 3D GRU fusion layer is presented in Figure 3. We firstly train the encoder and decoder on a large dataset ShapeNet [28] as Peng et al. [7] did to capture geometry priors. Next we fix the backbone and train the 3D GRU fusion layer T_{θ_f} on RIO [29], a real-world dataset of changing indoor environments. Next we will introduce our training strategy for fusion layer.

Data Generation: RIO [29] features every scene with multiple scans to cover changes of the environments. Based on RIO, we build our training data on voxel level. In every scene, we divide the 3D world into voxels, which are randomly selected for data generation. For the input data of a voxel, we arbitrarily pick the depth frames in one scan that capture this voxel and sample surface points and free space points based on the known camera poses, denoted as $\{p_{occ}^{t_0}, p_{occ}^{t_1}, p_{occ}^{t_2}, \cdots\}$ and $\{p_{fs}^{t_0}, p_{fs}^{t_1}, p_{fs}^{t_2}, \cdots\}$ respectively. For the corresponding ground truth data, we directly extract dense pointcloud from the ground truth meshes, denoted as p_{gt}. Then, we generate a segment of latent codes from these pointclouds, denoted as $seg_{occ} := \{z_{occ}^{t_0}, z_{occ}^{t_1}, z_{occ}^{t_2}, \cdots\}$, $seg_{fs} := \{z_{fs}^{t_0}, z_{fs}^{t_1}, z_{fs}^{t_2}, \cdots\}$ and z_{gt}. Finally, we randomly concatenate latent segments from different scans of a same environment to form a latent sequence, denoted as $seq_{occ} := \{seg_{occ}^{scan_0}, seg_{occ}^{scan_1}, \cdots\}$, $seq_{fs} := \{seg_{fs}^{scan_0}, seg_{fs}^{scan_1}, \cdots\}$ and $\{z_{gt}^{scan_0}, z_{gt}^{scan_1}, \cdots\}$. In our example, every sequence contains 6 segments, while every segment contains 4 frames.

Training Step: Similar to [6], we compute latent loss $\mathcal{L}_l$ and reconstruction loss $\mathcal{L}_r$ to both optimize the estimation $\hat{z}_{occ}$ in latent space and voxel occupancy space, which are calculated at the end of every segment.

$$\mathcal{L}_l = \|\hat{z}_{occ}^t - z_{gt}^t\|_1 \tag{5}$$

$$\mathcal{L}_r = \frac{1}{|\mathbb{Q}|} \sum_{q \in \mathbb{Q}} \|G_{\theta_d}(q, z_{occ}^t) - G_{\theta_d}(q, z_{gt}^t)\|_1 \tag{6}$$

Moreover, in the training step, we introduce observation space memory function to give different weights in the voxel space. Usually, a voxel is not entirely observed within the field of view of the camera. This function is added to guide the fusion layer to focus on the observation area and

	scene 1	scene 2	scene 3	scene 4	scene 5
FPS(voxel=0.6m)	6.34	6.28	9.34	4.85	4.74
FPS(voxel=1.0m)	13.53	14.08	19.93	11.06	10.67

Table 1: Run time on RIO dataset

keep the unobserved space unchanged. In addition, we use a decay memory mechanism for every segment to simulate giving more attention on the current observation. With the observation memory, we apply the weighted reconstruction loss $\mathcal{L}_r^w$ in our training step.

$$\mathcal{M}(t) = \alpha\mathcal{M}(t-1) + \mathcal{I}\left(G_{\theta_d}(q, z_{occ}^t) - \mathbf{p}_{th}\right) \cup \mathcal{I}\left(G_{\theta_d}(q, z_{fs}^t) - \mathbf{p}_{th}\right) \tag{7}$$

$$\mathcal{L}_r^w = \mathcal{L}_r \otimes \mathcal{N}(\mathcal{M}) \tag{8}$$

Eq. 7 formulates the evolution of the memory function, where $\mathcal{M}$ represents the memory weights, α represents the memory decay factor, $\mathcal{I}(x)$ equals 1 if $x > 0$ otherwise 0, $\cup$ represents the union function and $\mathbf{p}_{th}$ represents the threshold probability. Eq. 8 formulates the memory-weighted reconstruction loss, where $\otimes$ represents Hadamard product and $\mathcal{N}$ is a normalizing function.

4 Experimental Results

We tested our method on real-world datasets and mainly compare the performances of our method and NeuralBlox [6]. In the rest of this section, we presented the qualitative comparison between the reconstruction results in indoor changing scenes, and reported the runtime of the fusion layer under different resolutions. We also conducted an ablation study to investigate the role of our proposed observation space memory function. In addition, we did some extra experiments to explore the reasons that cause the incompleteness of reconstruction quality using our method.

4.1 Qualitative Results

We evaluated our method on RIO [29]. For every indoor scene, we concatenated the depth image stream from different scans to form a long image stream that includes long-term changes.

From the results shown in Figure. 4, we can see that our method is able to detect temporal changes in the environments and reconstruct new objects with current observations. However, it is obvious that our method is not comparable to NeuralBlox in terms of the reconstruction quality and completeness.

4.2 Runtime of the 3D GRU Fusion Layer

We choose 5 test scenes in RIO dataset to evaluate the efficiency of our algorithm. Yet there is space for improvement, our algorithm could run in real-time with voxel size $1.0m$ as shown in Table 1. But increasing the volume size will inevitably hamper the reconstruction quality. It's important to find a good balance between accuracy and efficiency.

4.3 Ablation Study

Figure 5 shows the reconstruction results of an indoor changing scene using different approaches under the same setting. From the results, we can see that NeuralBlox performs poorly in terms of detecting temporal changes in the environment and fails to delete the removed objects from the previous scan. In contrast, our proposed 3D GRU fusion layer is able to detect the changing area. However, without the observation space memory function, the network confuses with the unobserved space with free space, thus outputs an over-forgetting result. We also investigate different settings for the memory function. By giving equal memory weights in the observation space, the

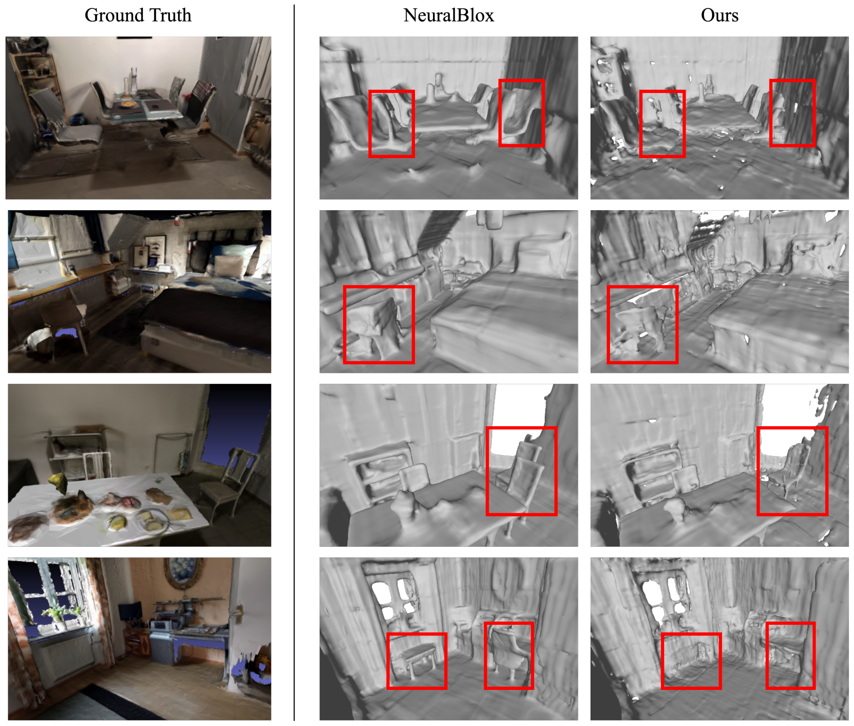


Figure 4: Reconstruction results of our method and NeuralBlox.

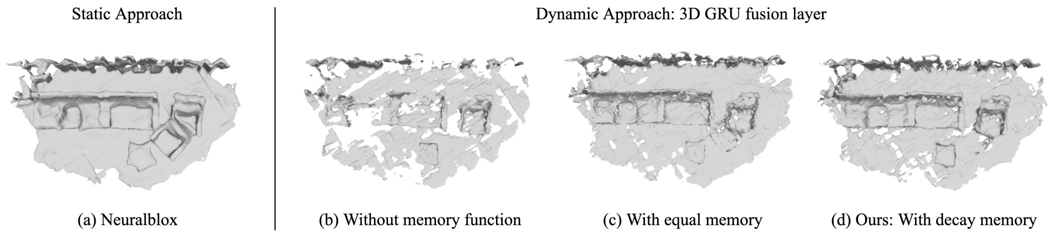

Static Approach
Dynamic Approach: 3D GRU fusion layer
(a) Neuralblox
(b) Without memory function
(c) With equal memory
(d) Ours: With decay memory

Figure 5: Reconstruction results of an indoor changing scene.

network can reconstruct the scene with a good quality but tend to remain the removed objects. We also test our network with a decay memory and show that it performs the best in terms of the balance between reconstruction quality and scene change detection.

4.4 Extra Exploration

From the above experiments, we observed that our method tends to forget unobserved area and validated our assumption by calculating *forget loss* $\mathcal{L}_f$ through our training process, which describes the unexpected forgetting behaviors of the network in the unobserved area.

We did not optimize $\mathcal{L}_f$, instead we only used it as a measurement of forgetting behaviors. From the evolution of $\mathcal{L}_f$ shown in Figure 6, we noticed that the network tends to forget more as the training steps grow though it reconstruct better in the observed area.

We also tested optimizing the network with $\mathcal{L}_f$. However, the network performed similarly to the fusion layer with equal memory in our ablation study. This can result from the incomplete utilization of the knowledge from free space. Thus, the network is still confused with unobserved areas and free space.

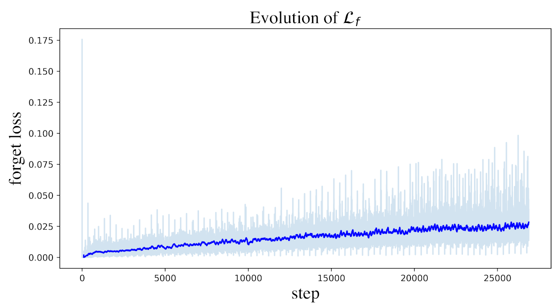


Figure 6: The evolution of $\mathcal{L}_f$ during the training process.

5 Discussion

Our proposed method can successfully detect the temporal changes in an environment. However, our method is not robust in terms of reconstruction quality and cannot generalize well in different datasets. We aim to improve our method and put forward the following directions:

Self-Supervised Holes Filling: Inspired by Sg-nn [30], which randomly create some holes from ground truth mesh as input and train a network to fill them in a self-supervised manner. For the future work, we could add a geometry refinement module after the 3D GRU fusion layer and train such a module separately.

Uncertainty Detection: Our current method uses observation space memory function as guidance to train the network focus on the current observation. However, the reconstruction result is subject to the parametrization of the memory function. In addition, current logic behind the memory function is to stay the same with the newest observation, which makes the network tendd to forget areas that are observed before but not included in the newest observation. To improve our method, we expect to give weights in the voxel space according to the estimation of uncertainty, which can be learned and embedded in our network.

Observation Map: Through our experiments, we notice that our network sometimes tend to delete objects that are not in observation area, which means the network is still confused by the unknown area and the clear area. This can result from partly understanding the information of free space. For future work, we assume a dual structure that incorporates surface points and free space points equally can fully utilize the knowledge of free space and generate the observation space at the same time.

Evaluation Metric: Reconstructing long-term changing environments is new challenge in 3D mapping tasks and lacks a uniform and systematic evaluation method. Conventional metrics, like Chamfer distance used in static 3D reconstruction tasks, are no longer proper, since it's hard to compare the growing and changing reconstruction result with ground truth data. It is expected to come up with a suitable metric to evaluate the reconstruction quantitatively in the future.

6 Conclusion

We developed an online 3D mapping method based on neural scene representations that can continuously update the geometric knowledge of a random size environment. We proposed a 3D GRU fusion layer and integrated free space encoding in our method to capture long-term dynamics from a sequence of partial observations. We also proposed a novel fusion training strategy by leveraging observation space memory function as an attention mechanism to focus on the current observation to generate reasonable 3D reconstructions. We demonstrated our algorithm on long-term changing indoor environments and showed that our system can run efficiently on GPU. Moreover, we indicated some drawbacks of our current method and discussed the potential directions to improve our system.

Acknowledgments

Thanks for the great ideas and inspirations from Lukas Schmid and Javier Rodriguez Puigvert for this project. Also thanks for the support from Cesar Cadena and Jen Jen Chung, and many valuable suggestions from the researchers and PhD students of ASL.

References

[1] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013.

[2] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[3] 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. *CoRR*, 2020.

[4] L. M. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. *CoRR*, 2018.

[5] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[6] S. Lionar, L. Schmid, C. Cadena, R. Siegwart, and A. Cramariuc. Neuralblox: Real-time neural representation fusion for robust volumetric mapping. In *2021 International Conference on 3D Vision (3DV)*, pages 1279–1289. IEEE, 2021.

[7] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. Convolutional occupancy networks. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 523–540, 2020.

[8] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[9] G. Gkioxari, J. Malik, and J. Johnson. Mesh r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[10] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011.

[11] C. M. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. A. Funkhouser. Local implicit grid representations for 3d scenes. *CoRR*, abs/2003.08981, 2020. URL https://arxiv.org/abs/2003.08981.

[12] S. Weder, J. L. Schonberger, M. Pollefeys, and M. R. Oswald. Neuralfusion: Online depth fusion in latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3162–3172, June 2021.

[13] Z. Yan, Y. Tian, X. Shi, P. Guo, P. Wang, and H. Zha. Continual neural mapping: Learning an implicit scene representation from sequential observations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15782–15792, 2021.

[14] D. Azinović, R. Martin-Brualla, D. B. Goldman, M. Nießner, and J. Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

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

[16] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[17] R. Finman, T. Whelan, M. Kaess, and J. J. Leonard. Toward lifelong object segmentation from change detection in dense rgb-d maps. In *2013 European Conference on Mobile Robots*, pages 178–185, 2013. doi:10.1109/ECMR.2013.6698839.

[18] M. Fehr, F. Furrer, I. Dryanovski, J. Sturm, I. Gilitschenski, R. Siegwart, and C. Cadena. Tsdf-based change detection for consistent long-term dense reconstruction and dynamic object discovery. In *2017 IEEE International Conference on Robotics and automation (ICRA)*, pages 5237–5244. IEEE, 2017.

[19] L. Tang, Y. Wang, X. Ding, H. Yin, R. Xiong, and S. Huang. Topological local-metric framework for mobile robots navigation: a long term perspective. *Autonomous Robots*, 43(1):197–211, 2019.

[20] S. Macenski, D. Tsai, and M. Feinberg. Spatio-temporal voxel layer: A view on robot perception for the dynamic world. *International Journal of Advanced Robotic Systems*, 17(2): 1729881420910530, 2020.

[21] L. Schmid, J. Delmerico, J. Schönberger, J. Nieto, M. Pollefeys, R. Siegwart, and C. Cadena. Panoptic multi-tsdfs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency. *arXiv preprint arXiv:2109.10165*, 2021.

[22] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021.

[23] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021.

[24] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *CoRR*, 2020.

[25] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *International Conference on Computer Vision (ICCV)*, 2019.

[26] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, aug 1987. ISSN 0097-8930.

[27] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[28] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical report, 2015.

[29] N. N. F. T. M. N. Johanna Wald, Armen Avetisyan. Rio: 3d object instance re-localization in changing indoor environments. In *Proceedings IEEE International Conference on Computer Vision (ICCV)*, 2019.

[30] A. Dai, C. Diller, and M. Nießner. Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2020.