NISB-Map: Scalable Mapping With Neural Implicit Spatial Block

Beichen Xiang, *Graduate Student Member, IEEE*, Yuxin Sun, Zhongqu Xie, Xiaolong Yang, and Yulin Wang

***Abstract*—Recently, neural implicit representations have been applied in the mapping process of simultaneous localization and mapping (SLAM), accompanied by less storage overhead and continuous representation. Nevertheless, related methods use a single neural network to represent the whole scene, resulting in forgetting the observed regions caused by the limited capacity of a single network in the large-scale scene. Several methods encode the scene into implicit voxels to avoid parameter forgetting while the memory is sacrificed. In this letter, we introduce a scalable mapping framework that utilizes extensible Neural Implicit Spatial Blocks (NISB) with fixed size to cover the entire scene by incrementally creating multiple Multi-Layer Perceptron (MLP) networks. In evaluations against alternative methods on 3 datasets of indoor environments, our method Avoids forgetting the observed areas during the mapping process with a small memory footprint and smoothly updates the global map at 2 Hz.**

***Index Terms*—Deep learning for visual perception, mapping.**

I. Introduction

MAPPING plays a vital role in many applications on mobile robots, from autonomous driving and drone flight to target retrieval and relocation. Even though classical methods such as point clouds, voxel grids, surfels, and signed distance fields (SDF) can realize real-time volumetric representation by saving as discrete maps, it is tough to achieve a balance between memory consumption and mapping resolution in the process of large-scale scene mapping, resulting in the bottleneck of 3D scene understanding and global foothold planning for robots.

Neural implicit representation has shown promising influences on object tracking [1], navigation [2], and volumetric reconstruction [3], [4], [5], especially when neural radiance field (NeRF) [6] emerges as a compelling strategy for learning to render a novel view. NeRF [6] utilizes a continuous volumetric function, parameterized as a compact multilayer perceptron (MLP), to map from each 5D coordinate (3D position and 2D view direction) to volume density and view-dependent emitted color flexibly, ensuring the continuity and efficiency of the data.

Manuscript received 12 March 2023; accepted 19 June 2023. Date of publication 26 June 2023; date of current version 30 June 2023. This letter was recommended for publication by Associate Editor X. Chen and Editor J. Civera upon evaluation of the reviewers' comments. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB4701500 and in part by the National Natural Science Foundation of China under Grant 52105023. *(Corresponding author: Yulin Wang.)*

The authors are with the Department of Mechanical Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: blacklioneer96@gmail.com; sjtusun@foxmail.com; xiezq@seu.edu.cn; xiaolongyang@njust.edu.cn; wyl_sjtu@126.com).

Digital Object Identifier 10.1109/LRA.2023.3289152

Several methods [7], [8] extend NeRF [6] to complex scenes, but the geometric consistency of scene is challenging to be guaranteed in featureless surfaces, so concurrent works [9], [10] incorporate depth priors to render density to near-surface regions on novel view synthesis. Accompanied by the continuous property of volumetric function in NeRF [6], some works [11], [12] show applicability in semantic representation with mismatch semantic correction and autonomous prediction with few samples. However, restricted by the model capacity, most of the abovementioned methods are difficult to scale up to large-scale scenes.

To address the limitation of one single network, one direct way is to use multiple networks to represent the whole scene. KiloNeRF [14] divides the scene into thousands of tiny MLP networks with a single large-capacity MLP as a teacher beforehand. Wu et al. [15] separate the indoor scene into many small tiles containing two tiny MLP networks to capture complex appearance highlights with fewer training images. A voxel octree for each view-independent MLP is implemented to fulfill fast rendering. Acron [16] divides a scene into multiple adaptive blocks by solving an Integer Linear Problem to focus on computation on harder regions. Combined with appearance embedding, Block-NeRF [17] places multiple overlapping NeRFs [6] at each intersection to realize city-scale representations without weather and lighting influence. However, focuses on detailed reconstruction with dense samples results in training in hours even days, which cannot meet the needs of online mapping for robots.

iMAP [18] is the first SLAM pipeline using neural implicit representation for both mapping and tracking, while iSDF [19] puts more emphasis on mapping neural sign distance fields in SLAM problems combined with poses from state-of-the-art visual odometry. With depth measurements and sparse active sampling, both achieve almost real-time mapping capability. However, observed regions forgetting caused by limited single network capacity still restricts the possibility of applying to large-scale scenes. By encoding the scene into hierarchical implicit voxel grids, NICE-SLAM [20] dexterously avoids parameter forgetting. Nevertheless, the memory of hierarchical implicit voxel grids with 32 dimensions is needed more than pure MLP networks. Similarly, InstantNGP [21] encodes the scene into the multiresolution hash voxel vertices to realize the real-time reconstruction. However, to maintain a similar reconstruction quality in large room-scale reconstruction, the hash table size increases exponentially while the memory footprint grows linearly with the hash table size.

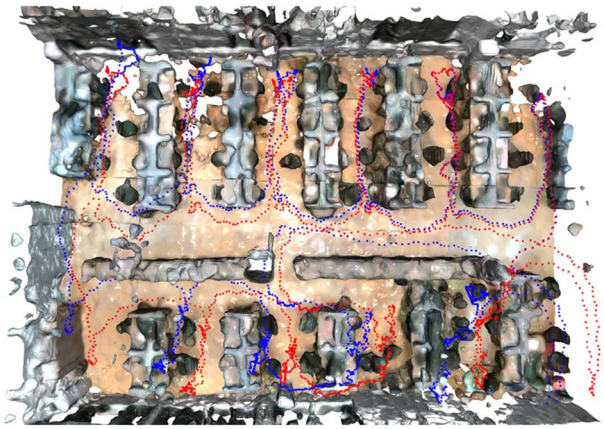

Fig. 1. Mapping in an Indoor Office over 200 m^2. The data is captured by a custom visual odometry, which is adapted from ORB-SLAM3 [13]. The blue trajectory is the visual odometry output, while the red one is optimized by our method. NISB-MAP reduces the cumulative drift during the mapping process and smoothly reconstructs the entire scene.

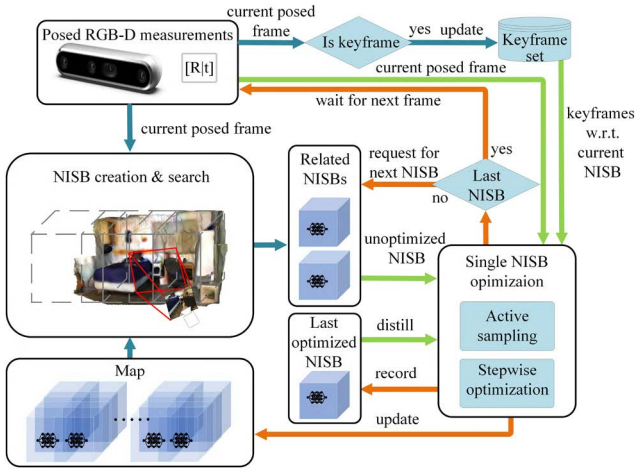

Fig. 2. Overview of the NISB-MAP. Arrows of the same color are transmitted at the same time step.

Different from encoding into voxels, we seek to leverage multiple small Multi-Layer Perceptron (MLP) networks representing neural implicit spatial blocks (NISB) to realize scalable mapping for large-scale indoor scenes with small memory consumption. To this end, we introduce NISB-MAP. Given a stream of posed RGB-D images, we dynamically create NISBs with fixed cube sizes, separating the large-scale environment into multiple small spatial blocks. Combined with sparse sampling strategies with depth priors, NISB-MAP keeps low memory consumption and a relatively fast learning speed. We found that sparsely sampling each ray results in different output density levels at the same location within two adjacent spatial blocks' overlap regions, leading to density discontinuity between adjacent blocks. For boundary artifacts caused by different densities between any two adjacent NISBs, we set each NISB to overlap with each other and apply knowledge distillation to overlapping areas to ensure geometric continuity. By minimizing the re-rendering losses, each NISB w.r.t. the current frame is optimized in turn. Optionally, with semantic masks as additional input, our method is also capable of implicit semantic representations and culling specified objects in NISBs. In evaluation against neural-implicit-representation-based mapping methods, we show the scalability of our method on two RGB-D datasets and a self-captured large-scale indoor dataset.

In summary, we make the following contributions:

- An incremental mapping framework is proposed to compensate for the non-scalability of current mapping methods based on neural implicit representations in the large-scale indoor environment.
- The ability to automatically keep the output density of any two adjacent NISBs at the same level with the assistance of block distillation.
- Experimental tests in the self-captured dataset (as shown in Fig. 1) demonstrate that our method allows work in the real environment and can reduce the cumulative drift from the visual odometry with stepwise optimization.

II. Methods

We focus on mapping rather than the full SLAM problem and therefore assume there is an external tracking process. Fig. 2 illustrates the operation process of the NISB-MAP. With a stream of RGB-D posed frames captured by a moving camera and corresponding semantic masks (optional). NISB-MAP automatically generates overlapping spatial blocks by searching within the current viewing frustum (II-A). Each spatial block (II-B) searches for relevant keyframes through active sampling and obtains a series of spatial points from the rays (II-C). Then the relative spatial block outputs the density, color, and semantic logit (optional) of the corresponding position to render the final results of each ray (II-D). When training the new block, we apply knowledge distillation to the areas that overlap with the previous block to ensure continuity (II-E). Finally, NISB-MAP incrementally optimizes the weights and camera poses w.r.t. the current spatial block by the stepwise optimization strategy (II-F). After optimizing all NISBs w.r.t. the current frame, the system waits for the next frame as input.

A. Incremental NISB Searching and Creation

Inspired by Block-NeRF [17], we place the whole environment within multiple cube NISBs with a 50% overlap between any adjacent NISBs to ensure the consistency of scenes. Once receiving a new RGB-D frame with pose $\mathbf{T}_c$, we uniformly sample N pixels and calculate the ray direction:

$$\mathbf{r}_i = \mathbf{T}_c \mathbf{K}^{-1} \left[u_i, v_i\right], i \in \{1, \ldots, N\}, \tag{1}$$

With the depth value D_i of the corresponding ray and the distance value t_{far} set by the valid distance of the RGB-D camera, each ray uniformly samples M points from 0 to $\min\{D_i, t_{far}\}$ distance:

$$\mathbf{x}_{ij} = \mathbf{o}_c + t_j \mathbf{r}_i, j \in \{1, \ldots, M\}, \tag{2}$$

According to the center position $\mathbf{p}_k, k \in \{1, 2, \ldots\}$ w.r.t. relative NISBs, and the distance a between any two adjacent NISBs, we search for all NISBs related to the current viewing frustum by

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

judging whether there is a NISB to accommodate the sampled points:

$$\|\mathbf{x}_{ij} - \mathbf{p}_k\|_\infty \leq a/2. \tag{3}$$

New NISBs will be created if corresponding NISBs are not found in the current viewing frustum. In all experiments we set $N = 16$, $M = 4$, $t_{far} = 4$ m, and $a = 4$ m.

B. Single Neural Implicit Spatial Block Architecture

Similar to the network architecture in iMAP [18], each NISB is composed of an MLP network with 4 128-width hidden layers, outputting the color $\mathbf{c}$ and the density σ for the spatial position $\mathbf{x}$ with:

$$F_\theta(\mathbf{x}) = (\mathbf{c}, \sigma), \tag{4}$$

where θ is network parameters. Following iSDF [19], we apply the "off-axis" positional embedding mentioned in [22] to transform 3D coordinates into high dimensional vectors before serving as input to the MLP network:

$$\gamma(\mathbf{x}) = \left(\alpha\left(2^0\mathbf{A}\mathbf{x}\right), \ldots, \alpha\left(2^5\mathbf{A}\mathbf{x}\right)\right), \tag{5}$$

where the function $\alpha(\cdot) = (\sin(\cdot), \cos(\cdot))$, and $\mathbf{A} \in \mathbb{R}^{21\times 3}$ is the unit-norm vertices of a twice-tessellated icosahedron. F_θ also outputs a feature vector $\mathbf{g}$ from the last hidden layer. If the input also has 2D semantic labels, the second MLP network with 64 activations for each hidden layer is activated and takes the feature vector and embedding as input to output the semantic logit $\mathbf{s}$ with:

$$F_\omega(\mathbf{g}, \gamma(\mathbf{x})) = \mathbf{s}. \tag{6}$$

We use Tiny-cuda-nn [23] to replace raw networks implemented in PyTorch [24] to speed up the optimization process.

C. Active Sampling

Keyframe Selection: Within the associated NISBs of the current frame, we use a hybrid keyframe selection method to dynamically add keyframes by considering both the information gain from iMAP [18] and the variation of position and angle w.r.t. to the previous keyframe. Based on the keyframe depth values and pose, we maintain an index table to relate each keyframe to its corresponding NISBs for the following keyframe sampling.

Keyframe Sampling: Similar to iMAP [18], we actively sample keyframes to mitigate the network forgetting problem. For each NISB associated with the current frame, we follow iSDF [19] to sample N_{fm} frames at each iteration. In addition to the current frame, $N_{fm} - 1$ keyframes are randomly sampled from the loss distribution formed by the normalized running loss of each keyframe in the current NISB's keyframe set.

Ray Sampling: We randomly sample N_{pix} pixels for each selected frame and then calculate the ray direction $\mathbf{r}$ for each pixel with (1). If 2D semantic labels are available, specified pixels are culled to reduce the impact of dynamic objects. For each ray with the depth value D, we sample $M_{ray} = M_{strat} + M_{surf}$ distance values $t_i, i \in \{1, \ldots, M_{ray}\}$ along the ray. The distance values are composed of M_{strat} stratified samples in the range $[0, \min\{D, t_{far}\}]$, and M_{surf} random samples in the surface interval $[0.9D, 1.1D]$. Combined with the corresponding frame poses, we calculate the 3D coordinates of the sampled points $\mathbf{x}_i$ on each ray with (2).

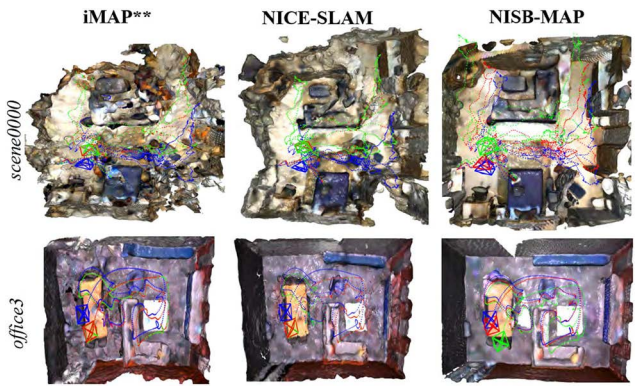

Fig. 3. Mapping Results with Drifting Poses. The green trajectory is the input poses with drifts, while the blue one and the red one are the ground-truth and optimized poses respectively. Mapping processes in iMAP** and NICE-SLAM [20] are unable to handle cumulative drift.

D. Volume Rendering

Based on the valid boundary of the current NISB $\mathbf{B}_k = \{\mathbf{x} | \|\mathbf{x} - \mathbf{p}_k\|_\infty \leq 3a/4\}$, we filter the valid rays by judging whether the current depth point of each ray $\mathbf{x}_D \in \mathbf{B}_k$. Then with M_{ray} points $\mathbf{x}_i$ of each valid ray, we get radiance $(\mathbf{c}_i, \sigma_i)$ and semantic logit $\mathbf{s}_i$ by (4) and (6):

$$(\mathbf{c}_i, \sigma_i, \mathbf{s}_i) = \begin{cases} (F_\theta(\mathbf{x}_i), F_\omega(\mathbf{g}_i, \gamma(\mathbf{x}_i))), & \text{if } \mathbf{x}_i \in \mathbf{B}_k \\ (\mathbf{0}, 0, (1, 0 \ldots, 0)), & \text{otherwise.} \end{cases} \tag{7}$$

Finally, we follow [18], [11] to render the depth $\hat{D}$, the color $\hat{\mathbf{I}}$, and the semantic logit $\hat{\mathbf{S}}$ of each ray:

$$\hat{D} = \sum_{i=1}^{M_{ray}} \omega_i d_i, \quad \hat{\mathbf{I}} = \sum_{i=1}^{M_{ray}} \omega_i \mathbf{c}_i, \quad \hat{\mathbf{S}} = \sum_{i=1}^{M_{ray}} \omega_i \mathbf{s}_i, \tag{8}$$

where $\omega_i = o_i \prod_{j=1}^{i-1}(1 - o_j)$ denotes the ray termination probability along the ray; $o_i = 1 - \exp(-\sigma_i \delta_i)$ is the occupancy probability; $\delta_i = \|\mathbf{x}_{i+1} - \mathbf{x}_i\|$ is the distance between adjacent points.

E. Block Distillation

During the mapping process, we found that individually training each NISB with sparse samples and few iterations tends to result in different density levels between two adjacent blocks. Inspired by KiloNeRF [14], distillation for the overlapping areas between NISBs is applied to diminish the density gap.

Different from pretraining a global NeRF [8] as the teacher model in KiloNeRF [14], we use the last trained NISB as the teacher and only distill the knowledge within the areas that overlap with the current NISB. Computation costs and time spent on training an additional global NISB are saved. Points within the overlapping area are selected from samples in II-C. We query both of NISBs to obtain the respective occupancy probability

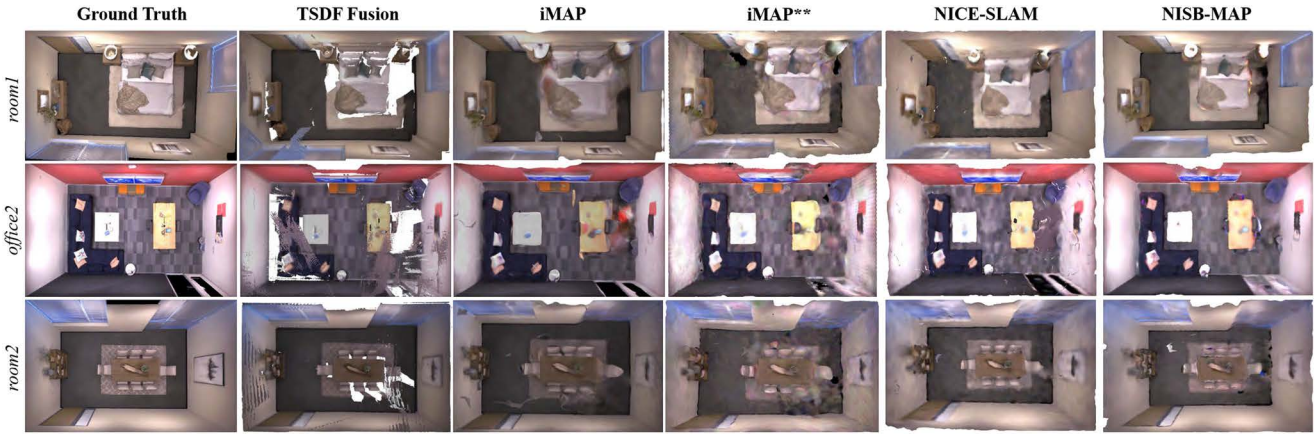

Fig. 4. Mesh outputs for replica [26].

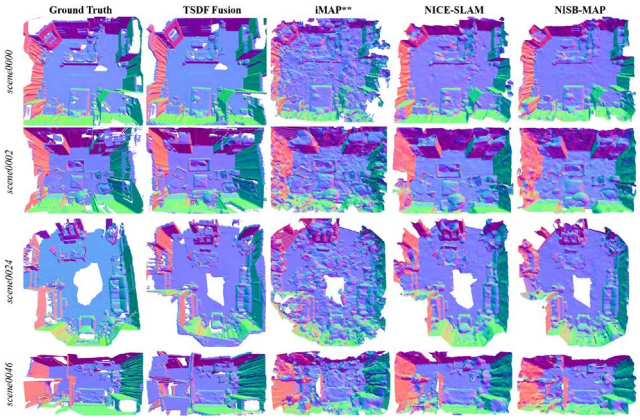

Fig. 5. Mesh outputs for ScanNet [25].

o_i^{last}, o_i^{cur} and optimize the current NISB's parameters using L_1 loss between them.

F. Stepwise Optimization

Pose refinement: Considering the cumulative drift brought by the visual odometry, we use a cumulative drift correction parameter $\mathbf{D} \in \mathbb{R}^{4\times4}$, which is initialized as an identity matrix before the first frame. Before optimizing each NISB, we update each raw input pose $\mathbf{T}_{raw}$ to $\mathbf{T}_c$ by:

$$\mathbf{T}_c = \mathbf{D}\mathbf{T}_{raw}. \tag{9}$$

For all NISBs associated with the current frame, it is difficult to ensure each has enough depth points within its valid range, resulting in incorrect pose optimization with insufficient measurements. For NISBs related to the current frame, we sort NISBs in descending order depending on the number of sampled points within each block and optimize each block one by one. If more than 60% of N_{pix} rays sampled from the current depth image terminate in the current NISB space, the current frame pose $\mathbf{T}_c$ is treated as the optimizable parameter and jointly optimized with the current NISB. $\mathbf{T}_c$ is updated by the optimized output at the end of the current NISB's optimization. We use the updated $\mathbf{T}_c$ as input to start optimizing the next NISB. After finishing optimizing all NISBs associated with the current frame, we update the current correction parameter by:

$$\mathbf{D} = \mathbf{T}_c\mathbf{T}_{raw}^{-1}. \tag{10}$$

Optimization: Followed by [18], we take the geometric loss $\mathcal{L}_g$ and the photometric loss $\mathcal{L}_p$ to scene reconstruction. Additionally, the free-space loss $\mathcal{L}_{fs}$ and the distillation loss $\mathcal{L}_{dt}$ are used for removing artifacts caused by sparse samples. We divide the optimization into 3 steps to sequentially optimize the current camera poses $\mathbf{T}_c$, scene geometry parameters θ, and semantic parameters ω (optional). In the first step, we optimize $\mathbf{T}_c$ with F_θ and F_ω frozen by considering the geometric loss $\mathcal{L}_g$ and the photometric loss $\mathcal{L}_p$. In the second step, we simultaneously optimize the pose of N_{fm} frames with F_θ, while freezing F_ω. In addition to $\mathcal{L}_p$ and $\mathcal{L}_g$, $\mathcal{L}_{fs}$ and $\mathcal{L}_{dt}$ are also taken into consideration. At last, if semantic masks are available, $\mathcal{L}_{dt}$ and the cross-entropy loss $\mathcal{L}_s$ applied in [11] are used to optimize ω.

$\mathcal{L}_g$ measures the L_1 loss between the rendered depths $\hat{D}_i$ and the observations D_i for N_{vd} valid rays which terminate within the boundary of NISB:

$$\mathcal{L}_g = \left(\sum_{i=1}^{N_{vd}} \left|\hat{D}_i - D_i\right|\right) \Big/ N_{vd}. \tag{11}$$

$\mathcal{L}_p$ measures the L_1 loss between the rendered color $\hat{\mathbf{I}}_i$ and the observations $\mathbf{I}_i$ for valid rays:

$$\mathcal{L}_p = \left(\sum_{i=1}^{N_{vd}} \left\|\hat{\mathbf{I}}_i - \mathbf{I}_i\right\|_1\right) \Big/ N_{vd}. \tag{12}$$

The cross-entropy loss $\mathcal{L}_s$ measures the rendered semantic logits $\hat{\mathbf{S}}_i$ and the observed labels $\mathbf{S}_i$ for L classes:

$$\mathcal{L}_s = -\sum_{i=1}^{N_{vd}}\sum_{l=1}^{L} \mathbf{S}_i^l \log\left(\text{softmax}\left(\hat{\mathbf{S}}_i^l\right)\right). \tag{13}$$

Based on N_{fs} rays passing through the current NISB space and the corresponding farthest distance t_{bd} of each ray to the

current NISB's boundary, free-space loss $\mathcal{L}_{fs}$ is applied to M_{in} occupancy probability values o_{ij}, which are predicted from M_{in} samples beneath $\min\{0.8D, t_{bd}\}$ on each ray:

$$\mathcal{L}_{fs} = \left(\sum_{i=1}^{N_{fs}} \left(\left(\sum_{j=1}^{M_{in}} o_{ij} \right) \Big/ M_{in} \right) \right) \Big/ N_{fs}. \tag{14}$$

As mentioned in II-E, we take the distillation loss $\mathcal{L}_{dt}$ in overlapping areas to ensure both NISBs have similar densities. For N_{ol} rays passing through current overlapping areas, we select M_{ol} points from sampled points within the overlapping areas on each ray:

$$\mathcal{L}_{dt} = \left(\sum_{i=1}^{N_{ol}} \left(\left(\sum_{j=1}^{M_{ol}} |o_{ij}^{last} - o_{ij}^{cur}| \right) \Big/ M_{ol} \right) \right) \Big/ N_{ol}. \tag{15}$$

The optimization is achieved by minimizing the objective function $\mathcal{L}$. During current iteration e_c in e_a iterations, $\mathcal{L}$ is formed by:

$$\mathcal{L} = \begin{cases} \lambda_g \mathcal{L}_g + \lambda_p \mathcal{L}_p, & \text{if } e_c < t_p e_a \\ \lambda_g \mathcal{L}_g + \lambda_p \mathcal{L}_p + & \\ \lambda_{fs} \mathcal{L}_{fs} + \lambda_{dt} \mathcal{L}_{dt}, & \text{if } t_p e_a \le e_c < t_g e_c \\ \mathcal{L}_s + \lambda_{dt} \mathcal{L}_{dt}, & \text{otherwise.} \end{cases} \tag{16}$$

We use ADAM optimizer for the objective function $\mathcal{L}$ with multiple weighting factors $\lambda_g, \lambda_p, \lambda_{fs}, \lambda_{dt}$, and iteration thresholds t_p, t_g.

III. Experiments

A. Experiment Setups

Datasets: We perform experiments on ScanNet (V2) [25], Replica [26], and the self-collected dataset. ScanNet [25] and Replica [26] both contain ground-truth camera poses, RGB-D frames, and surface reconstructions except that ScanNet [25] also has semantic masks. To simulate the cumulative drift produced by the visual odometry on ScanNet [25] and Replica [26], the ground-truth pose of each new frame is multiplied by the cumulative drift before input, and the cumulative drift has a 20% probability of producing additional slight drift. In addition, we construct RGB-D visual odometry based on ORB-SLAM3 [13] and capture a large-scale indoor visual sequence using the RealSense d455 camera. Optionally, to generate initial semantic labels, we also pre-train a 2D semantic segmentation network based on MobileNetV3 [27].

Metrics: 3D metrics are used to evaluate the scene geometry. We follow iMAP [18] to consider Accuracy [cm], Completion [cm], and Completion Ratio [<5 cm %] for 200,000 points from both ground-truth and reconstructed mesh. Following NICE-SLAM [20], we remove unseen regions that are not inside any camera's viewing frustum. For the evaluation of pose refinement, we use ATE RMSE [28] for trajectories with drifting poses. By default, each result is the average of 5 runs.

Baselines: We compare NISB-MAP with TSDF Fusion [29], iMAP [18], and NICE-SLAM [20]. Since iMAP [18] does not release the source code, We re-implemented iMAP** based on iMAP* [20], replacing the random keyframe selection with the active sampling mentioned in iMAP [18]. As shown in Table II and Fig. 4, iMAP** has a similar reconstruction performance to iMAP [18] in terms of metrics and mesh outputs. For a fair comparison, the tracking process is removed from iMAP** and NICE-SLAM [20] while they both have the pose refinement ability in the mapping process. All methods use the same drifting trajectories for pose refinement evaluation. Semantic ability is deactivated in comparisons with baseline methods. We extract mesh with Marching Cubes algorithm [30].

TABLE I
Pose Refinement Results. ATE RMSE [Cm] ↓ is Used as the Evaluation Metric on Repilca [26] and ScanNet [25]

Method	scene0000	sene0024	office-3
Drifting trajectory	56.6	28.3	18.5
iMAP**	54.3	26.8	17.6
NICE-SLAM	54.7	27.7	15.3
Ours	9.1	12.0	4.2

TABLE II
Reconstruction Results on Replica [26]. (Average Over 6 Scenes) Memory [MB] Refers to the Size of Network Parameters, While the Size of Implicit Grids are Also Included in NICE-SLAM [20]. Acc., Comp., and Comp. Ratio are Mean Accuracy [Cm], Completion [Cm], and Completion Ratio [<5 cm %] Respectively

Method	Memory ↓	Acc. ↓	Comp. ↓	Comp. Ratio ↑
iMAP[1]	1.04	4.60	5.67	78.53
iMAP**	1.04	4.26	4.45	80.66
NICE-SLAM	59.78	2.68	3.08	88.25
ours	1.42	3.83	3.52	87.71

[1] The results are taken directly from [18].

Implementation Details: We run our method on a PC platform with an Intel i7-13700 K CPU and an NVIDIA RTX 4080 GPU. We set $N_{fm} = 3$ frame samples per iteration, $N_{pix} = 200$ pixel samples per frame, $M_{strat} = 20$ uniform samples per ray, $M_{surf} = 12$ surface samples per ray, iteration times $e_a = 12$, iteration thresholds $t_p = 0.33$, $t_g = 0.83$, weighting factors $\lambda_p = 0.4, \lambda_g = 1, \lambda_{fs} = 0.02$, and $\lambda_{dt} = 0.1$ in all experiments. Since each NISB has the same valid boundary, 3D point coordinates input to the current NISB are transformed to the corresponding coordinate frame and normalized to the $[0, 1]$ range. iMAP** and NICE-SLAM [20] adopt the same sampling and hyperparameter settings mentioned in [18] and [20] except that the number of N_{pix}, and e_a are set the same with our method. The voxel size of TSDF Fusion is set as 0.04 m.

B. Pose Refinement Evaluation

We evaluate pose refinement capabilities on both ScanNet [25] and Replica [26]. As shown in Table I and Fig. 3, even though iMAP** and NICE-SLAM [20] are able to refine pose, they are difficult to deal with cumulative drift. Our method successfully reduces cumulative drift to reconstruct the scene.

TABLE III
RECONSTRUCTION RESULTS ON SCANNET [25]. MEMORY [MB] IN TSDF FUSION REFERS TO THE SIZE OF THE VOXEL GRIDS

Method	scene0000				scene0002			
	Memory ↓	Acc. ↓	Comp. ↓	Comp. Ratio ↑	Memory ↓	Acc. ↓	Comp. ↓	Comp. Ratio ↑
TSDF-Fusion	32.62	7.17	3.60	78.16	24.06	14.79	3.56	76.38
iMAP**	1.04	11.45	5.78	59.93	1.04	8.11	4.78	64.26
NICE-SLAM	34.88	7.13	4.05	82.81	25.79	6.34	4.08	76.66
Ours	1.79	6.04	3.77	78.00	1.19	5.17	4.93	72.22
Method	scene0024				scene0046			
	Memory ↓	Acc. ↓	Comp. ↓	Comp. Ratio ↑	Memory ↓	Acc. ↓	Comp. ↓	Comp. Ratio ↑
TSDF-Fusion	16.29	11.42	3.16	87.96	19.93	12.50	2.20	91.27
iMAP**	1.04	13.08	6.29	54.05	1.04	8.28	5.60	55.43
NICE-SLAM	17.53	8.82	3.29	86.03	21.40	7.51	5.24	81.42
Ours	1.79	6.96	3.28	82.70	1.59	5.84	3.76	79.07

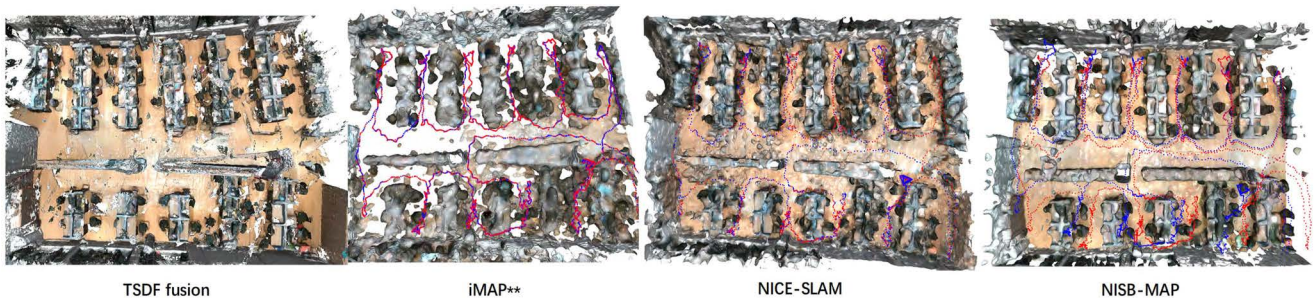

Fig. 6. Mesh outputs for large-scale scene.

C. Mapping Evaluation

Evaluation on Replica [26]: We use the same RGB-D sequences used in [18] with ground-truth trajectories. In Table II, the performance of our method is similar to that of NICE-SLAM [20]. However, NICE-SLAM [20] consumes tens of times more memory than us. In contrast to using 4 32-dimensional implicit voxel grids with different resolutions, we split the scene into several smaller MLP networks to save memory. Mesh results are shown in Fig. 4. Our method is able to reconstruct smooth results with limited iterations and predicts unobserved regions in all sequences like iMAP [18] and NICE-SLAM [20], while TSDF Fusion has difficulty filling holes in unobserved regions.

Evaluation on ScanNet [25]: Unlike RGB-D sequences rendered from multiple simulated room scenes on Replica, scenes on ScanNet [25] are more complicated and there exists depth noise in all sequences. We select several scenes from ScanNet [25] with ground-truth trajectories to evaluate the effectiveness of all baseline methods. As shown in Fig. 5, limited by the capacity of one single network, iMAP** cannot reconstruct the whole scene smoothly and forgets part of the observed region. NICE-SLAM [20] produces sharper geometry and fewer artifacts than iMAP**. Compared with NICE-SLAM [20], our method reconstructs planes more smoothly. 3D evaluation metrics are presented in Table III. Limited by its capacity, the completion ratio of iMAP** is at least 20% lower than other methods. The accuracy of our method is more precise than NICE-SLAM [20] while keeping less memory consumption. We found that a lower completion ratio than NICE-SLAM [20] is caused by artifacts generated behind the surface, which will not occur within observed viewing frustums.

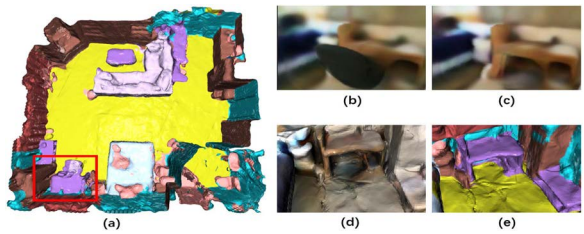

Fig. 7. Semantic mapping results on ScanNet [25]. (a) Is the semantic representation on scene0000 without object culling. (b) Is the rendered RGB image. (c) Is the rendered color image without the chair. (d) Is the mesh output with colors. (e) Is the mesh output with semantic logits. The chair is successfully culled in (c), (d), and (e).

Evaluation on self-captured dataset: To verify the scalability of our method, we capture a larger RGB-D sequence in an office. Fig. 6 shows the mesh output of each method. Due to the drift in the poses, TSDF Fusion has many errors and artifacts in the results. At the same time, iMAP** suffers from many holes and blurs caused by the limited capacity of a single network, while NICE-SLAM [20] is tough to handle cumulative drift and map the scene with errors. With stepwise optimization, our method

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

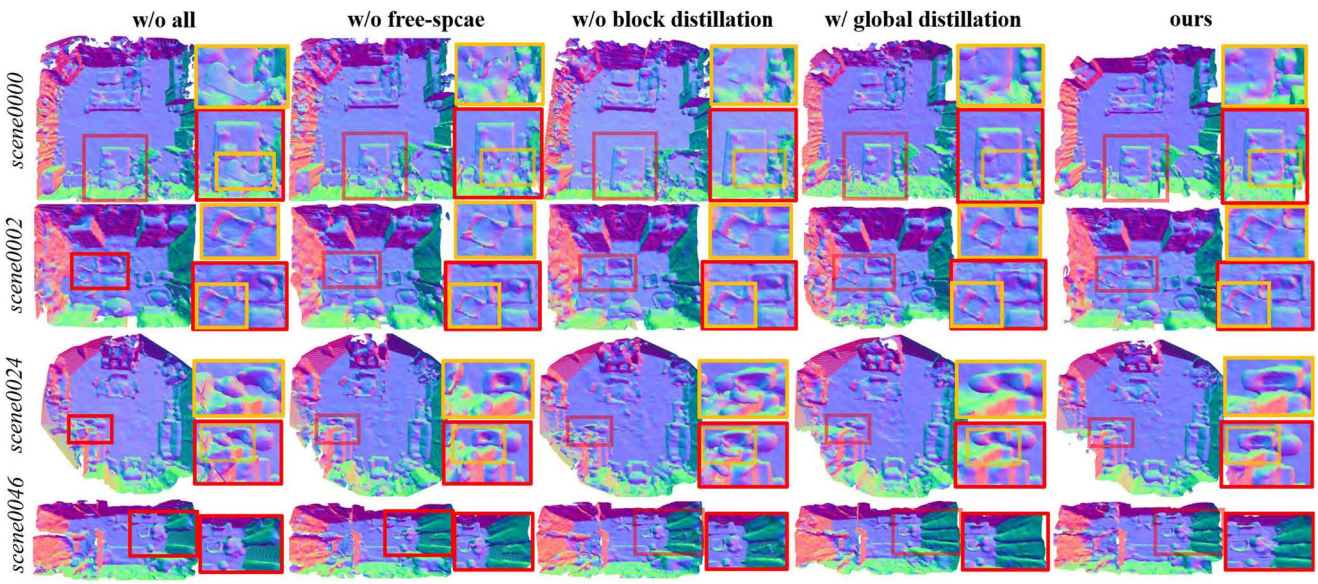

Fig. 8. Ablation study on loss functions.

TABLE IV
PERFORMENCE ANALYSIS. RUNTIME IS THE OPTIMIZATION TIME [MS] FOR EACH FRAME WHILE GPU MEM. IS THE AVERAGE GPU MEMORY [MB] USAGE FOR THE WHOLE SEQUENCE

Methods	Runtime ↓	GPU Mem.↓
iMAP**	512	4072
NICE-SLAM	1042	5850
Ours	533	2880

TABLE V
ABLATION STUDY ON STEPWISE OPTIMIZATION. ATE RMSE [CM] ↓ IS THE BEST RESULTS IN 5 RUNS

Methods	scene0000	sene0024	office-3
Ours w/o stepwise	12.6	15.6	8.4
Ours	9.1	12.0	4.2

TABLE VI
ABLATION STUDY ON LOSS FUNCTIONS. THE REPORTED RESULTS ARE THE MEAN OF 4 SCENES OF SCANNET [25]

Method	Acc. ↓	Comp. ↓	Comp. Ratio↑
Ours w/o all	7.84	4.06	72.11
Ours w/o block distillation	7.43	4.31	70.05
Ours w/o free-space	7.01	4.30	68.60
Ours w/ global distillation	6.36	4.86	70.98
Ours	6.00	3.94	74.74

still achieves stable results and reduces the cumulative drift from VO to some degree.

D. Performance Analysis.

In Table IV, we compare the runtime and average GPU memory consumption for optimizing each frame with the same number of pixel samples and iterations during optimization. NICE-SLAM [20] requires more GPU memory and runtime to update the network embedded with hierarchical 32-channel implicit voxel grids. Accompanied by Tiny-cuda-nn [23] and optimizing multiple smaller MLP networks than one single large network for each frame, our method keeps pace with iMAP on runtime while using half of the GPU memory than NICE-SLAM [20].

Semantic labeling and object culling: Optionally, with semantic labels as input, our method is able to learn the semantic information of the scene simultaneously. In addition, specific objects with semantic labels such as the chair marked in Fig. 7(a), can be culled entirely.

E. Ablation Studies.

Stepwise optimization: As shown in Table V for the ATE RMSE results, stepwise optimization further reduces the camera drift. It indicates that refining poses individually before optimizing the geometry is more beneficial to reduce the cumulative drift.

Distillation and free-space loss: Since the rendered depth and color loss are necessary for the geometry and texture reconstruction, we only compare our method without the block distillation and free-space loss. In addition, a global NISB with network $width = 256$ for distillation is also evaluated, which needs about 200 ms additionally to process each frame than ours that utilize the last trained NISB. Fig. 8 shows that sampling sparsely without block distillation and free-space loss results in boundary artifacts and geometric inconsistency (the same plane has different density levels in two adjacent blocks). Implementing knowledge distillation can effectively keep both densities at the same level while the free-space loss can reduce the artifacts in viewing frustum. Quantitative results in Table VI show that

TABLE VII
ABLATION STUDY ON HYPERPARAMETERS. DISTANCE [M] IS THE CENTER DISTANCE BETWEEN ANY TWO ADJACENT NISBS WHILE WIDTH THE NETWORK WIDTH FOR HIDDEN LAYERS

Width	64		128	
Distance	Comp. Ratio ↑	Runtime ↓	Comp. Ratio ↑	Runtime ↓
3	81.17	653	79.38	805
4	82.47	450	86.28	533
5	79.39	468	80.84	448
6	76.08	390	80.12	400

our method is more accurate than ours with a global NISB for distillation, which we think is caused by the catastrophic forgetting problem that occurred in the global NISB with limited capacity.

Hyperparameters settings: We analyze the effect of parameter design on performance using the same Replica [26] scene: office-2. Network $width = 128$ and center distance $a = 4$ m offer the best trade-off on quality and performance as shown in Table VII.

IV. CONCLUSION

We investigate the scalability of neural implicit representations for mapping in large-scale indoor environments. By splitting the scene into multiple neural implicit spatial blocks, catastrophic forgetting caused by learning the whole scene with a limited capacity network is solved. Additionally, a 2 Hz update speed and a GPU memory footprint of fewer than 3 GB are guaranteed in the mapping process. Finally, NISB-MAP can reconstruct the large-scale scene in a compact and continuous way. Currently, NISB-MAP is still a bit slow for online optimization. In the next stage, we will leverage the parallel computing capacity of CUDA to boost computational efficiency and enable deployment on onboard computers, such as Nvidia Xavier NX. Another direction is to extend the method to outdoor environments and explore the collaborative mapping and optimization of multiple robots in unstructured environments.

REFERENCES

[1] J. Ye, Y. Chen, N. Wang, and X. Wang, "Online adaptation for implicit object tracking and shape reconstruction in the wild," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 8909–8916, Oct. 2022.

[2] M. Adamkiewicz et al., "Vision-only robot navigation in a neural radiance world," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4606–4613, Apr. 2022.

[3] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4460–4470.

[4] J. Sun, Y. Xie, L. Chen, X. Zhou, and H. Bao, "NeuralRecon: Real-time coherent 3D reconstruction from monocular video," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 15598–15607.

[5] Z. Murez, T. Van As, J. Bartolozzi, A. Sinha, V. Badrinarayanan, and A. Rabinovich, "Atlas: End-to-end 3D scene reconstruction from posed images," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 414–431.

[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," *Commun. ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[7] 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 *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 5855–5864.

[8] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "NeRF in the wild: Neural radiance fields for unconstrained photo collections," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 7210–7219.

[9] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, "Depth-supervised NeRF: Fewer views and faster training for free," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 12882–12891.

[10] Y. Wei, S. Liu, Y. Rao, W. Zhao, J. Lu, and J. Zhou, "NerfingMVS: Guided optimization of neural radiance fields for indoor multi-view stereo," in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 5610–5619.

[11] S. Zhi, T. Laidlow, S. Leutenegger, and A. J. Davison, "In-place scene labelling and understanding with implicit scene representation," in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 15838–15847.

[12] S. Zhi, E. Sucar, A. Mouton, I. Haughton, T. Laidlow, and A. J. Davison, "ILabel: Interactive neural scene labelling," *IEEE Robot. Automat. Lett.*, 2022.

[13] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1874–1890, Dec. 2021.

[14] C. Reiser, S. Peng, Y. Liao, and A. Geiger, "KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs," in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 14335–14345.

[15] X. Wu et al., "Scalable neural indoor scene rendering," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 1–16, 2022.

[16] J. N. Martel, D. B. Lindell, C. Z. Lin, E. R. Chan, M. Monteiro, and G. Wetzstein, "ACORN: Adaptive coordinate networks for neural scene representation," *ACM Trans. Graph.*, vol. 40, pp. 1–13, 2021.

[17] M. Tancik et al., "Block-NeRF: Scalable large scene neural view synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 8248–8258.

[18] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "iMAP: Implicit mapping and positioning in real-time," in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 6229–6238.

[19] J. Ortiz et al., "iSDF: Real-time neural signed distance fields for robot perception," *Robot.: Sci. Syst.*, 2022.

[20] Z. Zhu et al., "NICE-SLAM: Neural implicit scalable encoding for SLAM," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 12786–12796.

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

[22] J. T. Barron, B. Mildenhall, D. Verbin, P.P. Srinivasan, and P. Hedman, "Mip-NeRF 360: Unbounded anti-aliased neural radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 5470–5479.

[23] T. Müller, "Tiny cuda neural network framework," 2021. [Online]. Available: https://github.com/NVlabs/tiny-cuda-nn

[24] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. 3rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[25] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5828–5839.

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

[27] A. Howard et al., "Searching for MobileNetV3," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 1314–1324.

[28] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Rob. Syst.*, 2012, pp. 573–580.

[29] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. SIGGRAPH*, 1996, pp. 303–312.

[30] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987.