MuSR: MULTI-SCALE 3D SCENES RECONSTRUCTION BASED ON MONOCULAR VIDEO

Han Gao, Hao Wu, Peiwen Dong, Yixin Xu, Fengyuan Xu, and Sheng Zhong*

National Key Lab for Novel Software Technology, Nanjing University

ABSTRACT

Three-dimensional (3D) scene reconstruction, particularly from monocular videos, is a significant challenge in large-scale scenarios due to difficulty handling varying object sizes and high computational resource needs. This paper introduces MuSR, a novel multi-scale reconstruction method addressing these issues. MuSR features a dynamic multi-resolution spatial structure that adaptively adjusts voxel resolution for objects of different sizes to improve reconstruction quality. MuSR also employs a block-based sparse 3D data structure and hardware resource management strategy to reduce GPU memory usage while maintaining efficient reconstruction. Evaluated on ScanNet and 7-Scenes datasets, as well as real-world scenes, MuSR outperforms state-of-the-art methods in terms of efficiency, completeness, and geometric shape reconstruction, proving its applicability in practical multi-scale 3D reconstructions.

Index Terms— Monocular Video-based 3D Reconstruction, Multi-Scale Scene Reconstruction, Multi-Resolution Spatial Structure

1. INTRODUCTION

Plentiful 3D Reconstruction methods have been proposed to build the 3D representation of real-world space based on the monocular posed video [1, 2, 3, 4, 5, 6]. However, these methods fall short in quality and efficiency when applied to multi-scale scenes.

The quality deficiency comes from the varying sizes of objects within multi-scale scenes. In 3D scene reconstruction, real-world spaces are abstracted into multiple voxels. For the same scene, the smaller the size of the voxels, the higher the quality of the reconstructed 3D model. Existing deep learning-based methods can only produce one fixed-size voxel when reconstructing the scene [2, 4, 5, 6]. However, single-size voxels struggle to balance the scope and quality of reconstruction. For example, a scale suitable for reconstructing larger objects may be too large to capture smaller details.

The efficiency deficiency comes from the fact that existing methods are GPU memory-intensive. They store 3D space data as a Truncated Signed Distance Function (TSDF) [7]. The GPU memory usage increases proportionately with the scene's size to reconstruct. Existing methods [8, 9] set different voxel sizes for different parts of space to reduce the total number of voxels, thus reducing the GPU memory. They can only alleviate the problem of a large GPU storage footprint. When faced with large space, e.g., the entire floor, the required memory resources can still surpass GPU limitations, leading to unsuccessful reconstructions.

To address these deficiencies, we propose MuSR — a novel multi-scale scene reconstruction method designed to adaptively adjust the voxel size of different scene parts. For the quality issue, we reconstruct the scene after dividing it dynamically by scale and then perform weighted fusion. For the efficiency issue, MuSR introduces a block-based sparse 3D data structure combined with a hardware resource management strategy to reduce GPU memory usage. We evaluate our MuSR using ScanNet [10], 7-Scenes [11] datasets, and real-world scenes. Our findings indicate that MuSR is efficient—requiring less than 1GB of GPU memory while maintaining swift reconstruction speeds—and adept at delivering superior reconstruction detail and overall scene completeness in multi-scale scenes. Moreover, our system's reconstruction efficiency surpassed that of leading algorithms in extensive scenes, demonstrating superior completeness and enhanced geometric shape reconstruction of objects. For more experiment results on reconstruction quality and efficiency, please visit our anonymous website [1].

The contributions of our work are threefold:

- We propose a novel multi-scale scene reconstruction method, MuSR, which skillfully addresses the quality and efficiency issues inherent in existing monocular video-based 3D reconstruction methods.
- We dynamically reconstruct the scene by scale in blocks and propose a sparse 3D data structure coupled with a corresponding GPU memory management policy for reconstruction quality and efficiency.
- We comprehensively evaluate our system using ScanNet, 7-Scenes datasets, and real-world scenes. The evaluation results demonstrate that MuSR outperforms SOTA efforts regarding efficiency, completeness, and geometric shape.

*Corresponding author: fengyuan.xu@nju.edu.cn

[1]`https://anoymousauthors.github.io/MuSR.github.io`

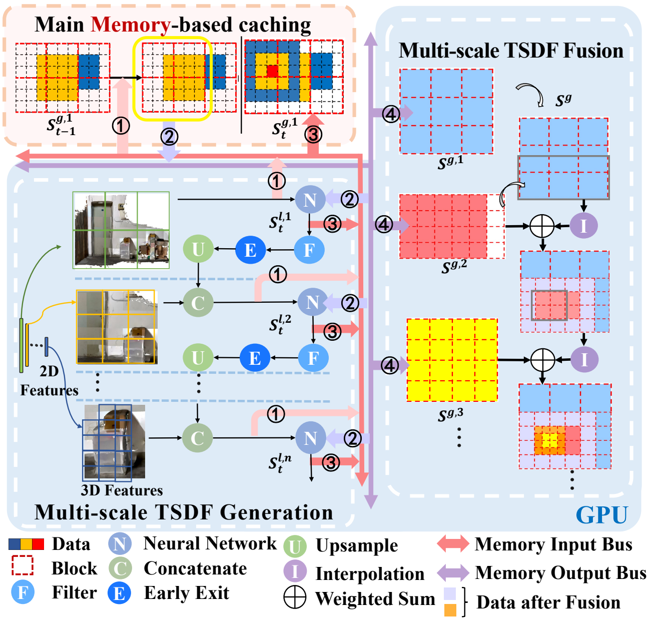


Fig. 1: MuSR architecture

2. MuSR DESIGN

2.1. System Overview

Fig. 1 illustrates MuSR's architecture. MuSR inputs a posed monocular video and outputs the scene mesh generated based on TSDF data. The current frame is recorded as a new key frame when the change of camera pose exceeds a certain threshold compared with the previous key frame, and every N key frame forms a fragment for the following 2D image feature extraction process. Our MuSR's design shares a similar idea with NeuralRecon[2], the SOTA 3D scene reconstruction methods, regarding applying coarse-to-fine network to obtain 3D spatial features and fusing new back-projected local features with global features. However, due to the limitation of NeuralRecon in reconstruction quality and efficiency, we make the following two key designs to enable 3D scene reconstruction in a multi-scale scene.

The first is the multi-scale TSDF generation and fusion design for reconstruction quality. For each resolution layer, we customize the voxel size and reconstruction scope of 3D feature space and predict local TSDF for the current fragment. $S_t^{l,n}$ represents a local TSDF volume at time t, whose resolution level is l. We also designed an early-exit scheme to decide which resolution is enough based on a valid number of voxels. Currently, the local TSDF should be updated to the global TSDF. $S^{g,n}$ represents a global TSDF volume with resolution n. To achieve multi-scale TSDF seamless integration, we design a block-based TSDF split strategy to generate mesh data in a divide-and-conquer manner. Specifically, For blocks located in the boundary of TSDF data with different resolutions, we design a weighted TSDF fusion algorithm for smoother transitions according to the situation of data change.

The second is a memory-assisted caching mechanism for reconstruction efficiency. The increasing global features and TSDF data cause the reconstruction crash problem because of insufficient GPU memory. Our core idea is to replace the cold data not needed in the current reconstruction step into the main memory for use by the GPU on demand when fusing current features with global features and updating current local TSDF data to global TSDF data. We design a two-level block structure and management policy to manage cold data.

2.2. Scale-aware Reconstruction

Multi-scale TSDF Generation According to the observation, low-resolution representations can describe larger spaces but lack local details. In contrast, high-resolution representations are better at demonstrating details in small spaces; we set larger spatial sizes and coarse-grain voxels for low-resolution layers and smaller spatial sizes and fine-grain voxels for high-resolution layers by setting the maximum depth of the viewing frustum (i.e., the distance from the far clipping plane to the camera) for each level. We set a large viewing frustum depth for low-resolution space to coarsely reconstruct distant objects and a small viewing frustum depth for the high-resolution space to reconstruct near objects finely.

If the number of voxels upsampled to the higher resolution space is small, the following reconstruction process can be skipped to save system overhead. We design a filter and early-exit scheme to help determine the resolution of the current fragment. Some voxels in the low-resolution space will be filtered because of the smaller reconstruction scope of the high-resolution space. We define the ratio of the number of voxels entering the next resolution layer to the number before filtering to check the valid range. The inference process will terminate early in the current resolution layer if it does not exceed the threshold.

Multi-scale TSDF Fusion. Since Marching Cubes (MC) algorithm [12] can not process multiple-resolution TSDF, we solve this problem in a divide-and-conquer manner. We divide TSDF data at different resolutions into blocks and choose the highest-resolution TSDF in the block. For non-boundary blocks, meshes can be directly generated through the MC algorithm. However, some voxels within the boundary block have a lower resolution TSDF than neighbor voxels. We design a weighted fusion algorithm for smoother high-resolution TSDF and low-resolution TSDF transitions. We calculate the local variance of TSDF changes in three axes directions within a $5 \times 5 \times 5$ local scope and use the square mean method to calculate the composite variance, which is used as a weight to indicate the preference between two resolutions. When the variance is less than t_1, low-resolution TSDF is directly used for smooth reconstruction results. When greater than t_2, high-resolution TSDF is chosen to restore geometric shapes. The weights between these two thresholds are linearly varied to ensure a smoother transition.

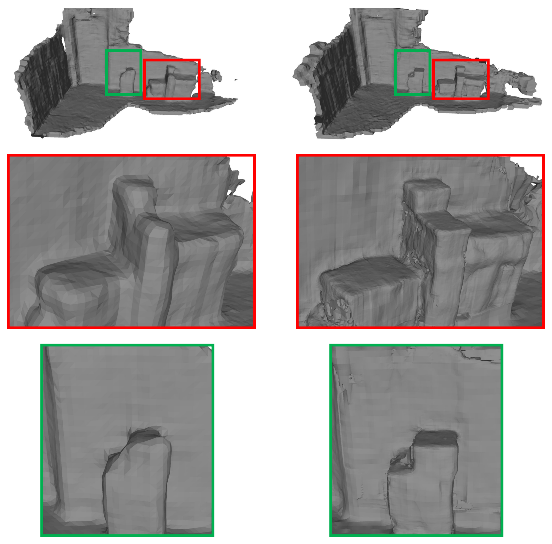


Fig. 2: Qualitative results for quality of local object details in real-world scenes.

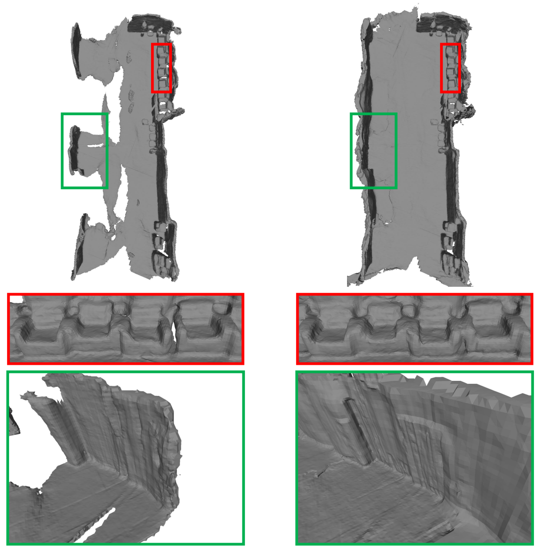


Fig. 3: Qualitative results for quality of global completeness in real-world scenes.

2.3. Main Memory-based Caching

We observe that the global intermediate features and TSDF data produced by neural networks (in Fig. 1) continuously increase, occupying a large amount of GPU memory. Failure to manage the continuously-increasing data will cause the reconstruction to crash. Our core idea is to split the global intermediate features and TSDF data into two parts, i.e., the cold data and the hot data, according to the requirement of the current reconstruction step. We can put the cold data into the main memory, which is usually sufficient on the server, for the subsequent GPU on-demand access.

Data Structure. The computational overhead of traversing the complete global data will seriously affect the system's running speed when acquiring and updating global data. We design a two-level block sparse structure to organize global intermediate features and TSDF data, which means maintaining block indices in GPU memory to find the involved blocks quickly and specific block values in the main memory.

Management Policy. As Fig. 1 shows, the reconstruction scope (denoted as ①) is determined by the local 3D features of current fragment. Then, we can obtain indices of related blocks and transfer the values of these blocks (denoted as ②) from main memory to GPU memory. Based on the current fragment's local and global features, new features (denoted as ③) can be computed by neural networks and transferred from GPU memory back to main memory to update global data. Some new blocks may be created during this process, so we need to record indices of related blocks to update block indices. All updated TSDF data (denoted as ④) will be transferred back to the main memory for the following TSDF fusion process.

3. EVALUATION

Implementation: For each layer, the voxel sizes are 16cm, 8cm, 4cm, 2cm, and 1cm; the maximum depths of the view frustum are 6m, 6m, 3m, 0.8m, and 0.8m. Except for the reconstruction scope in the 16 cm layer being 48^3, the rest are all 96^3. The early exit threshold is 0.25. The weighted fusion algorithm's block size is 0.4^3 cubic meters. It uses 8cm, 4cm, and 1cm for final fusion. t_1 and t_2 are 0.01 and 0.04. We perform network fine-tuning based on the pre-trained weights provided by NeuralRecon, and we freeze the weights of lower-resolution layers during the training at each layer.

Datasets: We conducted experiments on two indoor datasets, ScanNet and 7-Scenes, and supplemented with Replica dataset[13]. We used Pangolin to render RGB images and depth maps from the Replica dataset. We simulated the user's scanning method during rendering, generating camera trajectories. We added small random perturbations to the camera extrinsic each time to simulate real situations.

Metrics: The 3D geometry metrics are defined in [1] and 2D depth metrics are defined in [14].

3.1. Reconstruction Quality

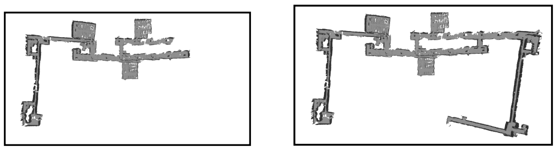
Local Details: Table 1 reports the reconstruction quality on ScanNet and 7-Scenes datasets. On ScanNet, MuSR outperforms NeuralRecon in all metrics. On 7-Scenes, MuSR lags slightly behind NeuralRecon regarding RMSE but surpasses it in all other metrics. Experimental results show that our MuSR outperforms NeuralRecon, the SOTA monocular reconstruction method, regarding reconstruction quality. We report the reconstruction results in real-world scenes in Fig. 2 to show the reconstruction quality more intuitively.

Table 1: Results for quality of local details.

Dataset	System	2D depth						3D geometric		
		Abs Rel↓	**Abs Diff↓**	**Sq Rel↓**	**RMSE↓**	**RMSE log↓**	$\delta < 1.25$↑	**Acc↓**	**Prec↑**	**F-score↑**
ScanNet	NeuralRecon	0.113	0.081	0.033	0.126	0.148	0.885	0.041	0.772	0.720
	MuSR	**0.084**	**0.060**	**0.023**	**0.102**	**0.120**	**0.928**	**0.035**	**0.816**	**0.780**
7-Scenes	NeuralRecon	0.216	0.171	0.085	**0.223**	0.228	0.719	0.177	0.372	0.450
	MuSR	**0.190**	**0.154**	**0.081**	0.227	**0.221**	**0.757**	**0.128**	**0.400**	**0.500**

Table 2: Results for quality of global completeness.

Dataset	System	2D depth	3D geometric	
		2D Comp↑	**3D Comp↓**	**Recall↑**
ScanNet	NeuralRecon	0.902	0.627	0.678
	MuSR	**0.967**	**0.380**	**0.750**
7-Scenes	NeuralRecon	0.822	0.354	0.555
	MuSR	**0.936**	**0.236**	**0.678**



(a) NeuralRecon (b) MuSR

Fig. 4: Results for reconstruction efficiency.

Global Completeness: Table 2 reports the results of reconstruction completeness on ScanNet and the 7-Scenes datasets. In both datasets, MuSR surpasses the NeuralRecon in all metrics, showing that our system has a better overall completeness. We show some reconstruction results in real-world scenes in Fig. 3. Our method outperforms the baseline and has high overall completeness.

3.2. System efficiency

Memory analysis. We count the GPU memory by the amount of memory requested by Pytorch's memory allocator from the GPU during the reconstruction process. An out-of-memory error occurs when Pytorch's memory request pushes the total GPU memory usage over its limit. Fig. 5 reports the GPU memory usage under a large real-world scene reconstruction process. The GPU memory usage of NeuralRecon continuously increases until an out-of-memory error occurs (marked as ×), failing to complete the reconstruction (Fig. 4a). However, MuSR's GPU memory usage is constant, less than 1GB, enabling efficient reconstruction (Fig. 4b).

Performance Analysis: Fig. 6 reports the time used throughout the entire reconstruction process. NeuralRecon takes less time to reconstruct each fragment in the initial phase (before the dashed line in Figure 6). After this line, MuSR performs faster. Because global data is organized using a sparse array in NeuralRecon, selecting local data and updating global data both require complete traversal, which becomes increasingly computationally expensive as the scale of the scene grows. Our design ensures that the amount of data processed in each fragment reconstruction is capped by a certain number of blocks, regardless of the scene size. Thus, MuSR takes less time to reconstruct each fragment in larger-scale scenes, showcasing efficiency in system performances.

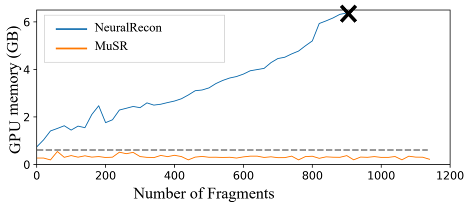


Fig. 5: GPU memory statistics.

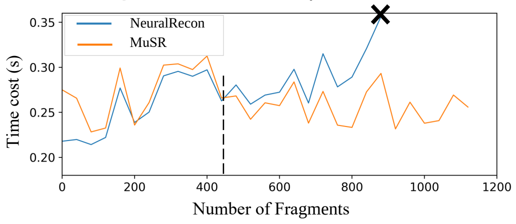


Fig. 6: Time statistics.

4. CONCLUSION

In conclusion, we proposed a multi-scale scene reconstruction method, MuSR, to address the challenges of large-scale 3D reconstruction from monocular videos. Our dynamic range multi-resolution spatial structure enables adaptive granularity adjustments for objects of varying sizes in real-world scenes. Additionally, our approach leverages a novel block-based sparse 3D data structure and hardware resource management strategy to reduce GPU memory usage. MuSR demonstrated superior reconstruction detail, overall scene completeness, and efficiency in memory usage and speed. Impressively, it surpassed state-of-the-art algorithms in terms of reconstruction quality and efficiency, offering marked potential for practical applications in large-scale 3D scene reconstructions.

5. ACKNOWLEDGEMENTS

This work was supported in part by NSFC under Grants 62341201, 62272224, 62302207, and 62272215, in part by the Leading Edge Technology Program of Jiangsu Natural Science Foundation under Grant BK20202001, and in part by the Science Foundation for Youths of Jiangsu Province under Grant BK20220772.

6. REFERENCES

[1] Zak Murez, Tarrence Van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich, "Atlas: End-to-end 3d scene reconstruction from posed images," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*. Springer, 2020, pp. 414–431.

[2] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao, "Neuralrecon: Real-time coherent 3d reconstruction from monocular video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15598–15607.

[3] Noah Stier, Alexander Rich, Pradeep Sen, and Tobias Höllerer, "Vortx: Volumetric 3d reconstruction with transformers for voxelwise view selection and fusion," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 320–330.

[4] Aljaz Bozic, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner, "Transformerfusion: Monocular rgb scene reconstruction using transformers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1403–1414, 2021.

[5] Chenyangguang Zhang, Zhiqiang Lou, Yan Di, Federico Tombari, and Xiangyang Ji, "SST: Real-time end-to-end monocular 3d reconstruction via sparse spatial-temporal guidance," *arXiv preprint arXiv:2212.06524*, 2022.

[6] Zi-Xin Zou, Shi-Sheng Huang, Yan-Pei Cao, Tai-Jiang Mu, Ying Shan, and Hongbo Fu, "Mononeuralfusion: Online monocular neural 3d reconstruction with geometric priors," *arXiv preprint arXiv:2209.15153*, 2022.

[7] Brian Curless and Marc Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312.

[8] Emanuele Vespa, Nils Funk, Paul HJ Kelly, and Stefan Leutenegger, "Adaptive-resolution octree-based volumetric slam," in *2019 International Conference on 3D Vision (3DV)*. IEEE, 2019, pp. 654–662.

[9] Lukas Schmid, Jeffrey Delmerico, Johannes L Schönberger, Juan Nieto, Marc Pollefeys, Roland Siegwart, and Cesar Cadena, "Panoptic multi-tsdfs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8018–8024.

[10] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5828–5839.

[11] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon, "Scene coordinate regression forests for camera relocalization in rgb-d images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2930–2937.

[12] William E Lorensen and Harvey E Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[13] Julian Straub et al., "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[14] David Eigen, Christian Puhrsch, and Rob Fergus, "Depth map prediction from a single image using a multi-scale deep network," *Advances in neural information processing systems*, vol. 27, 2014.