

NICE-SLAM: Neural Implicit Scalable Encoding for SLAM

Zihan Zhu^{1,2*}

Songyou Peng^{2,4*}

Viktor Larsson³

Weiwei Xu¹

Hujun Bao¹

Zhaopeng Cui^{1†}

Martin R. Oswald^{2,5}

Marc Pollefeys^{2,6}

¹State Key Lab of CAD&CG, Zhejiang University

⁴MPI for Intelligent Systems, Tübingen

²ETH Zurich

³Lund University

⁵University of Amsterdam

⁶Microsoft

Abstract

Neural implicit representations have recently shown encouraging results in various domains, including promising progress in simultaneous localization and mapping (SLAM). Nevertheless, existing methods produce over-smoothed scene reconstructions and have difficulty scaling up to large scenes. These limitations are mainly due to their simple fully-connected network architecture that does not incorporate local information in the observations. In this paper, we present NICE-SLAM, a dense SLAM system that incorporates multi-level local information by introducing a hierarchical scene representation. Optimizing this representation with pre-trained geometric priors enables detailed reconstruction on large indoor scenes. Compared to recent neural implicit SLAM systems, our approach is more scalable, efficient, and robust. Experiments on five challenging datasets demonstrate competitive results of NICE-SLAM in both mapping and tracking quality. Project page: <https://pengsongyou.github.io/nice-slam>.

1. Introduction

Dense visual Simultaneous Localization and Mapping (SLAM) is a fundamental problem in 3D computer vision with many applications in autonomous driving, indoor robotics, mixed reality, etc. In order to make a SLAM system truly useful for real-world applications, the following properties are essential. First, we desire the SLAM system to be real-time. Next, the system should have the ability to make reasonable predictions for regions without observations. Moreover, the system should be able to scale up to large scenes. Last but not least, it is crucial to be robust to noisy or missing observations.

In the scope of real-time dense visual SLAM system, many methods have been introduced for RGB-D cameras in the past years. Traditional dense visual SLAM systems [30,

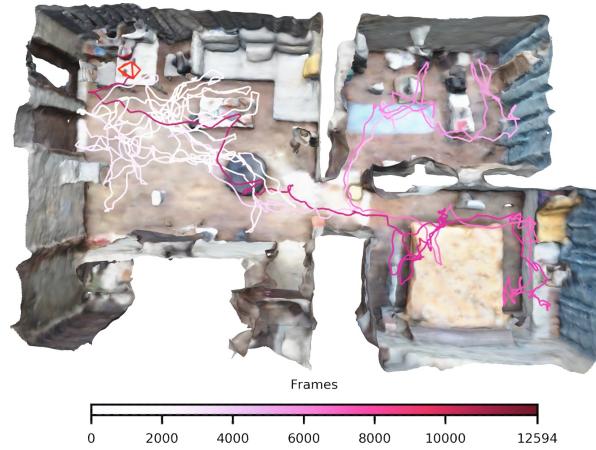


Figure 1. **Multi-room Apartment 3D Reconstruction using NICE-SLAM.** A hierarchical feature grid jointly encodes geometry and color information and is used for both mapping and tracking. We depict the final mesh and camera tracking trajectory.

[42, 59, 60] fulfil the real-time requirement and can be used in large-scale scenes, but they are unable to make plausible geometry estimation for unobserved regions. On the other hand, learning-based SLAM approaches [3, 12, 48, 68] attain a certain level of predictive power since they typically train on task-specific datasets. Moreover, learning-based methods tend to better deal with noises and outliers. However, these methods are typically only working in small scenes with multiple objects. Recently, Sucar et al. [47] applied a neural implicit representation in the real-time dense SLAM system (called iMAP), and they showed decent tracking and mapping results for room-sized datasets. Nevertheless, when scaling up to larger scenes, e.g., an apartment consisting of multiple rooms, significant performance drops are observed in both the dense reconstruction and camera tracking accuracy.

The key limiting factor of iMAP [47] stems from its use of a single multi-layer perceptron (MLP) to represent the entire scene, which can only be updated globally with every new, potentially partial RGB-D observations. In contrast, recent works [38, 49] demonstrate that establishing multi-

*Equal contribution.

†Corresponding author.

level grid-based features can help to preserve geometric details and enable reconstructing complex scenes, but these are offline methods without real-time capability.

In this work, we seek to combine the strengths of hierarchical scene representations with those of neural implicit representations for the task of dense RGB-D SLAM. To this end, we introduce NICE-SLAM, a dense RGB-D SLAM system that can be applied to large-scale scenes while preserving the predictive ability. Our key idea is to represent the scene geometry and appearance with hierarchical feature grids and incorporate the inductive biases of neural implicit decoders pretrained at different spatial resolutions. With the rendered depth and color images from the occupancy and color decoder outputs, we can optimize the features grids only within the viewing frustum by minimizing the re-rendering losses. We perform extensive evaluations on a wide variety of indoor RGB-D sequences and demonstrate the scalability and predictive ability of our method. Overall, we make the following contributions:

- We present NICE-SLAM, a dense RGB-D SLAM system that is real-time capable, scalable, predictive, and robust to various challenging scenarios.
- The core of NICE-SLAM is a hierarchical, grid-based neural implicit encoding. In contrast to global neural scene encodings, this representation allows for local updates, which is a prerequisite for large-scale approaches.
- We conduct extensive evaluations on various datasets which demonstrate competitive performance in both mapping and tracking.

The code is available at <https://github.com/cvg/nice-slam>.

2. Related Work

Dense Visual SLAM. Most modern methods for visual SLAM follow the overall architecture introduced in the seminal work by Klein et al. [19], decomposing the task into mapping and tracking. The map representations can be generally divided into two categories: view-centric and world-centric. The first anchors 3D geometry to specific keyframes, often represented as depth maps in the dense setting. One of the early examples of this category was DTAM [30]. Because of its simplicity, DTAM has been widely adapted in many recent learning-based SLAM systems. For example, [55, 69] regress both depth and pose updates. DeepV2D [52] similarly alternates between regressing depth and pose estimation but uses test-time optimization. BA-Net [51] and DeepFactors [12] simplify the optimization problem by using a set of basis depth maps. There are also some methods, e.g., CodeSLAM [3], SceneCode [68] and NodeSLAM [48], which optimize a latent representation that decodes into the keyframe or object depth maps. DROID-SLAM [53] uses regressed optical flow to define geometrical residuals for its refinement.

TANDEM [20] combines multi-view stereo with DSO [15] for a real-time dense SLAM system. On the other hand, the world-centric map representation anchors the 3D geometry in uniform world coordinates, and can be further divided into surfels [43, 59] and voxel grids, typically storing occupancies or TSDF values [11]. Voxel grids have been used extensively in RGB-D SLAM, e.g., KinectFusion [29] among other works [5, 14, 18, 34]. In our proposed pipeline we also adopt the voxel-grid representation. In contrast to previous SLAM approaches, we store implicit latent codes of the geometry and directly optimize them during mapping. This richer representation allows us to achieve more accurate geometry at lower grid resolutions.

Neural Implicit Representations. Recently, neural implicit representations demonstrated promising results for object geometry representation [8, 22, 25, 33, 35–37, 41, 56, 61, 64, 65], scene completion [6, 17, 38], novel view synthesis [24, 26, 39, 67] and also generative modelling [7, 31, 32, 44]. A few recent papers [1, 4, 9, 28, 49, 58, 62] attempt to predict scene-level geometry with RGB-(D) inputs, but they all assume given camera poses. Another set of works [21, 57, 66] tackle the problem of camera pose optimization, but they need a rather long optimization process, which is not suitable for real-time applications.

The most related work to our method is iMAP [47]. Given an RGB-D sequence, they introduce a real-time dense SLAM system that uses a single multi-layer perceptron (MLP) to compactly represent the entire scene. Nevertheless, due to the limited model capacity of a single MLP, iMAP fails to produce detailed scene geometry and accurate camera tracking, especially for larger scenes. In contrast, we provide a scalable solution akin to iMAP, that combines learnable latent embeddings with a pretrained continuous implicit decoder. In this way, our method can reconstruct complex geometry and predict detailed textures for larger indoor scenes, while maintaining much less computation and faster convergence. Notably, the works [17, 38] also combine traditional grid structures with learned feature representations for scalability, but neither of them is real-time capable. Moreover, DI-Fusion [16] also optimizes a feature grid given an RGB-D sequence, but their reconstruction often contain holes and their camera tracking is not robust for the pure surface rendering loss.

3. Method

We provide an overview of our method in Fig. 2. We represent the scene geometry and appearance using four feature grids and their corresponding decoders (Sec. 3.1). We trace the viewing rays for every pixel using the estimated camera calibration. By sampling points along a viewing ray and querying the network, we can render both depth and color values of this ray (Sec. 3.2). By minimizing the re-rendering losses for depth and color, we are able to optimize both the

NICE-SLAM

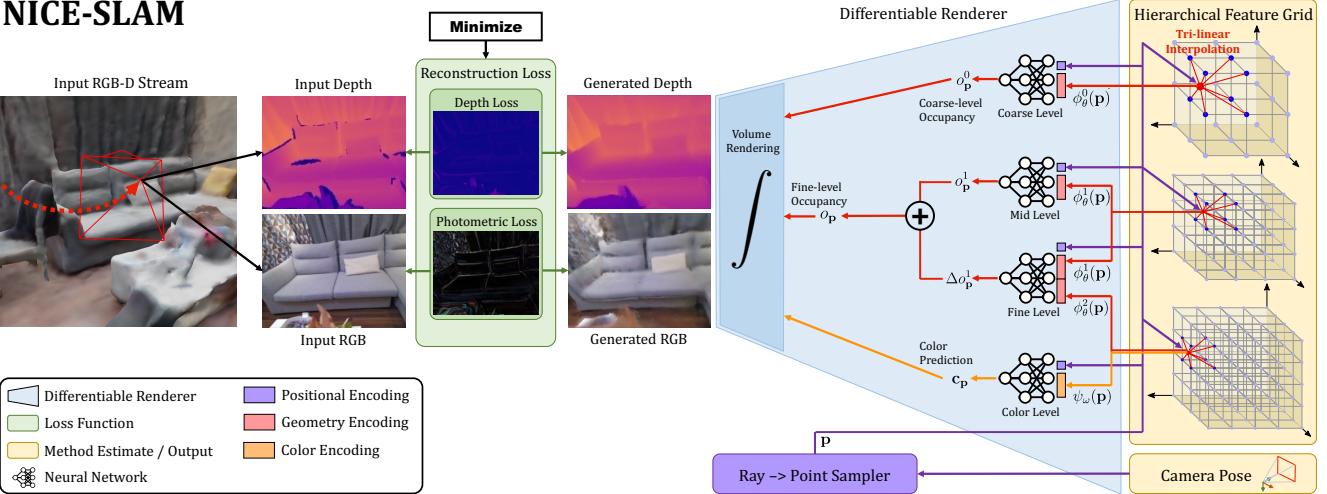


Figure 2. System Overview. Our method takes an RGB-D image stream as input and outputs both the camera pose as well as a learned scene representation in form of a hierarchical feature grid. From *right-to-left*, our pipeline can be interpreted as a generative model which renders depth and color images from a given scene representation and camera pose. At test time we estimate both the scene representation and camera pose by solving the inverse problem via backpropagating the image and depth reconstruction loss through a differentiable renderer (*left-to-right*). Both entities are estimated within an alternating optimization: **Mapping:** The backpropagation only updates the hierarchical scene representation; **Tracking:** The backpropagation only updates the camera pose. For better readability we joined the fine-scale grid for geometry encoding with the equally-sized color grid and show them as one grid with two attributes (red and orange).

camera pose and the scene geometry in an alternating fashion (Sec. 3.3) for selected keyframes (Sec. 3.4).

3.1. Hierarchical Scene Representation

We now introduce our hierarchical scene representation that combines multi-level grid features with pre-trained decoders for occupancy predictions. The geometry is encoded into three feature grids ϕ_θ^l and their corresponding MLP decoders f^l , where $l \in \{0, 1, 2\}$ is referred to coarse, mid and fine-level scene details. In addition, we also have a single feature grid ψ_ω and decoder g_ω to model the scene appearance. Here θ and ω indicate the optimizable parameters for geometry and color, i.e., the features in the grid and the weights in the color decoder.

Mid-&Fine-level Geometric Representation. The observed scene geometry is represented in the mid- and fine-level feature grids. In the reconstruction process we use these two grids in a coarse-to-fine approach where the geometry is first reconstructed by optimizing the mid-level feature grid, followed by a refinement using the fine-level. In the implementation we use voxel grids with side lengths of 32cm and 16cm respectively, except for TUM RGB-D [46] we use 16cm and 8cm. For the mid-level, the features are directly decoded into occupancy values using the associated MLP f^1 . For any point $\mathbf{p} \in \mathbb{R}^3$, we get the occupancy as

$$o_{\mathbf{p}}^1 = f^1(\mathbf{p}, \phi_\theta^1(\mathbf{p})), \quad (1)$$

where $\phi_\theta^1(\mathbf{p})$ denotes that the feature grid is tri-linearly interpolated at the point \mathbf{p} . The relatively low-resolution al-

low us to efficiently optimize the grid features to fit the observations. To capture smaller high-frequency details in the scene geometry we add in the fine-level features in a residual manner. In particular, the fine-level feature decoder takes as input both the corresponding mid-level feature and the fine-level feature and outputs an offset from the mid-level occupancy, i.e.,

$$\Delta o_{\mathbf{p}}^1 = f^2(\mathbf{p}, \phi_\theta^1(\mathbf{p}), \phi_\theta^2(\mathbf{p})), \quad (2)$$

where the final occupancy for a point is given by

$$o_{\mathbf{p}} = o_{\mathbf{p}}^1 + \Delta o_{\mathbf{p}}^1. \quad (3)$$

Note that we fix the pre-trained decoders f^1 and f^2 , and only optimize the feature grids ϕ_θ^1 and ϕ_θ^2 throughout the entire optimization process. We demonstrate that this helps to stabilize the optimization and learn consistent geometry.

Coarse-level Geometric Representation. The coarse-level feature grid aims to capture the high-level geometry of the scene (e.g., walls, floor, etc), and is optimized independently from the mid- and fine-level. The goal of the coarse-grid is to be able to predict approximate occupancy values outside of the observed geometry (which is encoded in the mid/fine-levels), even when each coarse voxel has only been partially observed. For this reason we use a very low resolution, with a side-length of 2m in the implementation. Similarly to the mid-level grid, we decode directly into occupancy values by interpolating the features and passing through the MLP f^0 , i.e.,

$$o_{\mathbf{p}}^0 = f^0(\mathbf{p}, \phi_\theta^0(\mathbf{p})). \quad (4)$$

During tracking, the coarse-level occupancy values are only used for predicting previously unobserved scene parts. This *forecasted* geometry allows us to track even when a large portion of the current image is previously unseen.

Pre-training Feature Decoders. In our framework we use three different fixed MLPs to decode the grid features into occupancy values. The coarse and mid-level decoders are pre-trained as part of ConvONet [38] which consists of a CNN encoder and an MLP decoder. We train both the encoder/decoder using the binary cross-entropy loss between the predicted and the ground-truth value, same as in [38]. After training, we only use the decoder MLP, as we will directly optimize the features to fit the observations in our reconstruction pipeline. In this way the pre-trained decoder can leverage resolution-specific priors learned from the training set, when decoding our optimized features.

The same strategy is used to pre-train the fine-level decoder, except that we simply concatenate the feature $\phi_\theta^1(\mathbf{p})$ from the mid-level together with the fine-level feature $\phi_\theta^2(\mathbf{p})$ before inputting to the decoder.

Color Representation. While we are mainly interested in the scene geometry, we also encode the color information allowing us to render RGB images which provides additional signals for tracking. To encode the color in the scene, we apply another feature grid ψ_ω and decoder \mathbf{g}_ω :

$$\mathbf{c}_\mathbf{p} = \mathbf{g}_\omega(\mathbf{p}, \psi_\omega(\mathbf{p})), \quad (5)$$

where ω indicates learnable parameters during optimization. Different from the geometry that has strong prior knowledge, we empirically found that jointly optimizing the color features ψ_ω and decoder \mathbf{g}_ω improves the tracking performance (c.f. Table 5). Note that, similarly to iMAP [47], this can lead to forgetting problems and the color is only consistent locally. If we want to visualize the color for the entire scene, it can be optimized globally as a post-processing step.

Network Design. For all MLP decoders, we use a hidden feature dimension of 32 and 5 fully-connected blocks. Except for the coarse-level geometric representation, we apply a learnable Gaussian positional encoding [47, 50] to \mathbf{p} before serving as input to MLP decoders. We observe this allows discovery of high frequency details for both geometry and appearance.

3.2. Depth and Color Rendering

Inspired by the recent success of volume rendering in NeRF [26], we propose to also use a differentiable rendering process which integrates the predicted occupancy and colors from our scene representation in Section 3.1.

Given camera intrinsic parameters and current camera pose, we can calculate the viewing direction \mathbf{r} of a pixel coordinate. We first sample along this ray N_{strat} points for stratified sampling, and also uniformly sample N_{imp} points

near to the depth¹. In total we sample $N = N_{\text{strat}} + N_{\text{imp}}$ points for each ray. More formally, let $\mathbf{p}_i = \mathbf{o} + d_i \mathbf{r}, i \in \{1, \dots, N\}$ denote the sampling points on the ray \mathbf{r} given the camera origin \mathbf{o} , and d_i corresponds to the depth value of \mathbf{p}_i along this ray. For every point \mathbf{p}_i , we can calculate their coarse-level occupancy probability $o_{\mathbf{p}_i}^0$, fine-level occupancy probability $o_{\mathbf{p}_i}$, and color value $\mathbf{c}_{\mathbf{p}_i}$ using Eq. (4), Eq. (3), and Eq. (5). Similar to [35], we model the ray termination probability at point \mathbf{p}_i as $w_i^c = o_{\mathbf{p}_i}^0 \prod_{j=1}^{i-1} (1 - o_{\mathbf{p}_j}^0)$ for coarse level, and $w_i^f = o_{\mathbf{p}_i} \prod_{j=1}^{i-1} (1 - o_{\mathbf{p}_j})$ for fine level.

Finally for each ray, the depth at both coarse and fine level, and color can be rendered as:

$$\hat{D}^c = \sum_{i=1}^N w_i^c d_i, \quad \hat{D}^f = \sum_{i=1}^N w_i^f d_i, \quad \hat{I} = \sum_{i=1}^N w_i^f \mathbf{c}_i. \quad (6)$$

Moreover, we also calculate depth variances along the ray:

$$\hat{D}_{var}^c = \sum_{i=1}^N w_i^c (\hat{D}^c - d_i)^2 \quad \hat{D}_{var}^f = \sum_{i=1}^N w_i^f (\hat{D}^f - d_i)^2. \quad (7)$$

3.3. Mapping and Tracking

In this section, we provide details on the optimization of the scene geometry θ and appearance ω parameters of our hierarchical scene representation, and of the camera poses.

Mapping. To optimize the scene representation mentioned in Section 3.1, we uniformly sample total M pixels from the current frame and the selected keyframes. Next, we perform optimization in a staged fashion to minimize the geometric and photometric losses.

The geometric loss is simply an L_1 loss between the observations and predicted depths at coarse or fine level:

$$\mathcal{L}_g^l = \frac{1}{M} \sum_{m=1}^M |D_m - \hat{D}_m^l|, \quad l \in \{c, f\}. \quad (8)$$

The photometric loss is also an L_1 loss between the rendered and observed color values for M sampled pixel:

$$\mathcal{L}_p = \frac{1}{M} \sum_{m=1}^M |I_m - \hat{I}_m|. \quad (9)$$

At the first stage, we optimize only the mid-level feature grid ϕ_θ^1 using the geometric loss \mathcal{L}_g^f in Eq. (8). Next, we jointly optimize both the mid and fine-level $\phi_\theta^1, \phi_\theta^2$ features with the same fine-level depth loss \mathcal{L}_g^f . Finally, we conduct a local bundle adjustment (BA) to jointly optimize feature grids at all levels, the color decoder, as well as the camera extrinsic parameters $\{\mathbf{R}_i, \mathbf{t}_i\}$ of K selected keyframes:

$$\min_{\theta, \omega, \{\mathbf{R}_i, \mathbf{t}_i\}} (\mathcal{L}_g^c + \mathcal{L}_g^f + \lambda_p \mathcal{L}_p), \quad (10)$$

¹We empirically define the sampling interval as $\pm 0.05D$, where D is the depth value of the current ray.

where λ_p is the loss weighting factor.

This multi-stage optimization scheme leads to better convergence as the higher-resolution appearance and fine-level features can rely on the already refined geometry coming from mid-level feature grid.

Note that we parallelize our system in three threads to speed up the optimization process: one thread for coarse-level mapping, one for mid-&fine-level geometric and color optimization, and another one for camera tracking.

Camera Tracking. In addition to optimizing the scene representation, we also run in parallel camera tracking to optimize the camera poses of the current frame, i.e., rotation and translation $\{\mathbf{R}, \mathbf{t}\}$. To this end, we sample M_t pixels in the current frame and apply the same photometric loss in Eq. (9) but use a modified geometric loss:

$$\mathcal{L}_{g_var} = \frac{1}{M_t} \sum_{m=1}^{M_t} \frac{|D_m - \hat{D}_m^c|}{\sqrt{\hat{D}_{var}^c}} + \frac{|D_m - \hat{D}_m^f|}{\sqrt{\hat{D}_{var}^f}}. \quad (11)$$

The modified loss down-weights less certain regions in the reconstructed geometry [47, 63], e.g., object edges. The camera tracking is finally formulated as the following minimization problem:

$$\min_{\mathbf{R}, \mathbf{t}} (\mathcal{L}_{g_var} + \lambda_{pt} \mathcal{L}_p). \quad (12)$$

The coarse feature grid is able to perform short-range predictions of the scene geometry. This extrapolated geometry provides a meaningful signal for the tracking as the camera moves into previously unobserved areas. Making it more robust to sudden frame loss or fast camera movement. We provide experiments in the supplementary material.

Robustness to Dynamic Objects. To make the optimization more robust to dynamic objects during tracking, we filter pixels with large depth/color re-rendering loss. In particular, we remove any pixel from the optimization where the loss Eq. (12) is larger than $10 \times$ the median loss value of all pixels in the current frame. Fig. 6 shows an example where a dynamic object is ignored since it is not present in the rendered RGB and depth image. Note that for this task, we only optimize the scene representation during the mapping. Jointly optimizing camera parameters and scene representations under dynamic environments is non-trivial, and we consider it as an interesting future direction.

3.4. Keyframe Selection

Similar to other SLAM systems, we continuously optimize our hierarchical scene representation with a set of selected keyframes. We maintain a global keyframe list in the same spirit of iMAP [47], where we incrementally add new keyframes based on the information gain. However, in contrast to iMAP [47], we only include keyframes which

have visual overlap with the current frame when optimizing the scene geometry. This is possible since we are able to make local updates to our grid-based representation, and we do not suffer from the same forgetting problems as [47]. This keyframe selection strategy not only ensures the geometry outside of the current view remains static, but also results in a very efficient optimization problem as we only optimize the necessary parameters each time. In practice, we first randomly sample pixels and back-project the corresponding depths using the optimized camera pose. Then, we project the point cloud to every keyframe in the global keyframe list. From those keyframes that have points projected onto, we randomly select $K - 2$ frames. In addition, we also include the most recent keyframe and the current frame in the scene representation optimization, forming a total number of K active frames. Please refer to Section 4.4 for an ablation study on the keyframe selection strategy.

4. Experiments

We evaluate our SLAM framework on a wide variety of datasets, both real and synthetic, of varying size and complexity. We also conduct a comprehensive ablation study that supports our design choices.

4.1. Experimental Setup

Datasets. We consider 5 versatile datasets: Replica [45], ScanNet [13], TUM RGB-D dataset [46], Co-Fusion dataset [40], as well as a self-captured large apartment with multiple rooms. We follow the same pre-processing step for TUM RGB-D as in [54].

Baselines. We compare to TSDF-Fusion [11] with our camera poses with a voxel grid resolution of 256^3 (results of higher resolutions are reported in the supp. material), DI-Fusion [16] using their official implementation², as well as our faithful iMAP [47] re-implementation: iMAP*. Our re-implementation has similar performance as the original iMAP in both scene reconstruction and camera tracking.

Metrics. We use both 2D and 3D metrics to evaluate the scene geometry. For the 2D metric, we evaluate the L1 loss on 1000 randomly-sampled depth maps from both reconstructed and ground truth meshes. For fair comparison, we apply the bilateral solver [2] to DI-Fusion [16] and TSDF-Fusion to fill depth holes before calculating the average L1 loss. For 3D metrics, we follow [47] and consider *Accuracy* [cm], *Completion* [cm], and *Completion Ratio* [$< 5\text{cm}\%$], except that we remove unseen regions that are not inside any camera's viewing frustum. As for the evaluation of camera tracking, we use ATE RMSE [46]. If not specified otherwise, by default we report the average results of 5 runs.

Implementation Details. We run our SLAM system on a desktop PC with a 3.80GHz Intel i7-10700K CPU and

²<https://github.com/huangjh-pub/di-fusion>



Figure 3. **Reconstruction Results on the Replica Dataset [45]**. iMAP* refers to our iMAP re-implementation.

	TSDF-Fusion [11]	iMAP* [47]	DI-Fusion [16]	NICE-SLAM
Mem. (MB) ↓	67.10	1.04	3.78	12.02
Depth L1 ↓	7.57	7.64	23.33	3.53
Acc. ↓	1.60	6.95	19.40	2.85
Comp. ↓	3.49	5.33	10.19	3.00
Comp. Ratio ↑	86.08	66.60	72.96	89.33

Table 1. **Reconstruction Results for the Replica Dataset [45] (average over 8 scenes)**. iMAP* indicates our re-implementation of iMAP. TSDF-Fusion uses camera poses from NICE-SLAM. Detailed metrics for each scene can be found in the supp. material.

an NVIDIA RTX 3090 GPU. In all our experiments, we use the number of sampling points on a ray $N_{\text{strat}} = 32$ and $N_{\text{imp}} = 16$, photometric loss weighting $\lambda_p = 0.2$ and $\lambda_{pt} = 0.5$. For small-scale synthetic datasets (Replica and Co-Fusion), we select $K = 5$ keyframes and sample $M = 1000$ and $M_t = 200$ pixels. For large-scale real datasets (ScanNet and our self-captured scene), we use $K = 10$, $M = 5000$, $M_t = 1000$. As for the challenging TUM RGB-D dataset, we use $K = 10$, $M = 5000$, $M_t = 5000$. For our re-implementation iMAP*, we follow all the hyperparameters mentioned in [47] except that we set the number of sampling pixels to 5000 since it leads to better performance in both reconstruction and tracking.

4.2. Evaluation of Mapping and Tracking

Evaluation on Replica [45]. To evaluate on Replica [45], we use the same rendered RGB-D sequence provided by the authors of iMAP. With the hierarchical scene representation, our method is able to reconstruct the geometry precisely within limited iterations. As shown in Table 1, NICE-SLAM significantly outperforms baseline methods on almost all metrics, while keeping a reasonable memory consumption. Qualitatively, we can see from Fig. 3 that our method produces sharper geometry and less artifacts.

Evaluation on TUM RGB-D [46]. We also evaluate the camera tracking performance on the small-scale TUM RGB-D dataset. As shown in Table 2, our method outperforms iMAP and DI-Fusion even though ours is by design more suitable for large scenes. As can be noticed, the state-

	fr1/desk	fr2/xyz	fr3/office
iMAP [47]	4.9	2.0	5.8
iMAP* [47]	7.2	2.1	9.0
DI-Fusion [16]	4.4	2.3	15.6
NICE-SLAM	2.7	1.8	3.0
BAD-SLAM [43]	1.7	1.1	1.7
Kintinuous [60]	3.7	2.9	3.0
ORB-SLAM2 [27]	1.6	0.4	1.0

Table 2. **Camera Tracking Results on TUM RGB-D [46]**. ATE RMSE [cm] (↓) is used as the evaluation metric. NICE-SLAM reduces the gap between SLAM methods with neural implicit representations and traditional approaches. We report the best out of 5 runs for all methods in this table. The numbers for iMAP, BAD-SLAM, Kintinuous, and ORB-SLAM2 are taken from [47].

Scene ID	0000	0059	0106	0169	0181	0207	Avg.
iMAP* [47]	55.95	32.06	17.50	70.51	32.10	11.91	36.67
DI-Fusion [16]	62.99	128.00	18.50	75.80	87.88	100.19	78.89
NICE-SLAM	8.64	12.25	8.09	10.28	12.93	5.59	9.63

Table 3. **Camera Tracking Results on ScanNet [13]**. Our approach yields consistently better results on this dataset. ATE RMSE (↓) is used as the evaluation metric.

of-the-art approaches for tracking (e.g. BAD-SLAM [43], ORB-SLAM2 [27]) still outperform the methods based on implicit scene representations (iMAP [47] and ours). Nevertheless, our method significantly reduces the gap between these two categories, while retaining the representational advantages of implicit representations.

Evaluation on ScanNet [13]. We select multiple large scenes from ScanNet [13] to benchmark the scalability of different methods. For the geometry shown in Fig. 4, we can clearly notice that NICE-SLAM produces sharper and more detailed geometry over TSDF-Fusion, DI-Fusion and iMAP*. In terms of tracking, as can be observed, iMAP* and DI-Fusion either completely fails or introduces large drifting, while our method successfully reconstructs the entire scene. Quantitatively speaking, our tracking results are also significantly more accurate than both DI-Fusion and iMAP* as shown in Table 3.

Evaluation on a Larger Scene. To evaluate the scalability of our method we captured a sequence in a large apartment with multiple rooms. Fig. 1 and Fig. 5 show the reconstructions obtained using NICE-SLAM, DI-Fusion [16] and iMAP* [47]. For reference we also show the 3D reconstruction using the offline tool Redwood [10] in Open3D [70]. We can see that NICE-SLAM has comparable results with the offline method, while iMAP* and DI-Fusion fails to reconstruct the full sequence.

4.3. Performance Analysis

Besides the evaluation on scene reconstruction and camera tracking on various datasets, in the following we also evaluate other characteristics of the proposed pipeline.

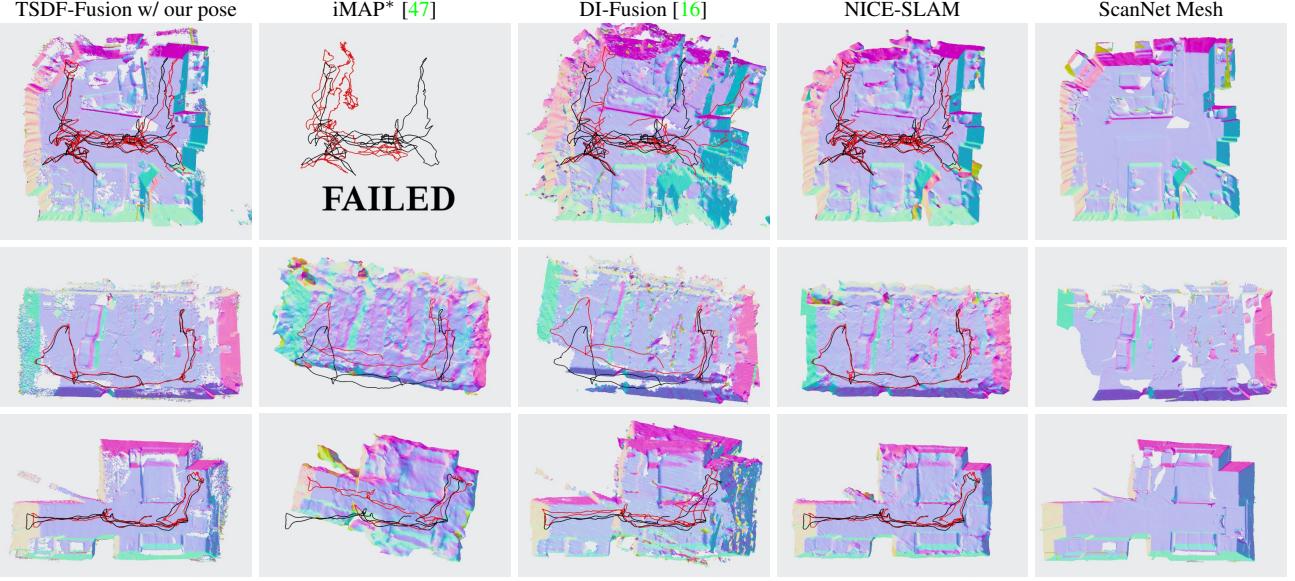


Figure 4. **3D Reconstruction and Tracking on ScanNet [13]**. The black trajectory is from ScanNet [13], the red trajectory is the methods’ tracking result. We tried various hyperparameters for iMAP* and present the best results which are mostly inferior.

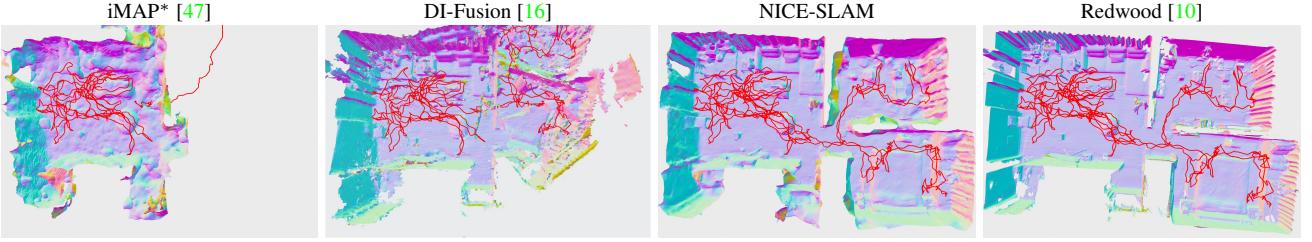


Figure 5. **3D Reconstruction and Tracking on a Multi-room Apartment**. The camera tracking trajectory is shown in red. iMAP* and DI-Fusion failed to reconstruct the entire sequence. We also show the result of an offline method [10] for reference.

Computation Complexity. First, we compare the number of floating point operations (FLOPs) needed for querying color and occupancy/volume density of one 3D point, see Table 4. Our method requires only 1/4 FLOPs of iMAP. It is worth mentioning that FLOPs in our approach remain the same even for very large scenes. In contrast, due to the use of a single MLP in iMAP, the capacity limit of the MLP might require more parameters that result in more FLOPs.

Runtime. We also compare in Table 4 the runtime for tracking and mapping using the same number of pixel samples ($M_t = 200$ for tracking and $M = 1000$ for mapping). We can notice that our method is over 2 \times and 3 \times faster than iMAP in tracking and mapping. This indicates the advantage of using feature grids with shallow MLP decoders over a single heavy MLP.

Robustness to Dynamic Objects. Here we consider the Co-Fusion dataset [40] which contains dynamically moving objects. As illustrated in Fig. 6, our method correctly identifies and ignores pixel samples falling into the dynamic

	FLOPs [$\times 10^3$] ↓	Tracking [ms] ↓	Mapping [ms] ↓
iMAP [47]	443.91	101	448
NICE-SLAM	104.16	47	130

Table 4. **Computation & Runtime.** Our scene representation does not only improve the reconstruction and tracking quality, but is also faster. The runtimes for iMAP are taken from [47].

object during optimization, which leads to better scene representation modelling (see the rendered RGB and depths). Furthermore, we also compare with iMAP* on the same sequence for camera tracking. The ATE RMSE scores of ours and iMAP* is 1.6cm and 7.8cm respectively, which clearly demonstrates our robustness to dynamic objects.

Geometry Forecast and Hole Filling. As illustrated in Fig. 7, we are able to complete unobserved scene regions thanks to the use of coarse-level scene prior. In contrast, the unseen regions reconstructed by iMAP* are very noisy since no scene prior knowledge is encoded in iMAP*.



Figure 6. Robustness to Dynamic Objects. We show the sampled pixels overlaid on an image with a dynamic object in the center (left), our rendered RGB (middle) and our rendered depth (right) to illustrate the ability of handling dynamic environments. The masked pixel samples during tracking are colored in black, while the used ones are shown in red.



Figure 7. Geometry Forecast and Hole Filling. The white colored area is the region with observations, and cyan indicates the unobserved but predicted region. Thanks to the use of coarse-level scene prior, our method has better prediction capability compared to iMAP*. This in turn also improves our tracking performance.

4.4. Ablation Study

In this section, we investigate the choice of our hierarchical architecture and the importance of color representation.

Hierarchical Architecture. Fig. 8 compares our hierarchical architecture against: a) one feature grid with the same resolution as our fine-level representation (Only High-res); b) one feature grid with mid-level resolution (Only Low-res). Our hierarchical architecture can quickly add geometric details when the fine-level representation participates in the optimization, which also leads to better convergence.

Local BA. We verify the effectiveness of local bundle adjustment on ScanNet [13]. If we do not jointly optimize camera poses for K keyframes together with the scene representation (w/o Local BA in Table 5), the camera tracking is not only significantly less accurate, but also less robust.

Color Representation. In Table 5 we compare our method without the photometric loss \mathcal{L}_p in Eq. (9). It shows that, although our estimated colors are not perfect due to the limited optimization budget and the lack of sampling points, learning such a color representation still plays an important role for accurate camera tracking.

Keyframe Selection. We test our method using iMAP’s keyframe selection strategy (w/ iMAP keyframes in Table 5) where they select keyframes from the entire scene. This is necessary for iMAP to prevent their simple MLP from forgetting the previous geometry. Nevertheless, it also leads to slow convergence and inaccurate tracking.

ATE RMSE (\downarrow)	w/o Local BA	w/o \mathcal{L}_p	w/ iMAP keyframes	Full
Mean	37.74	32.02	12.10	9.63
Std.	30.97	21.98	3.38	0.62

Table 5. Ablation Study. We investigate the usefulness of local BA, color representation, as well as our keyframe selection strategy. We run each scene 5 times and calculate their mean and standard deviation of ATE RMSE (\downarrow). We report the average values over 6 scenes in ScanNet [13].

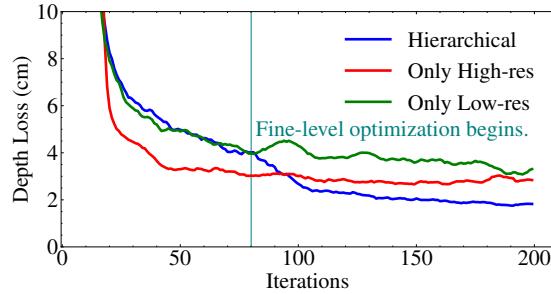


Figure 8. Hierarchical Architecture Ablation. Geometry optimization on a single depth image on Replica [45] with different architectures. The curves are smoothed for better visualization.

5. Conclusion

We presented NICE-SLAM, a dense visual SLAM approach that combines the advantages of neural implicit representations with the scalability of an hierarchical grid-based scene representation. Compared to a scene representation with a single big MLP, our experiments demonstrate that our representation (tiny MLPs + multi-res feature grids) not only guarantees fine-detailed mapping and high tracking accuracy, but also faster speed and much less computation due to the benefit of local scene updates. Besides, our network is able to fill small holes and extrapolate scene geometry into unobserved regions which in turn stabilizes the camera tracking.

Limitations. The predictive ability of our method is restricted to the scale of the coarse representation. In addition, our method does not perform loop closures, which is an interesting future direction. Finally, although traditional methods lack some of the features, there is still a performance gap to the learning-based approaches that needs to be closed.

Acknowledgements. The authors thank the Max Planck ETH Center for Learning Systems (CLS) for supporting Songyou Peng. We also thank Edgar Sucar for providing additional implementation details about iMAP. Special thanks to Chi Wang for offering the data collection site. This work was partially supported by the NSFC (No. 62102356), Zhejiang Lab (2021PE0AC01). Weiwei Xu is partially supported by NSFC (No. 61732016).

References

- [1] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgbd surface reconstruction. In *CVPR*, 2022. 2
- [2] Jonathan T Barron and Ben Poole. The fast bilateral solver. In *European conference on computer vision*, pages 617–632. Springer, 2016. 5
- [3] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J Davison. Codeslam—learning a compact, optimisable representation for dense visual slam. In *CVPR*, 2018. 1, 2
- [4] Aljaž Božič, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner. Transformerfusion: Monocular rgbd scene reconstruction using transformers. *Proc. Neural Information Processing Systems (NeurIPS)*, 2021. 2
- [5] Erik Bylow, Jürgen Sturm, Christian Kerl, Fredrik Kahl, and Daniel Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *RSS*, 2013. 2
- [6] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *ECCV*, 2020. 2
- [7] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *CVPR*, 2021. 2
- [8] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 2
- [9] Jaesung Choe, Sunghoon Im, François Rameau, Minjun Kang, and In So Kweon. Volumefusion: Deep depth fusion for 3d scene reconstruction. In *ICCV*, 2021. 2
- [10] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *CVPR*, 2015. 6, 7
- [11] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996. 2, 5, 6
- [12] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew J Davison. Deepfactors: Real-time probabilistic dense monocular slam. *IEEE Robotics and Automation Letters*, 2020. 1, 2
- [13] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 5, 6, 7, 8, 13, 14, 17
- [14] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 2017. 2
- [15] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE TPAMI*, 40(3):611–625, 2017. 2
- [16] Jiahui Huang, Shi-Sheng Huang, Haoxuan Song, and Shi-Min Hu. Di-fusion: Online implicit 3d reconstruction with deep priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8932–8941, 2021. 2, 5, 6, 7, 16
- [17] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *CVPR*, 2020. 2
- [18] Olaf Kähler, Victor Adrian Prisacariu, and David W. Murray. Real-time large-scale dense 3d reconstruction with loop closure. In *ECCV*, 2016. 2
- [19] Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In *ISMAR*, 2009. 2
- [20] Lukas Koestler, Nan Yang, Niclas Zeller, and Daniel Cremers. Tandem: Tracking and dense mapping in real-time using deep multi-view stereo. In *CoRL*, 2021. 2
- [21] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *ICCV*, 2021. 2
- [22] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *CVPR*, 2020. 2
- [23] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH*, 1987. 13
- [24] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2021. 2
- [25] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 2
- [26] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and N. Ren. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 4
- [27] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 2017. 6
- [28] Zak Murez, Tarrence van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End-to-end 3d scene reconstruction from posed images. In *ECCV*, 2020. 2
- [29] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molynaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 2
- [30] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *ICCV*, 2011. 1, 2
- [31] Michael Niemeyer and Andreas Geiger. Campari: Camera-aware decomposed generative neural radiance fields. In *3DV*, 2021. 2
- [32] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, 2021. 2
- [33] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learn-

- ing implicit 3d representations without 3d supervision. In *CVPR*, 2020. 2
- [34] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 2013. 2
- [35] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *ICCV*, 2021. 2, 4
- [36] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 2
- [37] Songyou Peng, Chiyu "Max" Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. In *NeurIPS*, 2021. 2
- [38] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 1, 2, 4, 12, 13
- [39] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV*, 2021. 2
- [40] Martin Rünz and Lourdes Agapito. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *ICRA*, 2017. 5, 7, 13
- [41] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morigima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV*, 2019. 2
- [42] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. BAD SLAM: bundle adjusted direct RGB-D SLAM. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 134–144. Computer Vision Foundation / IEEE, 2019. 1
- [43] Thomas Schops, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle adjusted direct RGB-D SLAM. In *CVPR*, 2019. 2, 6
- [44] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *NeurIPS*, 2020. 2
- [45] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 5, 6, 8, 13, 14, 15
- [46] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IROS*, 2012. 3, 5, 6, 13
- [47] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew Davison. iMAP: Implicit mapping and positioning in real-time. In *ICCV*, 2021. 1, 2, 4, 5, 6, 7, 8, 14, 15, 16, 17
- [48] Edgar Sucar, Kentaro Wada, and Andrew Davison. Nodeslam: Neural object descriptors for multi-view shape reconstruction. In *3DV*, 2020. 1, 2
- [49] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *CVPR*, 2021. 1, 2
- [50] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS*, 2020. 4
- [51] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment networks. In *ICLR*, 2018. 2
- [52] Zachary Teed and Jia Deng. Deepv2d: Video to depth with differentiable structure from motion. In *ICLR*, 2019. 2
- [53] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. In *NeurIPS*, 2021. 2
- [54] Zachary Teed and Jia Deng. Tangent space backpropagation for 3d transformation groups. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 5
- [55] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Niklaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *CVPR*, 2017. 2
- [56] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 2
- [57] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf-: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021. 2
- [58] Silvan Weder, Johannes L Schonberger, Marc Pollefeys, and Martin R Oswald. Neuralfusion: Online depth fusion in latent space. In *CVPR*, 2021. 2
- [59] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. In *RSS*, 2015. 1, 2
- [60] T. Whelan, J. B. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard. Kintinuous: Spatially Extended KinectFusion. In *RSS '12 Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012. 1, 6
- [61] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomír Mech, and Ulrich Neumann. DISN: deep implicit surface network for high-quality single-view 3d reconstruction. In *NeurIPS*, 2019. 2
- [62] Zike Yan, Yuxin Tian, Xuesong Shi, Ping Guo, Peng Wang, and Hongbin Zha. Continual neural mapping: Learning an implicit scene representation from sequential observations. In *ICCV*, 2021. 2
- [63] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *CVPR*, 2020. 5
- [64] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *NeurIPS*, 2021. 2

- [65] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *NeurIPS*, 2020. [2](#)
- [66] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. In *IROS*, 2020. [2](#)
- [67] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. [2](#)
- [68] Shuaifeng Zhi, Michael Bloesch, Stefan Leutenegger, and Andrew J Davison. Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations. In *CVPR*, 2019. [1](#), [2](#)
- [69] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In *ECCV*, 2018. [2](#)
- [70] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. [6](#)

– Supplementary Material –

NICE-SLAM: Neural Implicit Scalable Encoding for SLAM

Zihan Zhu^{1,2*} Songyou Peng^{2,4*} Viktor Larsson³ Weiwei Xu¹ Hujun Bao¹
 Zhaopeng Cui^{1†} Martin R. Oswald^{2,5} Marc Pollefeys^{2,6}

¹State Key Lab of CAD&CG, Zhejiang University ²ETH Zurich ³Lund University
⁴MPI for Intelligent Systems, Tübingen ⁵University of Amsterdam ⁶Microsoft

In the supplementary material we present the following:

- Implementation details and parameters (Section A)
- Additional experiments and ablations (Section B)

A. Implementation Details

A.1. Frustum Feature Selection

The grid-based representation allows us to only optimize the geometry within the current viewing frustum while keeping the rest of the scene geometry fixed. However, naive optimization for all voxels will affect features even just slightly outside the viewing frustum because of trilinear interpolation. This is illustrated in Fig. Aa. The rays A and B are viewing rays from the current frame and an active keyframe, respectively. Including these rays in the optimization will update the feature at X (marked in the figure) due to trilinear interpolation. However, updating this feature will also affect the ray C coming from an inactive keyframe.

To solve the problem, we propose to only update features fully inside the current viewing frustum during the optimization, see Fig. Ab. In this way, it will not only preserve the previously reconstructed geometry, but also significantly reduce the number of parameters during optimization.

A.2. Hierarchical Feature Grid Initialization

Coarse-level Feature Grid. The coarse-level feature grid is randomly initialized in all experiments.

Mid-level Feature Grid. The mid-level feature grid is also randomly initialized in all experiments, except for the result shown in Fig. 7 in the main paper, where it is initialized to free space to better visualize the predictions from the coarse-level grid. Empirically we find that the random initialization gives slightly better convergence compared to

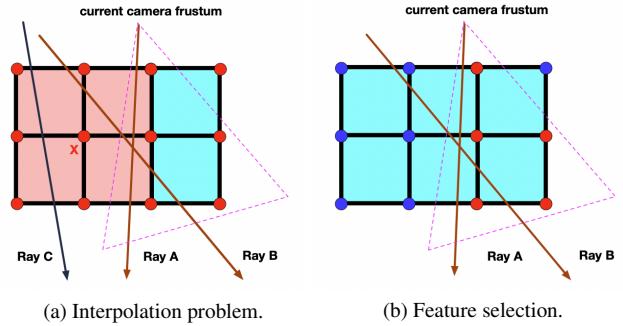


Figure A. 2D illustration of the feature grid. The lattice points correspond to features. The optimized and fixed features are shown in red and blue respectively.

initializing from a fixed feature vector corresponding to the free space.

Fine-level Feature Grid. The fine-level feature grid is initialized to ensure the output of the fine-level decoder f^2 as zero, as it is added in a residual manner onto the occupancy predicted from the mid-level features. This guarantees a smooth energy transition in the coarse-to-fine optimization. During the training of the fine-level decoder from ConvONet [38], we add additional regularization loss to enforce that, if the fine-level feature is zero, no matter what the concatenated mid-level feature is, the output residual should always be zero. This regularization allows us to zero-initialize the fine-level grid at runtime.

A.3. Justification for Design Choices.

Why 3-level Feature Grids? We show in Fig. 8 in the main paper that using hierarchical grids leads to better convergence compared to a single level, and we find that the current design guarantees a good balance between the quality and real-time capability / memory consumption (only 12 MB for Replica scenes). We also conduct an ablation study on the number of levels of feature grids in Table A. It shows that the 3-level feature grid is a good balance between the reconstruction quality and computational efficiency.

*Equal contribution.

†Corresponding author.

Levels	2	3	4
FLOPs [$\times 10^3$] ↓	58.45	104.16	155.95
Depth L1 [cm] ↓	1.86	1.87	1.96
Acc. [cm] ↓	2.87	2.78	3.15
Comp. [cm] ↓	2.76	2.76	2.40
Comp. Ratio [$< 5\text{cm} \%$] ↑	91.24	91.37	93.60

Table A. **Ablation on the Levels of Feature Grids.** Reconstruction results on Replica room-0 with ground truth camera pose.

Why is the Mid-level Output not a Residual to the Coarse-level Output? The coarse grid has a significantly larger voxel size (side of > 1 meter) than the mid and fine levels, so updating the coarse-level feature would affect a large area. To ensure small local updates for efficiency, we disconnect coarse level from mid and fine levels, and only use coarse level for prediction.

A.4. Mesh Visualization

The reconstructed scene is represented implicitly using hierarchical feature grids. We use the marching cubes algorithm [23] to create a mesh for the visualization purpose. For every observed point we predict its occupancy value using the fine-level decoder and color from the color decoder. For those unseen points in the predicted regions (i.e. voxels with partial observations in the coarse grid), we predict occupancy from the coarse-decoder and set the color to cyan for visualization as shown in Fig. 8 in our main paper and the supplementary video. Other points are assigned zero occupancy. The same resolution is used in marching cubes for both iMAP* and NICE-SLAM.

A.5. Decoder Pretraining

We use the Synthetic Indoor Scene Dataset provided in ConvONet [38] to pre-train the encoder-decoder. Furthermore, we use the Point Cloud Encoder instead of the Voxel Encoder. All levels are trained with room_grid64 setting in ConvONet [38]. The feature dimension for all the feature grids is 32. As for hyperparameters used for the pretraining process, we follow the same setting as ConvONet [38].

A.6. Hyperparameters

Here we report detailed hyperparameters of online tracking and mapping used for both NICE-SLAM and iMAP*. We perform tracking for every frame and optimize the geometry every fifth frame, except for TUM RGB-D where we optimize the geometry every frame. All parameters are tuned to keep a good balance between the accuracy and the efficiency.

NICE-SLAM. For scene geometry optimization, we use a maximum of 60 iterations for all datasets. In terms of tracking, we use 10 iterations for small-scale synthetic datasets (Replica and Co-Fusion). For the large-scale real datasets including ScanNet and our self-captured scene, we use 50

Mapping Iterations	15	30	60	120	240
Depth L1 [cm] ↓	2.31	2.03	1.87	1.74	1.59
Acc. [cm] ↓	2.90	2.84	2.78	2.80	2.78
Comp. [cm] ↓	3.14	2.91	2.76	2.65	2.50
Comp. Ratio [$< 5\text{cm} \%$] ↑	89.15	90.55	91.37	91.94	92.76

Table B. **Ablation on Mapping Iterations.** Reconstruction results on Replica room-0 with ground truth camera poses.

iterations for tracking. For TUM RGB-D dataset we use 200 iterations.

The learning rate for tracking on Replica [45], TUM RGB-D [46], ScanNet [13], Self-captured, and Co-Fusion [40] are $1e-3$, $1e-2$, $5e-4$, $3e-3$, $1e-3$ respectively. The learning rate for optimizing the coarse-level is $1e-3$, for mid-level is $1e-1$, for fine- and color-level is $5e-3$. The learning rate for selected keyframes' camera parameters during the mapping is $1e-3$, except for Co-Fusion where we set the learning rate to 0.

iMAP*. For all datasets except TUM RGB-D [46], we use 50 iterations for tracking and 300 iterations for joint optimization. For TUM RGB-D [46], we use 200 and 300 iterations respectively. The learning rate for tracking on Replica [45], TUM RGB-D [46], ScanNet [13], Self-captured, and Co-Fusion [40] are $5e-4$, $5e-3$, $2e-3$, $1e-3$, $5e-4$ respectively. The learning rate for joint optimization is $2e-4$.

B. Additional Experiments

B.1. Frame Loss Robustness

We simulate extreme frame loss on ScanNet scene0000_00 by skipping 100 frames from frame ID 2001 to 2100. As visualized in Fig. B, iMAP* struggles to recover camera poses and scene geometry, even given 1500 iterations. In contrast, our NICE-SLAM is able to recover the camera pose using only 300 iterations. This is due to the use of coarse-level geometric representation which improves the prediction capability.

B.2. Number of Mapping and Tracking Iterations

We show in Fig. C how the number of tracking and mapping iterations affects the tracking performance. We also give ground truth camera pose and evaluate reconstruction with different mapping iterations in Table B.

B.3. Frustum Feature Selection

To highlight the importance of current frustum feature selection (see Section A.1), we run our system with and without the selection process. The results are shown in Fig. D. Without fixing the border features, significant artifacts appear in the reconstruction (Fig. Aa).

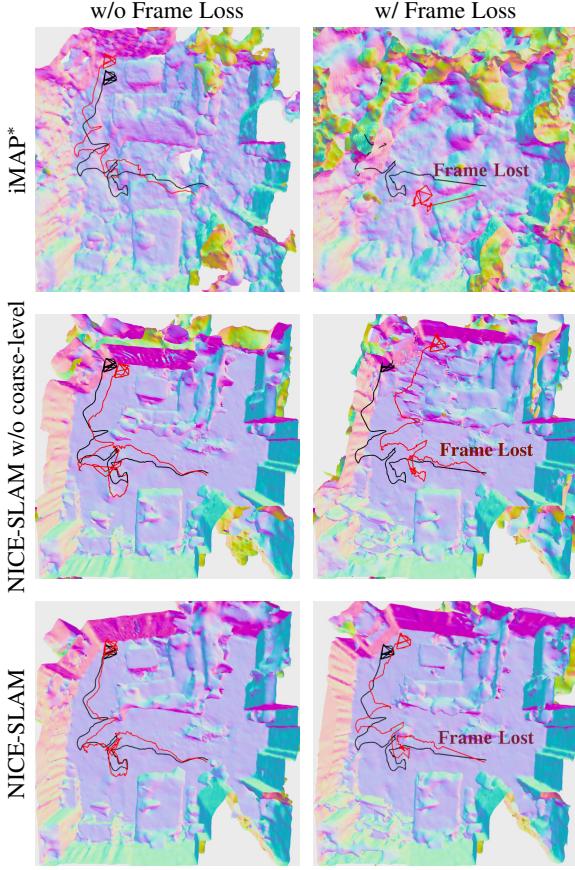


Figure B. **Robustness to Frame Loss.** We show the results at frame 2100 after frame loss at frame 2000. The black trajectory is the ground truth from ScanNet [13], and the red trajectory indicates tracking results. The missing frames corresponds to the straight line in the middle.

B.4. More Results on Replica Dataset [45]

Here we provide the detailed results for all Replica scenes. Table C shows the quantitative comparison when considering the average metric values for 5 consecutive runs, and only evaluate without unseen regions that are outside all camera’s viewing frustums. What is more, as done in [47] we also report the best metrics in 5 consecutive runs under all regions in Table D. As can be noticed, our iMAP re-implementation iMAP* has similar performance over the original iMAP.

In addition, to better highlight the performance differences, we provide additional visualizations using different rendering settings in Fig. E.

B.5. More Results on ScanNet [13]

We show the 3D reconstruction process of iMAP* and NICE-SLAM on ScanNet scene 0000 in Fig. F.

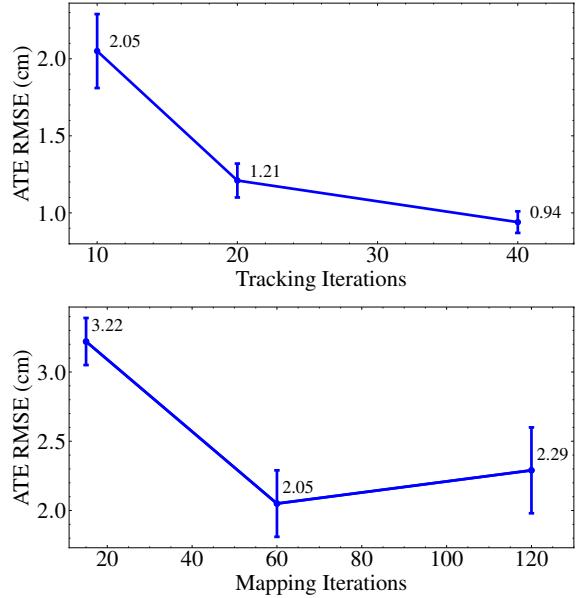


Figure C. **Ablation on the tracking performance.** ATE RMSE (cm) is used as the metric.

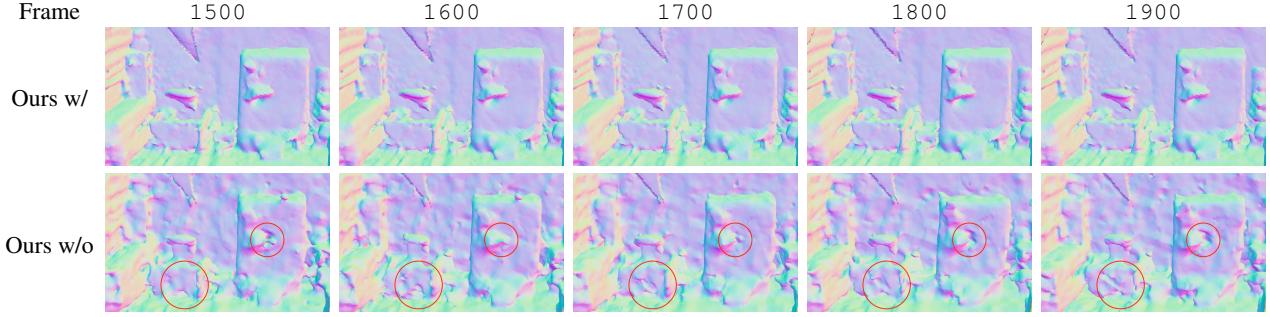


Figure D. Ablation on Frustum Feature Selection. We show our method with and without the frustum feature selection run on sequence scene0000_00 in the ScanNet datasets. During these frames the camera is scanning other parts of the scene. The cutout shown in the figure is part of the previously reconstructed geometry and should remain constant. The mesh is visualized with the vertex normal.

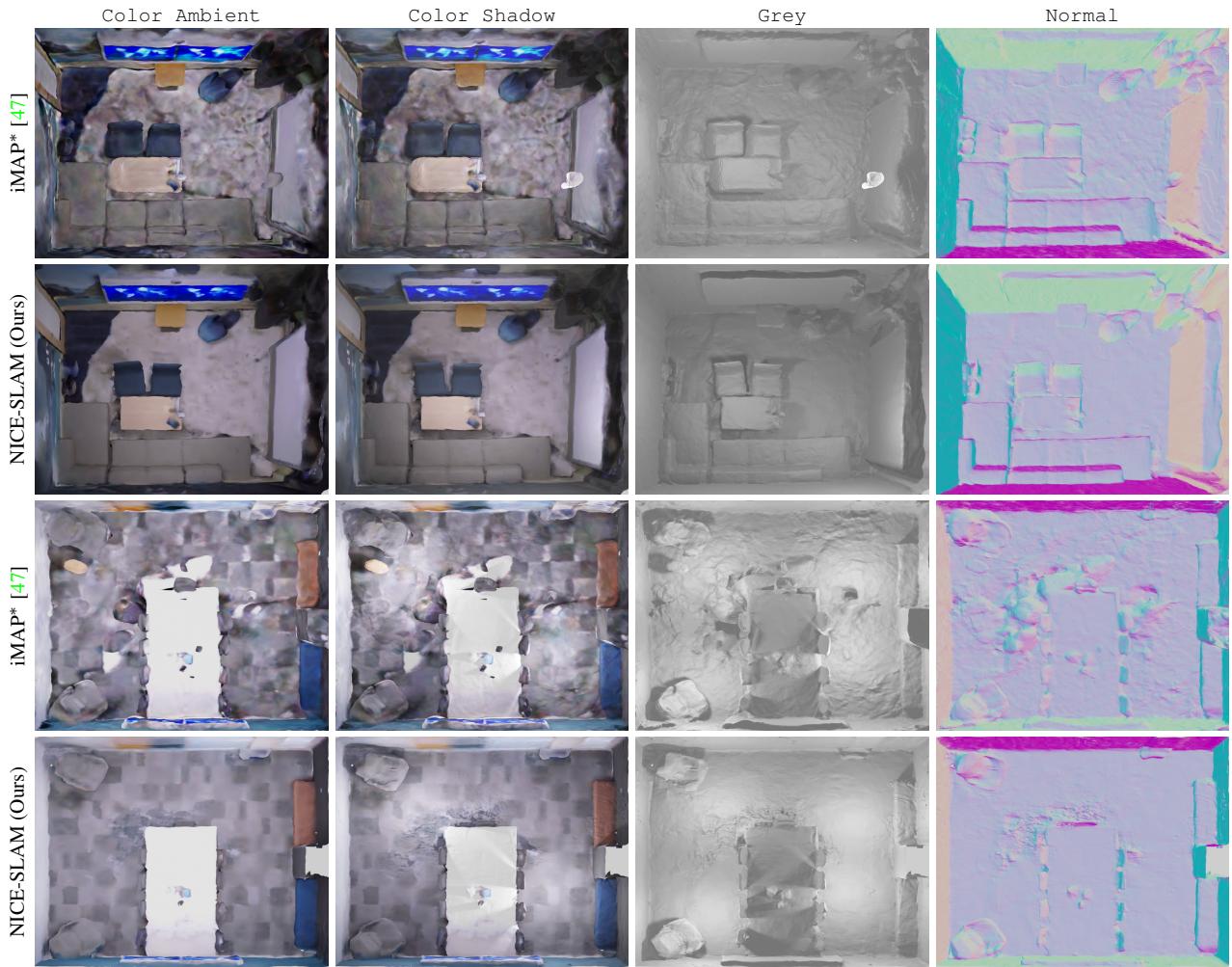


Figure E. More Results on Replica Dataset [45]. We visualize the final reconstruction on two scenes including office-0 (top two rows) and office-4 (bottom two rows). To better show the differences, we use different rendering settings. As can be visualized, our NICE-SLAM produces high-quality geometry and colors.

		room-0	room-1	room-2	office-0	office-1	office-2	office-3	office-4	Avg.
TSDF-Fusion Res. = 512 (536.87MB)	Depth L1 [cm] ↓	6.38	5.33	6.84	4.74	4.62	11.32	9.89	6.49	6.95
	Acc. [cm] ↓	1.87	2.48	1.69	1.14	0.96	1.63	2.08	1.74	1.70
	Comp. [cm] ↓	3.60	3.20	2.85	1.72	2.31	3.66	3.69	3.91	3.12
	Comp. Ratio [$< 5\text{cm} \%$] ↑	88.33	89.82	90.38	93.55	90.35	86.74	85.35	86.31	88.85
TSDF-Fusion Res. = 256 (67.10MB)	Depth L1 [cm] ↓	6.69	5.47	7.47	4.97	5.28	12.30	11.17	7.20	7.57
	Acc. [cm] ↓	1.76	2.11	1.59	1.15	0.97	1.56	1.98	1.66	1.60
	Comp. [cm] ↓	3.85	3.36	3.33	1.93	2.68	4.17	4.22	4.37	3.49
	Comp. Ratio [$< 5\text{cm} \%$] ↑	86.29	88.44	86.63	91.73	87.88	82.95	81.31	83.38	86.08
iMAP* [47] (1.04MB)	Depth L1 [cm] ↓	5.70	4.93	6.94	6.43	7.41	14.23	8.68	6.80	7.64
	Acc. [cm] ↓	5.66	5.31	5.64	7.39	11.89	8.12	5.62	5.98	6.95
	Comp. [cm] ↓	5.20	5.16	5.04	4.35	5.00	6.33	5.47	6.10	5.33
	Comp. Ratio [$< 5\text{cm} \%$] ↑	67.67	66.41	69.27	71.97	71.58	58.31	65.95	61.64	66.60
DI-Fusion [16] (3.78MB)	Depth L1 [cm] ↓	6.66	96.82	36.09	7.36	5.05	13.73	11.41	9.55	23.33
	Acc. [cm] ↓	1.79	49.00	26.17	70.56	1.42	2.11	2.11	2.02	19.40
	Comp. [cm] ↓	3.57	39.40	17.35	3.58	2.20	4.83	4.71	5.84	10.19
	Comp. Ratio [$< 5\text{cm} \%$] ↑	87.77	32.01	45.61	87.17	91.85	80.13	78.94	80.21	72.96
NICE-SLAM (12.02MB)	Depth L1 [cm] ↓	2.11	1.68	2.90	1.83	2.46	8.92	5.93	2.38	3.53
	Acc. [cm] ↓	2.73	2.58	2.65	2.26	2.50	3.82	3.50	2.77	2.85
	Comp. [cm] ↓	2.87	2.47	3.00	2.02	2.36	3.57	3.83	3.84	3.00
	Comp. Ratio [$< 5\text{cm} \%$] ↑	90.93	92.80	89.07	94.93	92.61	85.20	82.98	86.14	89.33

Table C. Reconstruction Results for the Replica Dataset (Average of 5 runs).

		room-0	room-1	room-2	office-0	office-1	office-2	office-3	office-4	Avg.
TSDF-Fusion Res. = 512 (536.87MB)	Acc. [cm]	5.20	2.83	1.60	1.66	1.06	2.29	2.50	2.18	2.42
	Comp. [cm]	5.05	4.60	4.50	1.06	9.57	5.84	4.16	4.30	4.89
	Comp. Ratio [$< 5\text{cm} \%$] ↑	75.07	79.03	86.01	80.19	77.80	80.69	82.29	83.00	80.51
	TSDF-Fusion Res. = 256 (67.10MB)	4.17	2.69	1.49	1.65	1.09	2.24	2.37	2.16	2.23
iMAP [47] (1.04MB)	Acc. [cm] ↓	3.58	3.69	4.68	5.87	3.71	4.81	4.27	4.83	4.43
	Comp. [cm] ↓	5.06	4.87	5.51	6.11	5.26	5.65	5.45	6.59	5.56
	Comp. Ratio [$< 5\text{cm} \%$] ↑	83.91	83.45	75.53	77.71	79.64	77.22	77.34	77.63	79.06
	iMAP* [47] (1.04MB)	4.07	3.86	5.17	5.40	4.04	5.23	4.30	4.98	4.63
DI-Fusion [16] (3.78MB)	Acc. [cm] ↓	4.73	4.32	5.53	4.95	5.27	5.40	4.94	5.08	5.03
	Comp. [cm] ↓	3.90	82.87	20.16	12.08	8.76	6.89	5.70	5.96	18.29
	Comp. Ratio [$< 5\text{cm} \%$] ↑	86.58	24.77	41.50	74.20	79.22	73.36	70.24	78.26	66.02
	NICE-SLAM (12.02MB)	2.97	3.23	3.46	5.47	3.33	4.40	3.55	2.87	3.66

Table D. Reconstruction Results for the Replica Dataset (Best in 5 runs). The numbers for iMAP are directly taken from [47].

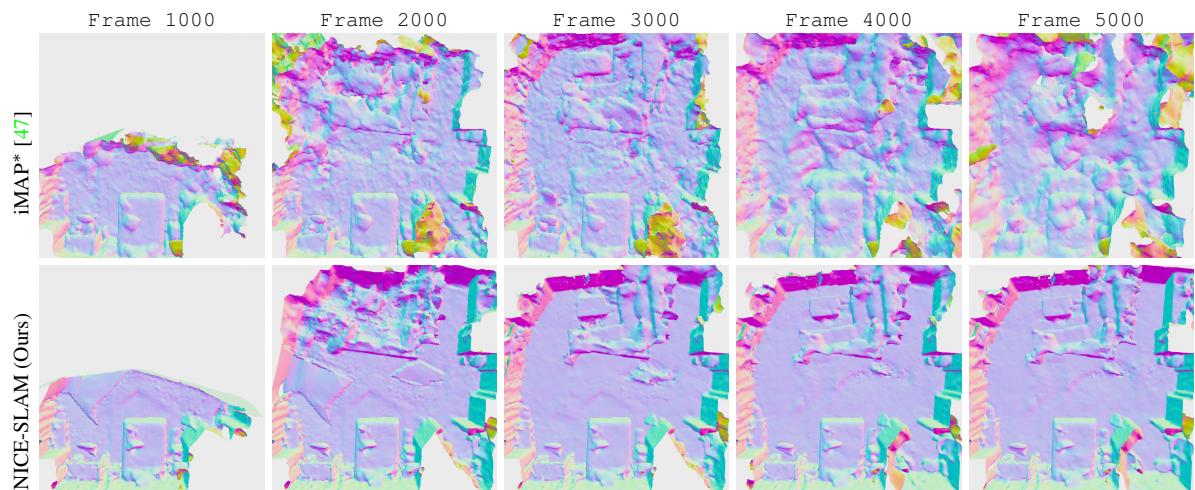


Figure F. 3D Reconstruction Process on ScanNet [13]. Due to our local map updates the resulting geometry is temporally more stable and often less noisy compared to iMAP* [47].