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

2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3503-7770-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IROS58592.2024.10802100

Fast Spatial Reasoning of Implicit 3D Maps through Explicit Near-Far Sampling Range Prediction

Chaerin Min*[2], Sehyun Cha*[3], Changhee Won[1], and Jongwoo Lim†[1,4]

***Abstract*— 3D mapping is critical for many robotics applications, such as autonomous navigation and object manipulation. Recently, deep implicit mapping approaches have received much attention for their compactness and ability to represent fine-grained details. However, without explicit guidance, such implicit representations are often cumbersome for searching the full range on the rays to find the object surfaces. As a result, several approaches, including hierarchical sampling, occupancy grids, and zero-level set baking, have been proposed to improve sampling where costly forward passes of the neural network should be performed. However, hierarchical sampling is still suboptimal in that it requires uniform coarse samples. Discrete occupancy grids of Instant NGP and zero-level sets of various baking methods are less suitable for large and noisy real scenes.**

In this paper, we present a novel framework for adaptively predicting the near-far range for sampling the query positions of the deep implicit map. For this purpose, the truncated signed distance grid for the map is pre-constructed and used to provide hints for near-far prediction during rendering. In addition, our recovery algorithm automatically detects failed near-far predictions and recovers only those rays by directly using the implicit map. We conduct extensive experiments on a synthetic dataset, a public real dataset, and a real dataset captured by our multi-camera robot system. The experimental results show that our algorithm achieves the same rendering quality with surprisingly fewer samples compared to the existing methods, which means that the robot can reason about the image and depth properties of the scene much faster. Finally, a thorough analysis of the sample distribution along the rays is provided to give a better understanding of our method's strong efficiency, adaptability, and robustness. https://chaerinmin.github.io/TSDF-sampling/

I. Introduction

In robotics, rendering of 3D maps is critical for understanding the environment, navigating effectively, and achieving higher levels of autonomy. Traditional map representations such as grid [1], [2], [3] or feature-based maps [4], [5] support basic navigation but have some limitations in complex 3D environments. Recent advances in implicit representations [6], [7], [8], [9], [10] using deep neural networks have facilitated the rendering of high-fidelity images of 3D scenes. This significantly increases the precision of robot localization and improves their ability to interact in complex environments. To extract visual or geometric information from the implicit map, the sampling phase along each ray is essential. Inadequate sampling can result in missing the surface, which leads to a catastrophic accident. Conversely, excessive sampling increases computational complexity, making it unsuitable for real-time robotic applications.

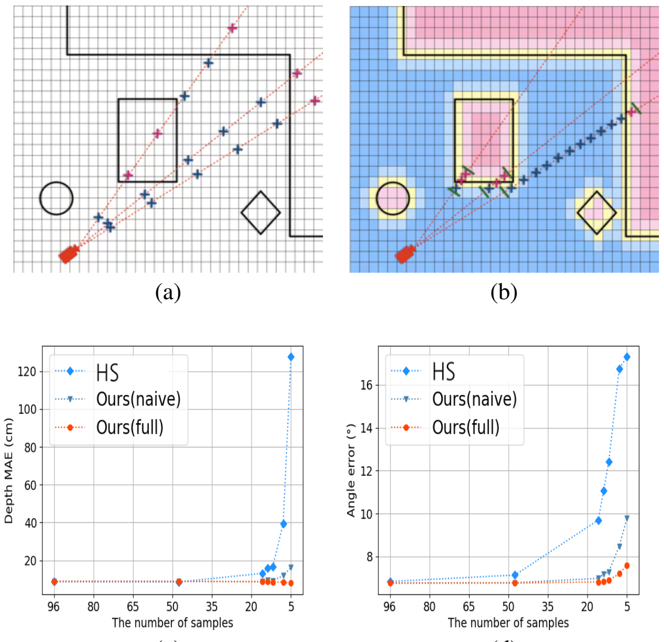
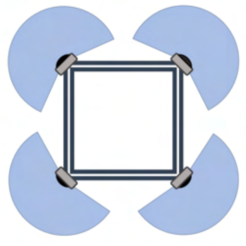


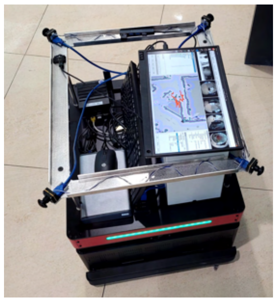
Fig. 1: (a): Baseline (hierarchical sampling). (b): Our method with near-far prediction. See the efficiency, accuracy, and robustness in the three rays, from left to right. **Both (a) and (b) have an average of 6 samples per ray.** Blue markers mean the model predicts SDF>0, and magenta markers SDF<0. Both (a) and (b) show only "coarse" samples for simplicity. (c-d): Trade-off between number of samples and accuracy. HS stands for Hierarchical Sampling, and "naive" is without adaptive sampling.

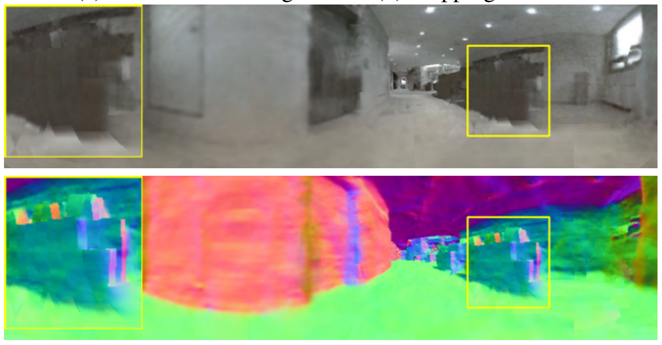
We break through this limitation by incorporating the traditional Truncated Signed Distance Field (TSDF) [11]. Essentially, our method, using the low-resolution grid as prior knowledge, sets reasonable range bounds for sampling per individual ray, focusing on areas where the surface is likely to be located. We also double-check and handle the erroneous bounds of the TSDF for proper rendering. As a result, our adaptive sampling allows for more sampling in difficult rays and less sampling in easier rays, thus minimizing unnecessary sampling.

The map plays a key role for autonomous robots because it is needed during the robot's execution. Therefore, the map should be efficient in providing images and depth maps to the robot. For this reason, several approaches have tried different

*Equal Contribution, †Corresponding Author
[1] MultiplEYE Co. Ltd.
[2] Department of Computer Science, Brown University
[3] Advanced Robotics Lab., CTO Division, LG Electronics
[4] Department of Mechanical Engineering, Seoul National University


(a) camera FOV design


(b) mapping robot


(c) Instant-NGP [12]

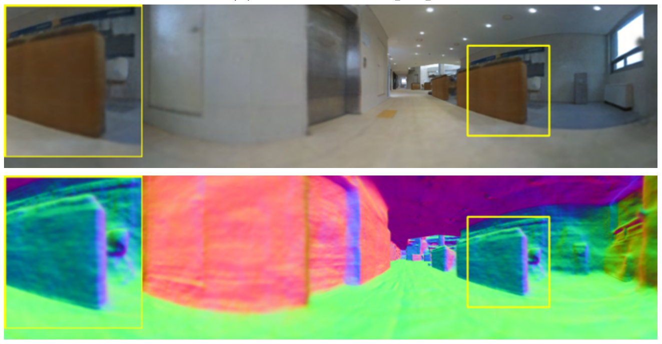

(d) Ours

Fig. 2: The mapping robot (a,b) and the result of a large real scene where our robot is driving (c,d). Thanks to the robust near-far paradigm, ray-level adaptive sampling, and recovery algorithm, our system provides more faithful results in a driving scenario.

sampling strategies of the neural map, including hierarchical sampling. However, such a two-stage hierarchical sampling approach is suboptimal in that it needs to explore the whole area with the neural network in the first stage. Therefore, fusion with an explicit representation is needed, and the explicit discrete representation should be robust enough to be seamlessly applied to real-world noisy environments.

In this paper, we present a sampling guidance method that can act as a modular building block for implicit 3D maps, together with the recovery algorithm for advanced robustness in large real-world environments. We show the qualitative evaluation of our framework in a space captured by our camera-only robot, in Fig. 2. In summary, we propose three contributions to the use of explicit guidance for implicit 3D maps:

- Given a 3D map, we propose a novel ray-level adaptive sampling that focuses on areas that has high contribution to the rendered result.
- The proposed method improved robustness in large scenes by introducing adaptive sampling and recovery algorithm.
- Throughout the experiments in both public and in-the-wild dataset, our method showed the efficiency achieved by reducing the number of samples while maintaining performance.

II. Related Works

A. 3D Map Representations

Feature-based maps [4] detect and describe essential features such as points or edges within an environment. Despite the compactness of such maps, feature-based maps contain only sparse landmarks and thus cannot reason about spatial occupancy. The voxel representation [2], [16], [17] divides a 3D space into a grid of cubic cells, or voxels, of uniform size and shape. Although voxels are straightforward and thus widely used, such methods have high memory consumption and are difficult to scale up to a large and highly detailed environment due to the limited resolution over the entire space. We conclude that a balance between detail and computational efficiency is needed. The implicit representation [6], [10], [18], [7] uses continuous functions to encode 3D shapes. This allows for both memory efficiency and detail, with theoretically infinite resolution. However, the drawback lies in the computational intensity, which makes real-time applications challenging, especially for the neural implicit representation. Our approach improves the inference speed of implicit functions, mitigating this problem without sacrificing structural detail.

B. Sampling Strategies

Tab. I lists some representative related works. Pioneering works [3], [19], [20] used explicit maps. On the other hand, recent works [6], [12] have introduced neural 3D maps built only from RGB images using continuous spatial locations. For such neural 3D maps, the density or SDF emerges from the 3D locations, and thus sampling the 3D location is inevitable. Thus, the algorithm to decide where to sample in the continuous 3D space is an essential problem that has a tremendous impact on both rendering quality and computational cost. The most widely used sampling strategy is Hierarchical Sampling (HS), proposed by [6] and used by [7], [21], [14], [18]. Due to the suboptimal efficiency of HS, [12] adopted hash coding and occupancy grid and [13] presented mesh baking. However, the "cache" occupancy grid and the discrete nature of the mesh make them less ideal for large environments, such as robot driving. Considering the large memory consumption of [15], in this paper we propose an alternative approach that can improve the sampling of 3D neural maps. such that our framework should concentrate the sampling only on object surfaces and memory efficient.

	VoxBlox [3]	NeRF [6]	MonoSDF [7]	NGP [12]	BakedSDF [13]	AdaNeRF [14]	GS [15]	Ours
Technique for efficient sampling at rendering?	TSDF	Hier.	Hier.	Density grid	Mesh	MLPs	Splats	TSDF
Requires only RGB (depth not given)?	✗	✓	✓	✓	✓	✓	✓	✓
Continuous spatial location?	✗	✓	✓	✓	✗	✓	✓	✓
Robust at large real driving scenes?	✓	✓	✓	✗	✗	✓	✓	✓
Showed geometric performance?	✓	✗	✓	✓	✓	✓	✗	✓
Time efficient?	✓	✗	✗	✓	✓	✓	✓	✓
Memory efficienet?	✓	✓	✓	✓	✓	✗	✗	✓
No additional neural network training?	✓	✓	✓	✓	✓	✗	✓	✓

TABLE I: Comparison of our near-far prediction framework to related methods. Heir. means Hierarchical.

Algorithm 1: Near-Far Prediction

```
Input:
    $\mathbf{o}$: camera origin
    $\mathbf{v}$: ray (unit direction vector)
Require:
    $t_f$: original far bound of $t$
    $D_s$: surface criteria
    $M$: maximum number of steps to confirm inside
    $\mathscr{V}$: TSDF volume
Output:
    $t_n$ : near sample bound
    $t_f$ : far sample bound

/* Determine near bound $t_n$ */
$t \leftarrow 0$
$\mathbf{p} \leftarrow \mathbf{o} + t\mathbf{v}$
$X \leftarrow \textsc{Voxel}(\mathbf{p})$
while $\mathbf{p} \in \textsc{RenderingCube}$ do
    if $\mathscr{V}(\textsc{VoxelCenter}(X)) \le D_s$ then
        $t_n \leftarrow t$
        break
    end
    $X, t \leftarrow \textsc{NextVoxel}(X, t, \mathbf{v})$
    $\mathbf{p} \leftarrow \mathbf{o} + t\mathbf{v}$
end

/* Determine far bound $t_f$ */
$m \leftarrow 0$
while $\mathbf{p} \in \textsc{RenderingCube}$ do
    if $\textsc{ForAll}(X_{nb} \leftarrow \textsc{NeighborVoxels}(\mathbf{p});$
            $\mathscr{V}(X_{nb}) < 0)$ then
        $m \leftarrow m + 1$
    else
        $m \leftarrow 0$
    end
    $X, t \leftarrow \textsc{NextVoxel}(X, t, \mathbf{v})$
    $\mathbf{p} \leftarrow \mathbf{o} + t\mathbf{v}$
    if $m == M$ then
        $t_f \leftarrow t$
        break
    end
end
Return: $t_n, t_f$
```

Algorithm 2: TSDF integration

```
Input:
$\mathscr{P}$: set of rays
$\mathscr{B}$: NDC
$\mathbf{o}_i$: camera origin of i-th frame
$\mathbf{v}_{ij}$: j-th ray direction of i-th frame
$\hat{D}_{ij}$: j-th ray depth of i-th frame
Require:
$D_T$: truncated distance
Output:
updated TSDF volume $\mathscr{V}$
updated weight volume $\mathscr{W}$
for $(\mathbf{o}_i, \mathbf{v}_{ij}, \hat{D}_{ij}) \in \mathscr{P}$ do
    $\mathbf{p}^* \leftarrow \mathbf{o}_i + \hat{D}_{ij}\mathbf{v}_{ij}$
    $t \leftarrow 0$
    $\mathbf{p} \leftarrow \mathbf{o}_i + t\mathbf{v}_{ij}$
    $X \leftarrow \textsc{Voxel}(\mathbf{p})$
    while $\mathbf{p} \in \mathscr{B}$ do
        $\mathbf{c}_X \leftarrow \textsc{VoxelCenter}(X)$
        $s_X \leftarrow \textsc{clamp}(\mathbf{v}_{ij} \cdot (\mathbf{p}^* - \mathbf{c}_X), -D_T, D_T)$
        if $s_X > -D_T$ then
            $w_X \leftarrow \textsc{Weight}(s_X)$
            $\mathscr{V}(X) \leftarrow \frac{\mathscr{W}(X)\mathscr{V}(X) + w_X s_X}{\mathscr{W}(X) + w_X}$
            $\mathscr{W}(X) \leftarrow \mathscr{W}(X) + w_X$
            $X, t \leftarrow \textsc{NextVoxel}(t, \mathbf{v}_{ij})$
            $\mathbf{p} \leftarrow \mathbf{o}_i + t\mathbf{v}_{ij}$
        else
            break
        end
    end
end
Return: $\mathscr{V}$, $\mathscr{W}$
```

III. Methodology

In this section, we elaborate a novel approach to accelerate the efficiency of the existing sampling methods. We voxelize the implied geometry from the trained model, inspired by the classical TSDF integration. We incorporate a geometric prior to define an appropriate interval along each ray in which points are sampled. This effectively avoids sampling points that are unnecessary for the rendered results. In Sec. III-D, we introduce our further technique that adapts to different rays with different characteristics.

A. Preliminaries

Volume Rendering Volume rendering is one of the most essential parts of the Neural Surface Field. For instance, RGB images and depth maps can be obtained by the volume rendering. To render the RGB image $\mathbf{C}$, given the ray $\mathbf{v}$ starting at the camera origin $\mathbf{o}$, the rendering algorithm numerically integrates the color radiance $\mathbf{c}$ at sampled 3D points $\mathbf{p}(t) = \mathbf{o} + t\mathbf{v}$ on the ray for the sample set $\mathscr{T} = \{t_i\}_{i=1}^{m}$ as follows:

$$\hat{\mathbf{C}}_{\mathscr{T}}(\mathbf{o},\mathbf{v}) = \sum_{t_i \in \mathscr{T}} T(t_i)\,\alpha(t_i)\,\mathbf{c}(t_i), \tag{1}$$

where $\alpha(t_i) = 1 - \exp(-\sigma(t_i)\Delta t_i)$ and $T(t_i) = \prod_{j=1}^{i-1}(1 - \alpha(t_j))$ are the opacity and the accumulated transmittance of the i-th ray segment, respectively. $(\sigma(t_i), \mathbf{c}(t_i)) = \mathrm{MLP}(\mathbf{p}(t_i), \mathbf{v})$ denotes the density transformed from SDF value and the RGB color of the point $\mathbf{p}(t_i)$.

Hierarchical Sampling Instead of using a one-stage sampling for querying implicit 3D maps, previous works [6], [21] used hierarchical sampling; "coarse" and "fine". Eq. 1 serves as the PDF (Probability Density Function) along a ray, and hierarchical sampling computes inverse CDF to sample a new set of "fine" points. The "fine" samples are then concentrated at points where the CDF grows most rapidly, i.e., where the coarse sample had a high density. In the following sections, we will describe our alternative sampling approach and the experimental advantages of our efficiency and effectiveness.

B. TSDF Grid Integration

To render a novel-view image of the scene, we only need to sample the area near the object surface. Since the depth for a novel ray is not known, the traditional sampling methods test the entire range to find where the surface is. Our main intuition is that we can precompute the space occupancy from the trained neural network and store it in a 3D TSDF volume. Unlike the color values, it is possible to cache the occupancy for a point because it does not change for different viewing directions. At render time, the SDF value for a sample can be efficiently queried without inferencing the neural network.

A TSDF grid $\mathscr{V}$ is constructed by ray casting from all images in the training set. The surface point $\mathbf{p}_{\mathbf{v}}^{*}$ for a ray $\mathbf{v}$ is estimated from the depth of the trained network. The value of each voxel $\mathbf{x}$ intersecting with the ray is updated as the weighted average of the clamped SDF value $s_{\mathbf{x}}$,

$$s_{\mathbf{x}} = \mathrm{CLAMP}(\mathbf{v} \cdot (\mathbf{p}_{\mathbf{v}}^{*} - \mathbf{x}),\, -D_T,\, D_T), \tag{2}$$

where D_T is a user-defined saturation distance, to reduce the noisy and inaccurate depth estimates from the network. We provide the time consumed for the TSDF grid integration in Tab. IV. Note that the TSDF integration of the environment will be completed before the robot starts navigation.

C. Near-Far Prediction

In this section, we describe our method for minimizing the sampling of empty, unseen, or space inside objects. To do this, Algorithm. 1 determines the sampling bound (t_n, t_f) so that the object surface lies within it, using the pre-computed $\mathscr{V}$ to obtain rough information about the scene geometry. To determine t_n, we let $\mathbf{p}_k$ visit the voxels in $\mathscr{V}$ along a ray until it encounters a voxel that falls into the *surface* group, i.e., $s_{\mathbf{X}} \le D_s$. Then, to determine t_f, we march the $\mathbf{p}_k$ a few more voxels after hitting a $s_{\mathbf{X}} < 0$. It is important to make sure that the $\mathbf{p}_k$ is actually inside an object. To ensure this, we check a certain number of consecutive $\mathbf{p}_k$'s to see if all their neighboring voxels are negative (occupied).

Fig. 1 illustrates three representative examples of our motivation for finding t_n and t_f. The magenta markers illustrate samples inside an object, while the blue markers show the samples outside objects. For simplicity, only the first-stage samples are shown. For the left ray, ours (b) considers a much smaller range compared to the conventional method [7] (a), and we can use much fewer samples without sacrificing the quality. The middle ray shows why thin objects can disappear when there are not enough samples in (a), as in Fig. 1. The last case is when a ray passes very close to the near object but hits the surface of the far object. If n_f is set too close, the bound will not contain the real object surface and will produce an invalid color or depth. Therefore, we need to enforce n_f to be definitely inside an object.

There are rare cases where the sampling bound does not contain the true surface, and thus produces invalid rendering results. We can detect this by checking the TSDF weights in the bound, since the weight sum must be close to 1 if the surface is inside the bound. In such cases, we revert to the conventional method of sampling the entire ray range with more samples. See Sec. III-E for a detailed discussion.

D. Adaptive Sampling

Most existing sampling methods [6], [22], [18], [21] sample the same number of points in the same depth range over all rays. Our algorithm adaptively finds the sampling range per ray according to the scene geometry, and we need to determine how many samples are needed. If the number of samples for all rays is set uniformly to a small number, the sampling interval for rays with a long range will become too large, resulting in a failure to find the surface at the coarse scale, and it cannot be recovered even with subsequent sampling.

For more efficient sampling and to eliminate the possibility of missing the surface in coarse sampling, we sample the points at equal intervals within the sampling range along the ray, and obviously the number of samples is proportional to the length of the sampling range. Since our sampling ranges are very tight for most rays, our approach can maintain rendering quality with a much smaller average number of samples than conventional methods.

E. Recovery Algorithm

We further investigate the ability of our method to work with a pre-trained model with less accurate geometry, such as NeRF[6], which was designed as a neural radiance field and thus has noisy geometry. Although NeRF does not explicitly handle SDF, our method can be applied to NeRF since our

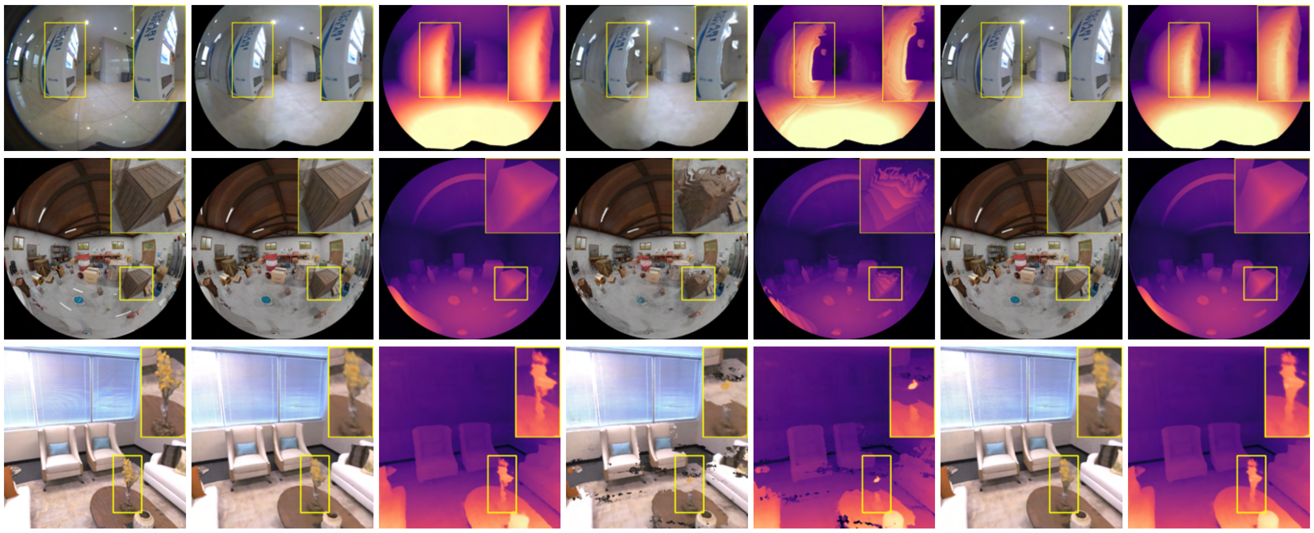

(a) GT (b) Baseline with many samples (c) Baseline with less samples (d) Ours with less samples

Fig. 3: Qualitative comparison against baseline (hierarchical sampling)

Sampling	Samples	Time[μs]	PSNR	SSIM	Depth	Normal
Error Bound	64+32	79.10	26.90	0.979	9.36	6.73
Hierarchical	64+32	28.34	28.05	0.980	**8.72**	6.84
Ours (naive)	64+32	26.09	**28.06**	**0.981**	8.89	6.77
Ours (full)	64+32	26.39	28.06	0.980	8.90	**6.76**
Hierarchical	6+8	8.81	24.46	0.955	15.87	11.07
Ours (naive)	6+8	8.84	26.60	0.973	9.50	7.18
Ours (full)	6+8	7.60	**28.04**	**0.980**	**8.65**	**6.85**
Hierarchical	6+6	8.61	23.77	0.947	16.57	12.42
Ours (naive)	6+6	8.62	26.49	0.972	9.23	7.28
Ours (full)	6+6	7.30	**28.00**	**0.980**	**8.50**	**6.89**

TABLE II: Quantiative comparison on the synthetic **Garage** dataset. Time is time per ray. Depth errors are in cm and Normal errors are in degree.

Sampling	Samples	Time [μs]	PSNR	SSIM
Error Bound [18]	64+32	70.40	19.99	0.851
Hierarchical	64+32	40.92	**20.01**	**0.848**
Ours (naive)	64+32	40.21	20.00	0.847
Ours (full)	64+32	30.87	19.99	0.847
Hierarchical	6+8	10.04	18.56	0.759
Ours (naive)	6+8	9.84	18.61	0.759
Ours (full)	6+8	8.94	**19.93**	**0.845**
Hierarchical	6+6	9.16	18.42	0.753
Ours (naive)	6+6	9.36	18.58	0.762
Ours (full)	6+6	8.89	**19.77**	**0.838**

TABLE III: Quantiative comparison on the real **Lobby** dataset. Time is per ray.

algorithm requires only the occupancy estimates from the model and can serve as a modular building block for a variety of volume rendering models.

In Fig. 5, the challenging fine structures show vulnerability when the reduced sampling range is applied naively. Given these challenges, to increase the flexibility of our method, we present another modification of our method: the recovery algorithm.

If the sum of the weights in a ray turns out to be less than a threshold, we recover the ray by resetting the sampling range to the full range and reconducting the sampling. It turns out that only 11% of the total rays trigger recovery on NeRF with reduced samplings, and the result is on par with [6], while cutting the inference time in 30% than [6].

IV. Experimental Results

We evaluate our proposed method using MonoSDF on a real large scene (Lobby), a synthetic scene (Garage) and a public dataset (Replica [23]). [7] has several encoding options, including single resolution, MLP, and hash encoding from [12], in which we chose the multi-resolution hash encoding for its overall superior quality and efficiency shown in [7]. Additionally, we apply our method to NeRF [6] on

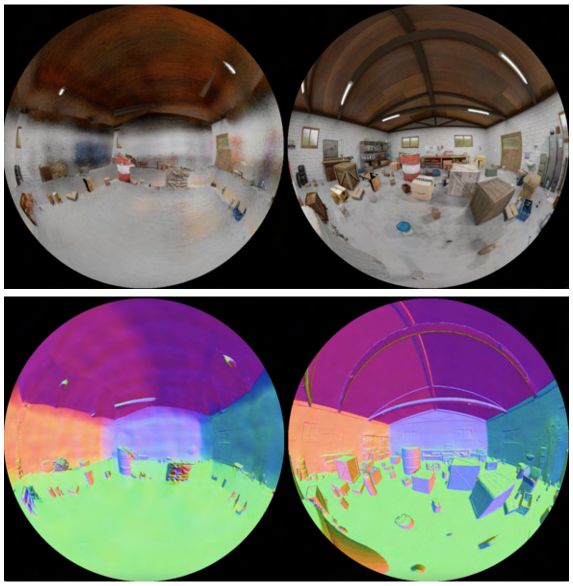

(a) HS average 4 samples (b) Ours average 4 samples

Fig. 4: The near-far prediction works robustly even with a very limited number of samplings per ray. Note the fine structures on the ceiling and the shelf near the windows. HS denotes Hierarchical Sampling.

Dataset	Garage	Lobby	Replica
# of rays	33M	782M	14M
Original near-far [m]	6.24	15.46	1.35
Reduced near-far [m]	1.15	3.80	0.08
# of surface not inside near-far	134	26.1k	2k
% of surface not inside near-far	0.00003	0.0004	0.014
Precomputation time [s]	2.99	30.44	1.57
Grid memory [MB]	67.1	537	67.1

TABLE IV: Analysis on our TSDF grids for near-far prediction

Approaches	Samples	Recovery	Time [μs]	PSNR [dB]	SSIM
Hierarchical	64+128	-	46.31	34.05	0.995
Hierarchical	32+64	-	17.36	28.04	0.980
Hierarchical	16+32	-	8.88	21.91	0.923
Ours (naive)	16+32	-	11.8	27.57	0.979
Ours (full)	16+32	-	8.41	27.57	0.980
Ours (naive)	48+0	✓	17.63	33.41	0.994
Ours (full)	48+0	✓	16.41	**33.65**	**0.995**

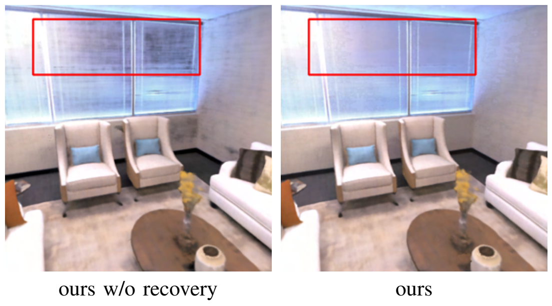


Fig. 5: Ablation study of our Recovery algorithm on the Replica dataset

Replica to verify the effectiveness when the network has poor geometry information.

A. Datasets

Lobby This is a real-world dataset that was captured using a mobile robot with four ultra-wide field-of-view (FOV) fisheye cameras in an indoor scene. We applied [5] to the data to obtain the camera poses in the scene. **Lobby** contains 388 images with a resolution of 1344×1080 and a FOV of 220 degrees. To mimic the depth and normal estimation in [7], we estimate the depth maps from the images using [24] and compute the surface normal map by backprojecting the 2D points using depth and performing the cross product between the nearest backprojected points.

Garage This dataset was created with Blender from a 3D model of a garage with various objects. The camera setup is the same as in Lobby. The Garage dataset consists of 80 images of size 832×832 and their corresponding depth and normal maps. While only using the RGB for training, we use this dataset to evaluate accuracy of our method's depth and normal.

Replica We use the officially provided Replica [23] dataset from MonoSDF [7], which contains the monocular depth and noromal estimations from Omnidata [25]. We choose room 0, which consists of 100 images of size 384×384.

B. Implementation Details

In all our experiments, the TSDF grid resolution is set to 512^3. When the pre-trained model is MonoSDF [7], we set the maximum distance D_T and D_s to $5\times$ and $1\times$ the voxel size, respectively. To determine t_f in the near-far prediction process, we explore all $5 \times 5 \times 5$ voxels around the voxel of interest and test if they are inside an object. In the case of NeRF [6], due to the low quality of its density field, we set D_T and D_s to $39\times$ and $27\times$ of the voxel size and explore the $7 \times 7 \times 7$ neighboring voxels. To ensure that the voxel of interest is really inside an object, we break the second while loop in the algorithm 1 only when such a neighborhood criterion is satisfied 15 times in a row.

We used the standard PSNR, SSIM for image quality. We used MAE for depth error and angle in degree for normal error. For comparison, the standard hierarchical sampling in state-of-the-art methods is used, i.e. MonoSDF[7]. For the experiments shown in Fig. 2, we run the official code of [26], [12] on our large real dataset including real-world effect, e.g., reflection, texture-less surfaces, etc,. We used the same hyper-parameters as the original papers, including the network size and the feature grid resolution. Note that we used the 2048 resolution for the NGP feature grid as they used, while we still adopted 512 resolution TSDF grid for our method. Nevertheless, with even less resolution, the near-far guidance from a TSDF grid proved its robustness at the large driving scene. We believe this is because we avoided aggressive pruning that both [12] and [26] adopted for their mainly object-centric settings. Plus, our adaptive sampling strategy allowed the samples to be concentrated mostly on object edges and corners, making the overall rendering more efficient.

C. Evaluation of TSDF Integration

Tab. IV describes the extent to which the sampling range has been reduced for each dataset. For all datasets, the average sampling range was reduced by **24**% or less compared to the original range in rays of the train set. Then, the TSDF grid can be used to provide a strong guidance for numerous upcoming novel views. We analyze the cases where the depth used for integration does not fall within the reduced sampling range of our method. This rare exception can occur due to the discrete nature of the grid representation and the finite number of samples on a ray. To deal with failure these cases we introduce the recovery method to our algorithm as in Sec. III-E. This allows us to detect the rays where the true surface is outside the sampling range, and only for these rays, we increase the sampling range to the entire ray length, so that we can efficiently avoid such failure cases.

D. Performance

In Tab. III, when the number of samples is sufficiently large (96), the performance is almost the same for all metrics, regardless of the sampling method. However, when the number of samples is limited to 12, the PSNR of [7] and our method deviate significantly to 18.42 and 19.77, respectively, indicating the advantages of our method.

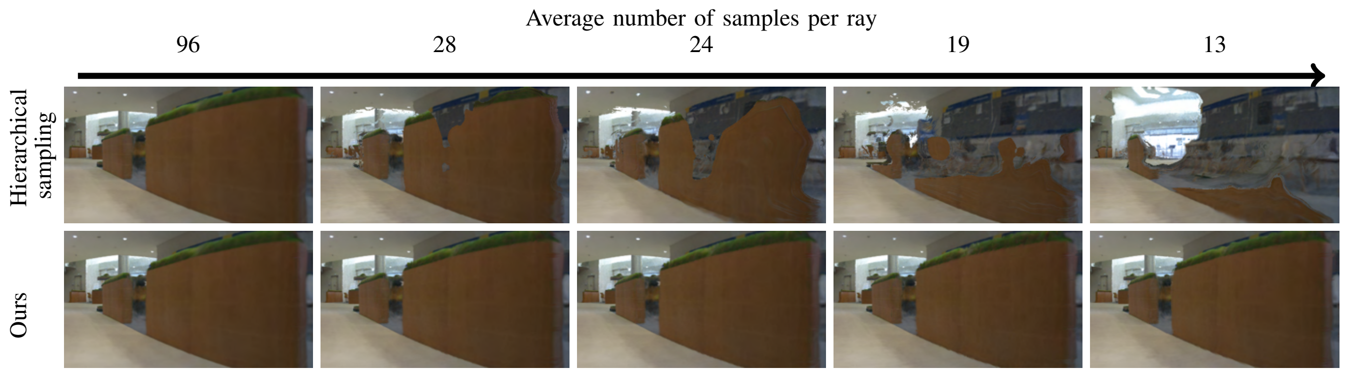


Fig. 6: Ablation study of reducing the samples.

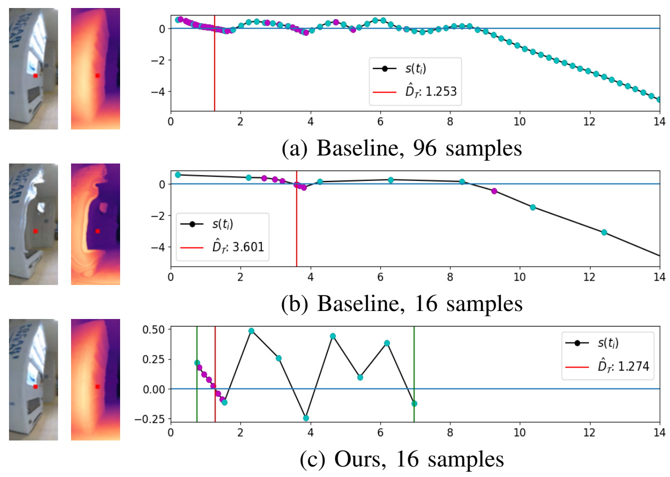


Fig. 7: Analysis of the sample distribution along a ray. Blue indicates the "coarse" samples. Magenta indicates the "fine" samples from the inverse CDF [6], [21], [18], [7] computation. This illustrates the need of using near-far, not only for the efficiency, but also for the accurate prediction of the true surface at challenging cases, such as object edges.

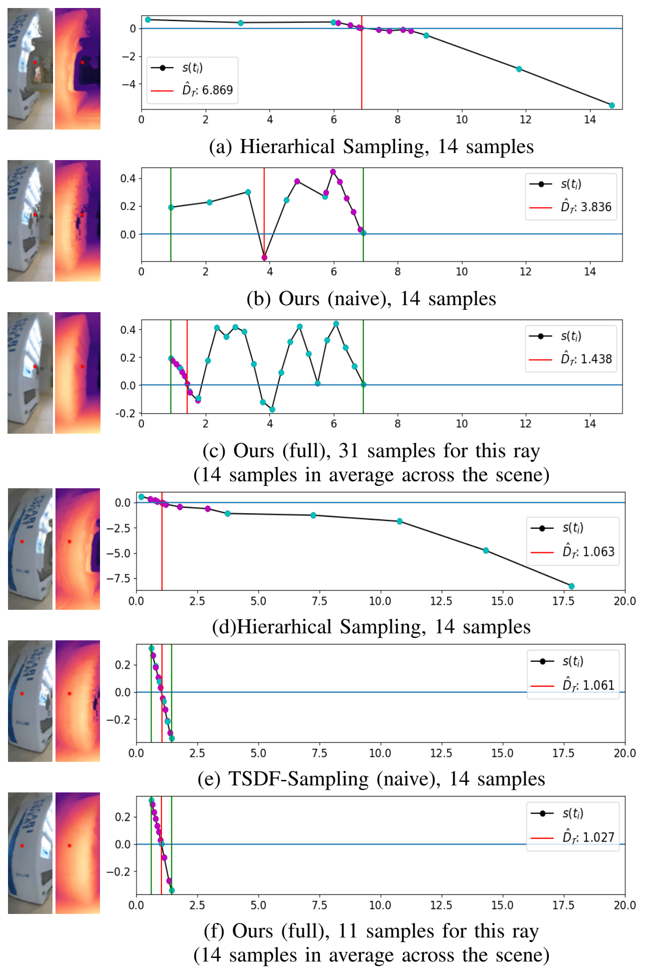


Fig. 8: (a)-(c) shows the challenging case when a ray hits an edge of an object, where the true surface fails to be detected by the reduced number of samples in (a) and (b). In (c), our adaptive near-far prediction automatically sets a narrower band than (a) and more samples than (b) for this particular ray, achieving successful rendering of the vending machine. (d)-(f) shows a ray hitting a plain surface, where (e) and (f) are more focused on the surface than (d). Furthermore, (f) automatically uses less samples for this easier ray.

In Fig. 3 (d), we demonstrate that simply reducing the number of samples causes the invisible area behind the near object to appear. Fig. 7 effectively shows that this is because the locations of the coarse samples (cyan) are too far apart, causing the first surface to be missing, which in turn causes the surface in the subsequent samples (magenta) to be missing. On the other hand, in (c), our method can find the surface with the same number of samples, because the course samples are distributed within our proposed sampling range. Additionally, While Fig. 7 shows a ray that penetrates the vending machine (medium hard), we show the cases of casting the ray to a plane (easy) and to an edge (hard) in Fig. 8. Fine samples (magenta) are upsampled from the PDF of the coarse samples (blue) by using the inverse CDF [6], [18]. s denotes the SDF values, and the rendered depths $\hat{D}_r$ on rays r are in meters. The green lines shows t_n and t_f of the TSDF-Sampling approach.

We evaluate two additional metrics (depth mean square error and normal angle error) on Garage to show that we can preserve not only the photometric quality but also the geometric performance. Tab. II shows that as the number of samples decreases, our method performs remarkably on par with results from much larger numbers of samples, with respect to depth and normal. However, [7] incurs a notable performance overhead of about twice the increase in both depth and normal errors due to the missing first surfaces.

E. Ablation Study

We compare the performance of the naive method and that of our proposed method, including the adaptive number of samples on each ray, according to the length of the sampling range. The naive method uses the same number of samples for each ray. Fig. 1 shows the stark differences in performance when the number of samples is reduced from 96 to 5. When the number of samples is halved, our naive method still maintains rendering quality. However, when the number of samples is reduced to less than 16, the naive method starts to lose performance, while our full method is much less affected. Fig. 6 on the Lobby dataset effectively shows the motivation of using the proposed near-far framework.

V. Conclusion

3D mapping of an environment is essential for robotic applications. We propose a novel framework to guide the implicit mapping with an adaptive and robust volumetric 3D representation. Our narrow band TSDF grid guides the mapping in a ray-by-ray adaptive manner. Moreover, our proposed framework accelerates the process of obtaining images and depth maps. We show that combining neural 3D maps with our framework can more efficiently provide images and depth maps in large real scenes.

Limitations Our method proposed the post-processing of a 3D mapping for efficient rendering. Therefore, the performance of the rendering could depend on the mapping model. Given the rapid advancement of 3D mapping techniques, We envision performance improvement in the future, when adopting our TSDF-Sampling before starting the rendering.

Acknowledgment

This work was supported by the Technology Innovation Program (No. 20018110, "Development of a wireless tele-operable relief robot for detecting searching and responding in narrow space") funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea), and Sehyun Cha's work was supported by LG Electronics.

References

[1] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[2] J. S. De Bonet and P. Viola, "Poxels: Probabilistic voxelized volume reconstruction," in *Proceedings of International Conference on Computer Vision*, vol. 2, 1999, p. 2.

[3] 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*, 2017.

[4] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, 2015.

[5] C. Won, H. Seok, Z. Cui, M. Pollefeys, and J. Lim, "Omnislam: Omnidirectional localization and dense mapping for wide-baseline multi-camera systems," in *IEEE International Conference on Robotics and Automation*, 2020.

[6] 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," *Communications of the ACM*, vol. 65, pp. 99–106, 2021.

[7] Z. Yu, S. Peng, M. Niemeyer, T. Sattler, and A. Geiger, "Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction," *Advances in neural information processing systems*, 2022.

[8] 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*, 2022, pp. 12 786–12 796.

[9] A. Rosinol, J. J. Leonard, and L. Carlone, "Nerf-slam: Real-time dense monocular slam with neural radiance fields," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2023.

[10] X. Kong, S. Liu, M. Taher, and A. J. Davison, "vmap: Vectorised object mapping for neural field slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 952–961.

[11] B. Curless and M. 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.

[12] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 1–15, 2022.

[13] L. Yariv, P. Hedman, C. Reiser, D. Verbin, P. P. Srinivasan, R. Szeliski, J. T. Barron, and B. Mildenhall, "Bakedsdf: Meshing neural sdfs for real-time view synthesis," *arXiv preprint arXiv:2302.14859*, 2023.

[14] A. Kurz, T. Neff, Z. Lv, M. Zollhöfer, and M. Steinberger, "Adanerf: Adaptive sampling for real-time rendering of neural radiance fields," in *European Conference on Computer Vision*. Springer, 2022, pp. 254–270.

[15] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–14, 2023.

[16] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *International journal of computer vision*, vol. 38, pp. 199–218, 2000.

[17] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," *International Journal of Computer Vision*, vol. 35, pp. 151–173, 1999.

[18] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume rendering of neural implicit surfaces," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4805–4815, 2021.

[19] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[20] Y. Pan, Y. Kompis, L. Bartolomei, R. Mascaro, C. Stachniss, and M. Chli, "Voxfield: Non-projective signed distance fields for online planning and 3d reconstruction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

[21] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," *arXiv preprint arXiv:2106.10689*, 2021.

[22] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5855–5864.

[23] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[24] C. Won, J. Ryu, and J. Lim, "End-to-end learning for omnidirectional stereo matching with uncertainty prior," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, pp. 3850–3862, 2020.

[25] A. Eftekhar, A. Sax, J. Malik, and A. Zamir, "Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 786–10 796.

[26] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *Advances in Neural Information Processing Systems*, 2020.