2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

3QFP: Efficient neural implicit surface reconstruction using Tri-Quadtrees and Fourier feature Positional encoding

Shuo Sun[1], Malcolm Mielle[2], Achim J. Lilienthal[1,3], and Martin Magnusson[1]

***Abstract*— Neural implicit surface representations are currently receiving a lot of interest as a means to achieve high-fidelity surface reconstruction at a low memory cost, compared to traditional explicit representations. However, state-of-the-art methods still struggle with excessive memory usage and non-smooth surfaces. This is particularly problematic in large-scale applications with sparse inputs, as is common in robotics use cases. To address these issues, we first introduce a sparse structure, *tri-quadtrees*, which represents the environment using learnable features stored in three planar quadtree projections. Secondly, we concatenate the learnable features with a Fourier feature positional encoding. The combined features are then decoded into signed distance values through a small multi-layer perceptron. We demonstrate that this approach facilitates smoother reconstruction with a higher completion ratio with fewer holes. Compared to two recent baselines, one implicit and one explicit, our approach requires only 10%–50% as much memory, while achieving competitive quality. The code is released on `https://github.com/ljjTYJR/3QFP`.**

I. INTRODUCTION

Most autonomous systems rely on an accurate model of the environment, i.e. a map, for localization and planning. Various methods have been designed to represent maps as point clouds [1, 2], spatial voxels with normal distributions [3, 4, 5] or SDF values [6, 7, 8, 9], occupancy grids [10, 11], surfels [12], meshes [13], etc. However, these methods often require substantial memory resources to maintain an accurate and detailed environment representation, especially in large-scale scenes. Limiting the available memory, on the other hand, will decrease the map quality. In addition, these methods encounter difficulties in accurately reconstructing the environment in detail when the input data is sparse, resulting in holes and non-smooth surfaces in the map (see Fig. 4 in [12] and Fig. 1 in [13]).

Recent neural implicit representation techniques have achieved notable success in shape representation [14, 15, 16] and scene representation [17]. These methods enable the implicit storage of environmental information within a neural network and/or within learnable feature volumes, thereby enabling compact yet detailed reconstruction of such environments. However, previous neural implicit reconstruction techniques primarily focus on objects or small scene reconstruction [18], and the memory footprint does not scale to large-scale environments. Recent research has explored the use of sparse structures such as octrees [19, 20] for better scalability. Furthermore, when inputs are sparse, these methods can often lead to the generation of non-smooth surfaces [21], a phenomenon that remains a significant challenge in the field.

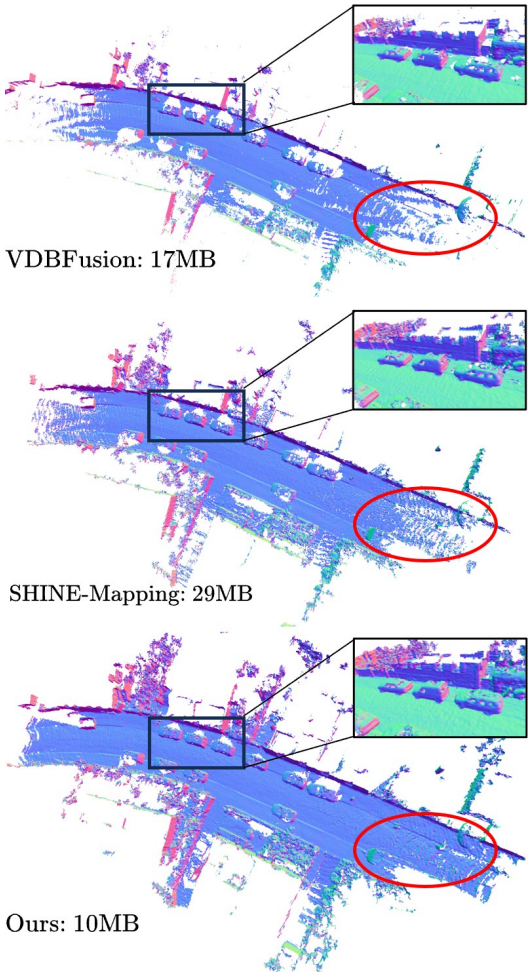


Fig. 1: Qualitative reconstruction result on `KITTI-Seq07`. Our method can achieve better reconstruction quality using less memory compared to *SHINE-Mapping* [20] and *VDB-Fusion* [22]. When given noisy and sparse lidar scans, our method can achieve a more complete reconstruction (see red circle and **black zoomed-in** square areas).

Our first contribution is a novel feature representation structure called *tri-quadtrees*. Instead of storing features in 3D voxel grids [20, 23, 24] or dense feature planes [21], we use three planar quadtrees to represent surfaces. Our method combines the sparsity of the octree and the efficiency of the

*This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017274 (DARKO).

[1]AASS MRO lab, Örebro University, Sweden. {`shuo.sun, achim.lilienthal, martin.magnusson`}`@oru.se`
[2]Independent researcher. `malcolm.mielle@protonmail.com`
[3]Technical University of Munich, Chair: Perception for Intelligent Systems. `achim.j.Lilienthal@tum.de`

feature planes, greatly reducing memory requirements while still producing comparable results, as shown in fig. 1.

Our second contribution is a hybrid feature representation method. Although learnable features can achieve detailed and high-fidelity reconstruction, they tend to degenerate when inputs are sparse [25, 26]. We combine the learnable features with the Fourier feature positional encoding [27], which can help fill holes and smoothen reconstruction when given sparse inputs.

The concatenated features are fed into a small multi-layer perceptron (MLP) for predicting a continuous signed distance field of the scene. Features and MLP parameters are optimized end-to-end under the supervision of the direct range measurement. As our experiments demonstrate, compared to the recent explicit SDF/TSDF representation VDBFusion [22], our method achieves more complete reconstructions; compared to the state-of-the-art neural implicit reconstruction [20], our method requires significantly less memory.

II. Related Work

A. Explicit Representation

The most common representations used in current robot mapping implementations are explicit representations, such as point clouds [1, 2], NDTs [5, 4], occupancy grids [11, 10], surfels [12], meshes [13, 28], TSDF values [6, 22], etc. *VDBFusion* [22] stores TSDF values in sparse voxels explicitly; each voxel will be queried during reconstruction. Unlike previous approaches that explicitly divide TSDF or SDF maps into voxels, our method adopts a neural network to store SDF values and allows queries of arbitrary coordinates in a continuous manner. The use of neural networks allows our method to achieve high reconstruction completion ratios while maintaining the smoothness of the reconstructed surface.

B. Neural Implicit Representation

1) Pretrained Encoder-Decoder: DeepSDF [14] and *OccNet* [15] propose to use a neural network to represent the SDF and occupancy probability for the shape. *CovONet* [17] maps 3D points to feature grids based on the PointNet encoder [29], and then uses an MLP as a decoder for predicting occupancy probability. The encoder and decoder are trained in an end-to-end manner. When querying a coordinate $\boldsymbol{p}$, the feature value at $\boldsymbol{p}$ is interpolated by the spatially adjacent grids and fed to the decoder. The trained encoders and decoders can be used directly in other environments for inference, which, however, often suffer from the generalization problem [21].

2) Test-time optimization: The seminal work of *NeRF* [30] on novel view synthesis has inspired a surge of research activity in geometry reconstruction. For example, *iMAP* [31] and *iSDF* [32] use an overfit large MLP to store environment information, with the environment representation stored within the network parameters. The training of such large MLPs is, however, time-consuming, prompting the development of various techniques aimed at addressing this issue. One efficient manner is to move the computational complexity from the neural network to the scene feature volume, introducing dense feature voxel grids [33, 34]. With learnable features, only a small MLP can be used as a decoder, reducing computational load and accelerating training. Accounting for the large memory footprint when applying dense feature voxel grids, several techniques have been proposed to reduce memory usage, such as hash-tables [35], octree-trees [16]; these compact data structures have been leveraged in recent robotic applications [20, 18, 19, 26, 24, 23, 36]. However, these voxel-based feature representations can still consume $\mathcal{O}(n^3)$ memory in the worst case. To further reduce memory usage, recent works [37, 21, 38, 39, 40, 41] has proposed using three orthogonal axis-aligned feature planes instead of 3d voxel grids.

Motivated by the feature plane representation, we propose a more compact representation called tri-quadtrees, which can further reduce memory usage.

The work similar to ours is *SHINE-Mapping* [20]. However, we propose a novel feature representation method that significantly reduces memory usage in large-scale scenes. In addition, inspired by *CO-SLAM* [26], we combine Fourier feature positional encoding with learnable features, resulting in smoother and more complete reconstructions when input data is sparse.

III. Method

Our method learns a continuous signed distance function representation of the environment given lidar scans and known poses. Specifically, the world coordinate $\boldsymbol{p}_i \in \mathbb{R}^3$ is mapped into a SDF value $s_i \in \mathbb{R}$. As shown in fig. 2, our neural implicit representation is composed of two components: the learnable features stored in the quadtree nodes and a globally shared MLP for predicting the SDF value. The features and the network parameters are learned during the test time by using direct lidar measurement to supervise network predictions.

Section III-A delves into how we use the novel *tri-quadtrees* structure to compactly represent a given scene. Section III-B elaborates on the application of Fourier features positional encoding. Finally, section III-C introduces the loss function and some training details. Based on the proposed neural implicit representation, we extract the mesh by Marching Cubes [42] for visualization and evaluation.

A. Tri-Quadtrees feature representation

Tri-Quadtrees: While storing learnable features in spatial voxel grids [34] can lead to fast convergence rates, dense 3D grids suffer from a cubical growth rate of memory usage as the environment size increases. To avoid storing unnecessary features in free space, prior work [20, 23, 16, 24] employs octree to store features only within voxel grids where surface points are located. Other research such as Johari, Carta, and Fleuret [21] adopts feature planes, which represent scene features by three orthogonal planes. In this paper, we propose a novel data structure to represent spatial

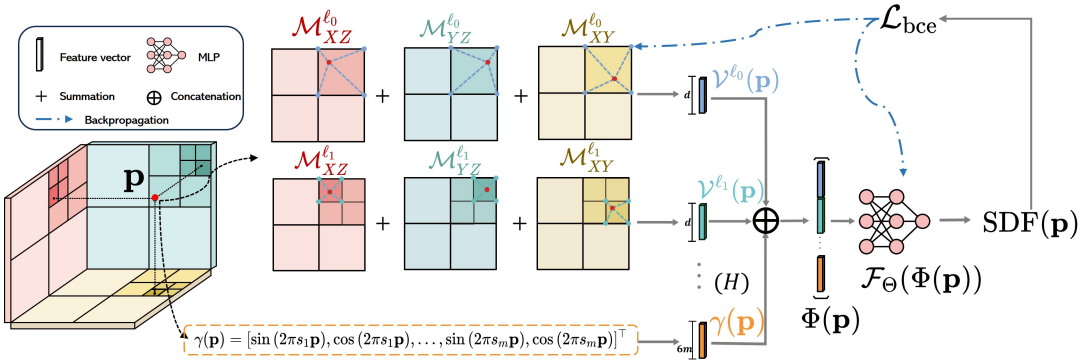


Fig. 2: **Overview of our method.** We represent the scene with three planar quadtrees $\mathcal{M}_i^\ell$, $i \in \{XZ, YZ, XY\}$ and ℓ represents the quadtree depth. We store features in the deepest H levels of resolution of quadtrees. When querying for a point $\boldsymbol{p}$, we project it onto planar quadtrees to identify the node containing $\boldsymbol{p}$ at the level ℓ. The feature of $\boldsymbol{p}$ is then calculated by bilinear interpolation based on the queried location and vertex features. We add features at the same level and concatenate among different levels. Concatenated with the positional encoding $\gamma(\boldsymbol{p})$, $\boldsymbol{p}$'s feature ($\Phi(\boldsymbol{p})$) is fed into a small MLP ($\mathcal{F}_\Theta$) to predict the SDF value. The learnable features stored in the quadtree nodes and the network parameters are learned by test-time optimization using the loss function $\mathcal{L}_{\text{bce}}$. The learnable feature vectors have length d and the positional encoding feature vector has length $6m$.

features—tri-quadtrees, which is more compact than previous methods.

Specifically, given lidar scans and known poses, we first project the 3D point clouds onto three axis-aligned orthogonal 2D planes and then construct quadtrees on each plane with a maximum depth of L_{max} and resolution of the leaf nodes equal to r. Unlike the method in *NGLoD* [16], which uses all levels of resolution in the octree to store features, we only use the deepest H levels of quadtrees for feature representation to balance the quality of the reconstruction and memory footprint. That is, only the levels $\ell \in \{L_{\text{max}} - H + 1, \cdots, L_{\text{max}}\}$ in the quadtree are used to interpolate features. By default, we set $H = 3$, which was empirically sufficient to achieve good results.

Feature Computation: We denote the "feature plane quadtree" as $\mathcal{M}_i^\ell$ where $i \in \{XY, XZ, YZ\}$ is the plane index and $\ell \in \{L_{\text{max}} - H + 1, \cdots, L_{\text{max}}\}$ is the depth level. Each node in the depth ℓ holds a one-dimensional learnable feature vector $\mathcal{V}_i^{\ell,(j)}$ at each of its four vertices (indexed by j), where $\mathcal{V}_i^{\ell,(j)} \in \mathbb{R}^d$ and $d \in \mathbb{N}$ is the length of the feature vector. The features are initialized randomly when created and optimized until convergence during training.

For quick queries, we store vertex features with hash tables. Specifically, for each level $\ell \in \{L_{\text{max}} - H + 1, \cdots, L_{\text{max}}\}$ of each quadtree, we maintain a hash table (totally $3H$) to store vertex features. The vertex Morton code is used as hash table keys, which can map two-dimensional vertex indices to one-dimensional scalars.

To get the feature for a query point $\boldsymbol{p}$ at the level ℓ, we first project $\boldsymbol{p}$ to planes and get the 2D points $\boldsymbol{p}_x$, $\boldsymbol{p}_y$ and $\boldsymbol{p}_z$. Then we traverse the quadtree at level ℓ to find the node containing the corresponding 2D point. The feature for the 2D point is bilinearly interpolated by the four vertex features in the node. The feature $\mathcal{V}^\ell(\boldsymbol{p})$ of level ℓ for the point $\boldsymbol{p}$ is calculated through summation for all 3 planes: $\mathcal{V}^\ell(\boldsymbol{p}) = \mathcal{V}_{XZ}^\ell(\boldsymbol{p}_y) + \mathcal{V}_{YZ}^\ell(\boldsymbol{p}_x) + \mathcal{V}_{XY}^\ell(\boldsymbol{p}_z)$, $\mathcal{V}^\ell(\boldsymbol{p}) \in \mathbb{R}^d$.

The final feature $\mathcal{V}(\boldsymbol{p})$ will be collected and concatenated across all H levels:

$$\mathcal{V}(\boldsymbol{p}) = [\mathcal{V}^{L_{\text{max}}-H+1}(\boldsymbol{p}), \mathcal{V}^{L_{\text{max}}-H+2}(\boldsymbol{p}), \cdots, \mathcal{V}^{L_{\text{max}}}(\boldsymbol{p})]^\top,$$

where $[\cdot, \cdot]$ denotes concatenation, $\mathcal{V}(\boldsymbol{p}) \in \mathbb{R}^{dH}$.

B. Fourier features positional encoding

Although the learnable features can contribute to a more accurate reconstruction, they do not provide the necessary hole filling and smoothness, as demonstrated by Wang, Wang, and Agapito [26]. To improve the completion ratio of the reconstruction, we combine the learnable features and the positional encoding. We demonstrate through experiments (section IV-D) that combining learnable features with the positional encoding can boost the completion ratio with only a marginal increase in computational cost.

In contrast to the setup in Wang, Wang, and Agapito [26], we do not employ the *one-blob* encoding. Instead, we found that Fourier feature positional encoding can achieve a smoother result in our cases. Inspired by Tancik et al. [27], we adopt Gaussian positional encoding which has been shown to achieve a higher completion ratio than *one-blob* and frequency encoding used in *NeRF* [30]. Specifically, the positional encoding for the query point $\boldsymbol{p}$ would be

$$\gamma(\boldsymbol{p}) = [\sin(2\pi s_1\boldsymbol{p}), \cos(2\pi s_1\boldsymbol{p}), \ldots, \sin(2\pi s_m\boldsymbol{p}), \cos(2\pi s_m\boldsymbol{p})]^\top$$

where s_i are coefficients ($i \in \{1, 2, \ldots, m\}$) sampled from an isotropic Gaussian distribution, i.e., $s_i \sim \mathcal{N}(0, \sigma^2)$, where σ^2 is a tuned hyperparameter; m is a hyperparameter controlling the length of the positional encoding feature. We determine σ^2 by doing a hyperparameter search based on the training loss.

Finally, for the query point $\boldsymbol{p}$, we concatenate the tri-quadtrees feature $\mathcal{V}(\boldsymbol{p}) \in \mathbb{R}^{dH}$ and Fourier positional encoding $\gamma(\boldsymbol{p}) \in \mathbb{R}^{6m}$ into the final feature vector $\Phi(\boldsymbol{p}) = [\mathcal{V}(\boldsymbol{p}), \gamma(\boldsymbol{p})]^\top \in \mathbb{R}^{dH+6m}$.

To query the SDF value, we feed the feature $\Phi(\boldsymbol{p})$ into a small MLP $\mathcal{F}_\Theta$ to obtain the SDF value, where Θ represents the network parameters. The whole process is differentiable, we jointly optimize the quadtree features and the MLP parameter Θ end-to-end during training.

C. Training and Optimization

1) Sampling: We take samples from both free-space and points close to the surface. For each lidar ray, we randomly select N_s points $\{\boldsymbol{p}_i\}_{i=1}^{N_s}$ in a truncated area of the ray's endpoint $\boldsymbol{p}_e$; we also take N_f points $\{\boldsymbol{p}_j\}_{j=1}^{N_f}$ in free space along the ray. For SDF supervision signals, we directly calculate the distance between the sampled points $\boldsymbol{p}_s$ and the endpoint $\boldsymbol{p}_e$[1]. The SDFs of the points located between the sensor and the endpoint will have a negative value, while those outside will be positive.

2) Loss Function: Our training only needs a simple SDF loss function[2]. Following the configuration in [20, 25], we adopt the Binary Cross Entropy (BCE) loss for faster convergence. Specifically, we map both the ground truth SDF_{gt} and the predicted SDF_{pred} to the range $[0, 1]$ using the sigmoid function ($\texttt{Sig}$). For one training pair, the BCE loss will be:

$$\mathcal{L}_{\text{bce}} = o_{\text{gt}} \cdot \log\left(o_{\text{pred}}\right) + \left(1 - o_{\text{gt}}\right) \cdot \log\left(1 - o_{\text{pred}}\right), \quad (1)$$

where $o_{\text{gt}} = \texttt{Sig}(\text{SDF}_{\text{gt}})$ and $o_{\text{pred}} = \texttt{Sig}(\text{SDF}_{\text{pred}})$ are the ground truth and predictions respecitvely.

IV. Experiments

In this section, we first demonstrate through quantitative experiments that tri-quadtrees is more memory efficient than the current state-of-the-art implicit representation methods (see section IV-C) while still achieving a higher completion ratio than explicit representation methods, as shown in section IV-B. Qualitative results demonstrate that our method is capable of producing smoother reconstructions when inputs are sparse. Finally, an ablation study reveals that the use of positional encoding can significantly enhance smoothness and hole-filling (section IV-D).

A. Experiment Setup

Baseline: We compare our approach to the state-of-the-art explicit reconstruction method *VDBFusion* [22] and the neural implicit reconstruction method *SHINE-Mapping* [20], both of which have publicly available implementations and are based on TSDF or SDF representations.

Evaluation Metric: Following the setup of the experiment in [13, 20], we evaluate the geometry of the reconstruction by indirectly examining the mesh constructed from the TSDF output (created by *marching cubes* [42]). We uniformly sample 10^7 points using `open3d` library from the generated meshes and compare with the ground truth point cloud for evaluation. We report accuracy [cm], completion [cm], completion ratio [%], and accuracy ratio [%]. Briefly, let $\mathcal{P}$ be the point cloud sampled from the prediction mesh, while $\mathcal{G}$ is the ground truth point cloud. For a point $\boldsymbol{p}_i \in \mathcal{P}$, we define the distance to $\mathcal{G}$ as

$$d(\boldsymbol{p}_i, \mathcal{G}) = \min_{\boldsymbol{g}_i \in \mathcal{G}} ||\boldsymbol{p}_i - \boldsymbol{g}_i||.$$

Similarly, the distance between the point $\boldsymbol{g}_i$ and $\mathcal{P}$ is $d(\boldsymbol{g}_i, \mathcal{P}) = \min_{\boldsymbol{p}_i \in \mathcal{P}} ||\boldsymbol{g}_i - \boldsymbol{p}_i||$. We compute $d(\boldsymbol{p}_i, \mathcal{G})$ as accuracy and $d(\boldsymbol{g}_i, \mathcal{P})$ as completion, the ratio of which less than set threshold works as accuracy ratio and completion ratio. In addition, we measure the memory usage of each method.

TABLE I: Parameter Setting: the table shows the important hyperparameters appearing in section III.

Category	Parameter	Value	Category	Parameter	Value
Quadtree	H	3	Feature	d	8
	L_{max}	12		m	16
MLP	depth	2	Sampling	N_s	3
	hidden_width	32		N_f	3

Datasets: We evaluate our approach on two public lidar datasets. One is `MaiCity`[3]—a synthetic urban-like outdoor scenario with 100 lidar frames. The other one is `NewerCollege` [44], a real lidar dataset including 1500 frames captured in a college campus environment. The two datasets provide registered dense point clouds as ground truth reference for quantitative evaluation. Figure 1 includes a qualitative comparison on the KITTI dataset, but since KITTI does not provide accurate ground truth we do not compare quantitative accuracy numbers on that dataset.

Implementation Details: In all experiments, our method uses the parameters shown in table I (we use these parameters to trade off efficiency and accuracy). We set $\sigma^2 = 50$ by parameter searching on `MaiCity` and apply it on both datasets.

We employ the default or recommended configurations for the baseline methods. Specifically, we use 0.1 m leaf node resolution in the *SHINE-Mapping* octree and our tri-quadtrees. Correspondingly, we use 0.1 m voxel size for *VDBFusion*. In the experiment, we use marching cubes with 0.1 m voxel resolution to extract meshes for all methods.

B. Map Quality

In our first map quality evaluation experiment, we use a *dense* input configuration: in `MaiCity` all frames are used, while `NewerCollege` selects one frame out of every three, to show that our method can achieve competitive performance with a smaller memory footprint.

The evaluation results are shown in table II. While *VDBFusion* achieves slightly better accuracy than the implicit methods, the completion metric is much lower than the neural implicit reconstruction methods. Our method and *SHINE-Mapping* have a significantly higher completion ratio with only a slight drop in accuracy.

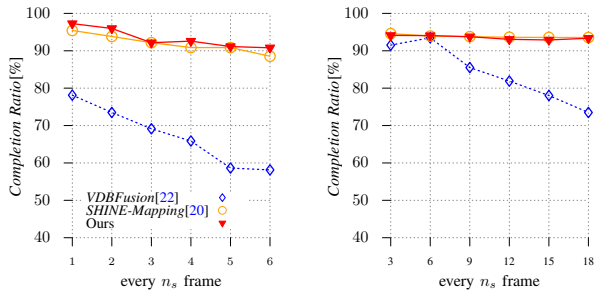
[1] Other supervision signals such as the calculated gradients [43] and the nearest distance [32] can also be used. Since lidar often provides accurate range measurement compared to RGB-D cameras, we adopt such *projected distance* directly [6, 20]

[2] We also attempted to include additional regularization losses such as the Eikonal loss [33], however, it did not make a significant difference; therefore, we decided to omit it for the sake of simplicity.

[3] https://www.ipb.uni-bonn.de/data/mai-city-dataset/

TABLE II: Quantitative evaluation of the reconstruction quality on the MaiCity and NewerCollege with *dense* inputs. We report the *Completion* (Comp.), *Accuracy* (Acc.), *Completion Ratio* (Comp.Ratio) and *Accuracy Ratio* (Acc.Ratio) with a threshold of 0.1 m for MaiCity and 0.2 m for NewerCollege. We also report the number of learnable parameters for neural implicit representation methods. Bold fonts represent the best results. Our method achieves a significantly higher completion ratio than *VDBFusion* with fewer parameters than *SHINE-Mapping*. (↓: lower better; ↑: higher better.)

Dataset	Method	#Param ↓	Comp.[cm] ↓	Acc.[cm] ↓	Comp.Ratio[%]↑	Acc.Ratio[%]↑
MaiCity	*VDBFusion* [22]	\	27.33	**1.36**	78.12	**99.13**
	SHINE-Mapping [20]	4.53×10^6	3.34	1.66	95.43	97.09
	Ours	$\mathbf{1.27 \times 10^6}$	**2.68**	1.52	**97.27**	97.60
NewerCollege	*VDBFusion*	\	13.20	**5.50**	91.51	**98.10**
	SHINE-Mapping	1.14×10^7	**9.55**	7.60	**94.58**	91.37
	Ours	$\mathbf{1.60 \times 10^6}$	9.68	6.72	94.10	93.69



(a) MaiCity (b) NewerCollege

Fig. 3: Comparison of *Completion Ratio*[%] versus the input frame numbers n_s on two datasets. The threshold is 0.1 m for MaiCity and 0.2 m for NewerCollege. As the inputs get sparser, the completion ratio of *VDBFusion* drops significantly, while our method maintains a high completion ratio. Though with similar performance, our method uses fewer parameters than *SHINE-Mapping* (see fig. 5).

Taking into account that many mapping pipelines use sparse keyframes and not all input frames [45, 18], we further evaluate the methods when given sparse input. We select one frame for every n_s frames for reconstruction. Compared to accuracy, completion can better reveal reconstruction ability. Focusing on completion-related metrics, one can see in fig. 3, that the neural implicit representation methods maintain a high completion ratio despite sparser inputs. A visualization of the reconstructed result is shown in fig. 4, where we highlight the ground truth points with errors larger than 10 cm in orange. We can clearly see that there are many "holes" in the reconstructed mesh of *VDBFusion*, resulting in a low completion ratio.

Although *SHINE-Mapping* has a similar completion ratio to our approach, we show that thanks to our compact and efficient data structure, our method needs substantially fewer parameters and achieves a similar performance. As demonstrated in fig. 5, our method needs about 25% and 10% as many parameters as *SHINE-Mapping* on MaiCity and NewerCollege respectively. We will discuss more about the memory footprint in the following section (IV-C). Also, in the second row of fig. 4, we show that when inputs are sparse, our method can achieve a smoother reconstruction.

TABLE III: Ablation study on the different encoding methods. The evaluation is conducted on MaiCity. The full model combining learnable features and positional encoding achieves the best performance overall. Using positional encoding alone can result in an over-smooth result and generate stripe artifacts (see Figure 7).

	w/o Pos.enc.	w/o Fea	Full
Comp.[cm] ↓	8.05	9.27	**7.88**
Comp.Ratio[%]↑	90.95	89.44	**91.10**
Acc.[cm] ↓	3.07	2.95	**2.60**
Acc.Ratio[%]↑	94.71	**96.30**	96.25

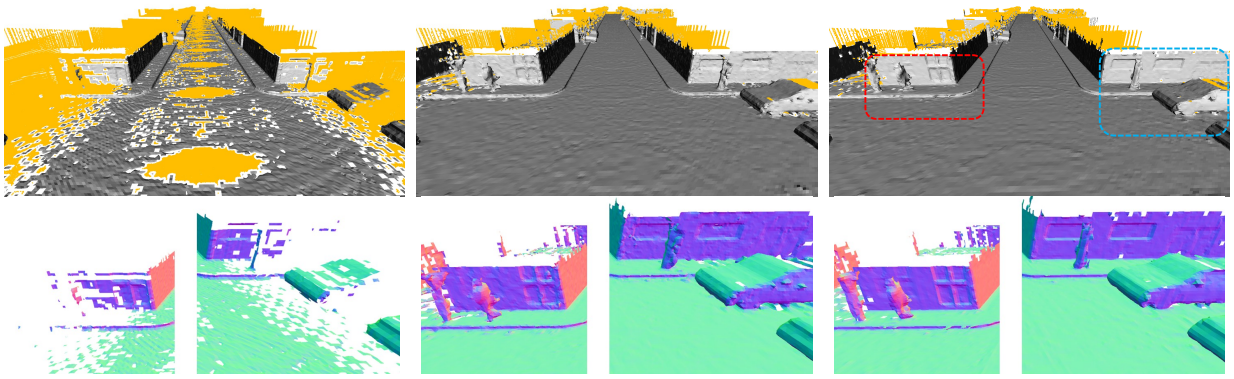
C. Memory Footprint

In this section, we analyze the memory usage in the experiment. For *VDBFusion*, we save its VDB structure as the map representation, which stores TSDF values in grids. For neural implicit representation methods, we can save the parameters in the model and thus determine the footprint by the size of the saved model, which consists of learned parameters and the MLP decoder parameters. Fixing the voxel size, fig. 6 shows the memory consumption of the different methods for the NewerCollege dataset. One can see that tri-quadtrees is the most efficient method with respect to memory and the number of input scans: the point cloud map stores dense 3D points and uses the most memory, and tri-quadtrees requires only 20% of *SHINE-Mapping* memory usage due to *SHINE-Mapping* storing learned features in 3D voxels. Tri-quadtrees can use less memory but achieve better reconstruction than *VDBFusion*. Overall, tri-quadtrees achieves comparable map quality to the state-of-art while being more memory efficient.

D. Ablation Study: Positional Encoding

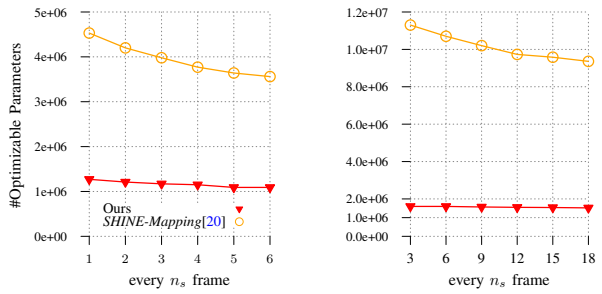
We have performed an ablation study to demonstrate the relative performance of our proposed combined feature encoding vs. the tri-quadtrees features and Fourier feature positional encoding separately.

Table III illustrates the quantitative evaluation with different encodings on the MaiCity dataset. Combining both the Fourier feature positional encoding and tri-quadtrees features (namely, "Full" in table III) achieves the best performance overall. Though using positional encoding alone can achieve a slightly higher accuracy ratio quantitatively, fig. 7 depicts



(a) *VDBFusion* (b) *SHINE-Mapping* (c) Ours

Fig. 4: Qualitative visualization of the map quality on the `MaiCity` dataset using every 6th frame. The first row depicts the difference between the dense ground truth point cloud and the reconstructed mesh; the ground truth points with an error of more than 0.1 m are highlighted in orange. The second row shows zoomed-in images of the dashed areas (indicated in the top-right image). When inputs are sparse (e.g., every 6th frame in this case), our method obtains visibly smoother results.



(a) `MaiCity` (b) `NewerCollege`

Fig. 5: Number of learnable parameters versus subsampling frequency given as n_s, the number of frames after which another frame was selected from the two datasets. Our method only needs about 25% and 10% parameters of *SHINE-Mapping* on `MaiCity` and the `NewerCollege` dataset, respectively.

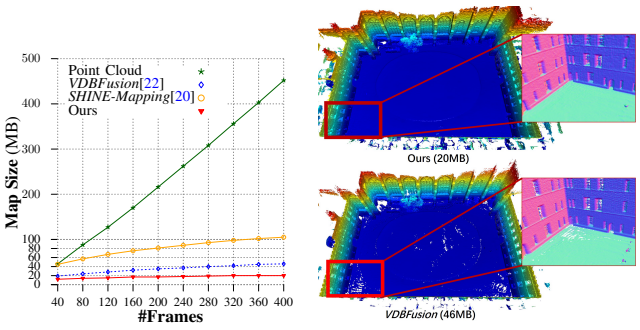
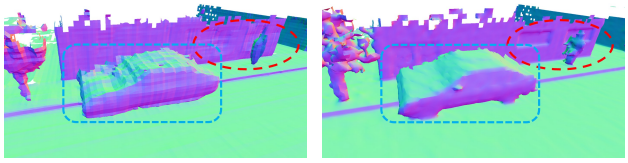


Fig. 6: Memory consumption for different reconstruction methods for the `NewerCollege` dataset. Storing pure point clouds consumes the largest memory usage. Our method is much more compact than *SHINE-Mapping*, requiring only 20% of the map size while still achieving comparable results. Compared to *VDBFusion*, our method requires less memory but produces a more accurate map; as can be seen in the right column, there are some holes in the map generated by *VDBFusion*.



(a) Positional encoding only (b) Full model

Fig. 7: A comparison of the map quality between using positional-encoding only and the Full model. Using positional encoding alone will generate an over-smooth mesh (see the wall in the red circle) and stripe artifacts (see the car in the blue square).

that the generated mesh lacks details and is with stripe artifacts.

V. Discussion & Conclusion

In this paper, we introduce a novel feature representation method, tri-quadtrees, for neural implicit representation. Our method combines the advantages of octree sparsity and the compactness of feature plane representation, which requires substantially less memory but achieves better results on completion and competitive results on accuracy. Thanks to the Fourier feature positional encoding, our method can still achieve a smooth result when the inputs are sparse.

The work in this paper focuses on mapping, which requires given poses. For future research, we will combine the proposed mapping method with tracking for a neural simultaneous localization and mapping (SLAM) system. In addition, though a small MLP can be optimized quickly, it lacks enough capacity to fully represent a large-scale environment. In the future, we will investigate the mapping and fusion with multiple MLPs based on submaps.

References

[1] Pierre Dellenbach, Jean-Emmanuel Deschaud, Bastien Jacquet, and François Goulette. “CT-ICP: Real-time elastic LiDAR odometry with loop closure”. In: *2022*

International Conference on Robotics and Automation (ICRA). IEEE. 2022, pp. 5580–5586.

[2] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. "KISS-ICP: In defense of point-to-point ICP–simple, accurate, and robust registration if done the right way". In: *IEEE Robotics and Automation Letters* 8.2 (2023), pp. 1029–1036.

[3] Martin Magnusson, Achim J. Lilienthal, and Tom Duckett. "Scan Registration for Autonomous Mining Vehicles Using 3D-NDT". In: 24.10 (Oct. 2007), pp. 803–827. DOI: 10.1002/rob.20204.

[4] Martin Magnusson, Henrik Andreasson, Andreas Nuchter, and Achim J Lilienthal. "Appearance-based loop detection from 3D laser data using the normal distributions transform". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 23–28.

[5] Todor Stoyanov, Martin Magnusson, Henrik Andreasson, and Achim J Lilienthal. "Path planning in 3D environments using the normal distributions transform". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 3263–3268.

[6] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 1366–1373.

[7] Yue Pan, Yves Kompis, Luca Bartolomei, Ruben Mascaro, Cyrill Stachniss, and Margarita Chli. "Voxfield: Non-projective signed distance fields for online planning and 3d reconstruction". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 5331–5338.

[8] Daniel Ricao Canelhas, Todor Stoyanov, and Achim J. Lilienthal. "SDF Tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2013), pp. 3671–3676. URL: https://api.semanticscholar.org/CorpusID:13787371.

[9] Daniel R. Canelhas, Todor Stoyanov, and Achim J. Lilienthal. "From Feature Detection in Truncated Signed Distance Fields to Sparse Stable Scene Graphs". In: *IEEE Robotics and Automation Letters* 1.2 (2016), pp. 1148–1155. DOI: 10.1109/LRA.2016.2523555.

[10] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. "OctoMap: An efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous robots* 34 (2013), pp. 189–206.

[11] Daniel Duberg and Patric Jensfelt. "UFOMap: An efficient probabilistic 3D mapping framework that embraces the unknown". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6411–6418.

[12] Jens Behley and Cyrill Stachniss. "Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments." In: *Robotics: Science and Systems*. Vol. 2018. 2018, p. 59.

[13] Ignacio Vizzo, Xieyuanli Chen, Nived Chebrolu, Jens Behley, and Cyrill Stachniss. "Poisson surface reconstruction for LiDAR odometry and mapping". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 5624–5630.

[14] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. "Deepsdf: Learning continuous signed distance functions for shape representation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 165–174.

[15] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. "Occupancy networks: Learning 3d reconstruction in function space". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4460–4470.

[16] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. "Neural geometric level of detail: Real-time rendering with implicit 3d shapes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11358–11367.

[17] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. "Convolutional occupancy networks". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer. 2020, pp. 523–540.

[18] Chenxing Jiang, Hanwen Zhang, Peize Liu, Zehuan Yu, Hui Cheng, Boyu Zhou, and Shaojie Shen. "H2-Mapping: Real-time Dense Mapping Using Hierarchical Hybrid Representation". In: *arXiv preprint arXiv:2306.03207* (2023).

[19] Xuan Yu, Yili Liu, Sitong Mao, Shunbo Zhou, Rong Xiong, Yiyi Liao, and Yue Wang. "NF-Atlas: Multi-Volume Neural Feature Fields for Large Scale LiDAR Mapping". In: *arXiv preprint arXiv:2304.04624* (2023).

[20] Xingguang Zhong, Yue Pan, Jens Behley, and Cyrill Stachniss. "Shine-mapping: Large-scale 3d mapping using sparse hierarchical implicit neural representations". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 8371–8377.

[21] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. "Eslam: Efficient dense slam system based on hybrid representation of signed distance fields". In: *Proceedings of the IEEE/CVF Conference*

on Computer Vision and Pattern Recognition. 2023, pp. 17408–17419.

[22] Ignacio Vizzo, Tiziano Guadagnino, Jens Behley, and Cyrill Stachniss. "VDBFusion: Flexible and Efficient TSDF Integration of Range Sensor Data". In: *Sensors* 22.3 (2022). ISSN: 1424-8220. DOI: 10.3390/s22031296. URL: https://www.mdpi.com/1424-8220/22/3/1296.

[23] Xingrui Yang, Hai Li, Hongjia Zhai, Yuhang Ming, Yuqian Liu, and Guofeng Zhang. "Vox-Fusion: Dense Tracking and Mapping with Voxel-based Neural Implicit Representation". In: *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2022, pp. 499–507. DOI: 10.1109/ISMAR55827.2022.00066.

[24] Hai Li, Xingrui Yang, Hongjia Zhai, Yuqian Liu, Hujun Bao, and Guofeng Zhang. "Vox-Surf: Voxel-Based Implicit Surface Representation". In: *IEEE Transactions on Visualization and Computer Graphics* (2022), pp. 1–12. DOI: 10.1109/TVCG.2022.3225844.

[25] Jiawei Yang, Marco Pavone, and Yue Wang. "FreeNeRF: Improving Few-Shot Neural Rendering with Free Frequency Regularization". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023), pp. 8254–8263. URL: https://api.semanticscholar.org/CorpusID:257505220.

[26] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. "Co-SLAM: Joint Coordinate and Sparse Parametric Encodings for Neural Real-Time SLAM". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 13293–13302.

[27] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. "Fourier features let networks learn high frequency functions in low dimensional domains". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7537–7547.

[28] Antoni Rosinol, Torsten Sattler, Marc Pollefeys, and Luca Carlone. "Incremental visual-inertial 3d mesh generation with structural regularities". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8220–8226.

[29] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[30] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[31] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. "iMAP: Implicit mapping and positioning in real-time". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6229–6238.

[32] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. "isdf: Real-time neural signed distance fields for robot perception". In: *arXiv preprint arXiv:2204.02296* (2022).

[33] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. "MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2022).

[34] Jingwen Wang, Tymoteusz Bleja, and Lourdes Agapito. "GO-Surf: Neural Feature Grid Optimization for Fast, High-Fidelity RGB-D Surface Reconstruction". In: *2022 International Conference on 3D Vision (3DV)*. 2022, pp. 433–442. DOI: 10.1109/3DV57658.2022.00055.

[35] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. "Instant neural graphics primitives with a multiresolution hash encoding". In: *ACM Transactions on Graphics (ToG)* 41.4 (2022), pp. 1–15.

[36] Dongyu Yan, Xiaoyang Lyu, Jieqi Shi, and Yi Lin. "Efficient Implicit Neural Reconstruction Using LiDAR". In: *arXiv preprint arXiv:2302.14363* (2023).

[37] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. "Efficient geometry-aware 3D generative adversarial networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16123–16133.

[38] Wenbo Hu, Yuling Wang, Lin Ma, Bangbang Yang, Lin Gao, Xiao Liu, and Yuewen Ma. "Tri-MipRF: Tri-Mip Representation for Efficient Anti-Aliasing Neural Radiance Fields". In: *arXiv preprint arXiv:2307.11335* (2023).

[39] Christian Reiser, Rick Szeliski, Dor Verbin, Pratul Srinivasan, Ben Mildenhall, Andreas Geiger, Jon Barron, and Peter Hedman. "Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes". In: *ACM Transactions on Graphics (TOG)* 42.4 (2023), pp. 1–12.

[40] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. "PET-NeuS: Positional Encoding Tri-Planes for Neural Surfaces". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 12598–12607.

[41] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. "K-planes: Explicit radiance fields in space, time, and appearance". In: *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023, pp. 12479–12488.

[42] William E. Lorensen and Harvey E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '87. New York, NY, USA: Association for Computing Machinery, 1987, pp. 163–169. ISBN: 0897912276. DOI: 10.1145/37401.37422. URL: https://doi.org/10.1145/37401.37422.

[43] L. Wiesmann, T. Guadagnino, I. Vizzo, N. Zimmerman, Y. Pan, H. Kuang, J. Behley, and C. Stachniss. "LocNDF: Neural Distance Field Mapping for Robot Localization". In: 8.8 (2023), pp. 4999–5006. ISSN: 2377-3766. DOI: 10.1109/LRA.2023.3291274.

[44] Milad Ramezani, Yiduo Wang, Marco Camurri, David Wisth, Matias Mattamala, and Maurice Fallon. "The newer college dataset: Handheld lidar, inertial and vision with ground truth". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 4353–4360.

[45] Daniel Adolfsson, Stephanie Lowry, Martin Magnusson, Achim J. Lilienthal, and Henrik Andreasson. "A Submap per Perspective - Selecting Subsets for SuPer Mapping that Afford Superior Localization Quality". In: *European Conference on Mobile Robots*. 2019.