NICE-SLAM: Neural Implicit Scalable Encoding for SLAM

Zihan Zhu\textsuperscript{1,2}, Songyou Peng\textsuperscript{1,3,}\* Songyou Peng\textsuperscript{1,3,}\* Viktor Larsson\textsuperscript{1}, Weiwei Xu\textsuperscript{2}, Hujun Bao\textsuperscript{2}, Zhaopeng Cui\textsuperscript{2,}\† Martin R. Oswald\textsuperscript{1,4}, Marc Pollefeys\textsuperscript{1,5}

\textsuperscript{1}ETH Zurich \quad \textsuperscript{2}State Key Lab of CAD&CG, Zhejiang University \quad \textsuperscript{3}MPI for Intelligent Systems, Tübingen \quad \textsuperscript{4}University of Amsterdam \quad \textsuperscript{5}Microsoft

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\cite{25,37,52,53} 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\cite{2,11,43,61} 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.\cite{42} 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\cite{42} 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\cite{33,44} demonstrate that establishing multi-
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 will be made publicly available soon.

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. [16], 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 [25]. Because of its simplicity, DTAM has been widely adapted in many recent learning-based SLAM systems. For example, [49, 62] regress both depth and pose updates. DeepV2D [47] similarly alternates between regressing depth and pose estimation but uses test-time optimization. BA-Net [46] and DeepFactors [11] simplify the optimization problem by using a set of basis depth maps. There are also methods, e.g., CodeSLAM [2], SceneCode [61] and NodeSLAM [43], which optimize a latent representation that decodes into the keyframe or object depth maps. DROID-SLAM [48] uses regressed optical flow to define geometrical residuals for its refinement. On the other hand, the world-centric map representation anchors the 3D geometry in uniform world coordinates, and can be further divided into surfels [38, 52] and voxel grids, typically storing occupancies or TSDF values [10]. Voxel grids have been used extensively in RGB-D SLAM, e.g. KinectFusion [24] among other works [4, 13, 15, 29]. 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 during them 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 [7, 18, 20, 28, 30–32, 36, 50, 54, 57, 58], scene completion [5, 14, 33], novel view synthesis [19, 21, 34, 60] and also generative modelling [6, 26, 27, 39]. A few recent papers [1, 3, 8, 23, 44] attempt to predict scene-level geometry with RGB-(D) inputs, but they all assume given camera poses. Another set of works [17, 51, 59] 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 [42]. 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. iMAP shows the ability to simultaneously estimate the scene geometry and color, as well as camera tracking. Nevertheless, due to the limited model capacity of a single MLP, their predicted reconstruction and colors do not capture enough details. Moreover, we also notice that iMAP fails when the scene is getting larger. Similarly, Continual Neural Mapping [55] fuses a sequential stream of depth maps into a continuous neural scene representation. Similar to iMAP, the entire scene is encoded within a single MLP which limits the scalability of both methods.

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 guaranteeing faster convergence. Notably, the works [14, 33] also combine traditional grid structures with learned feature representations for scalability, but neither of them is real-time capable.

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

---

1 Our re-implementation, details in Section 4.
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 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 $\phi_\theta^l$ 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 $\psi_\omega$ and decoder $g_\omega$ to model the scene appearance. Here $\theta$ and $\omega$ indicate the optimizable parameters for geometry and colors, 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 the 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 32cm and 16cm respectively, except for TUM RGB-D [4] 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 $p \in \mathbb{R}^3$, we get the occupancy as

$$o^1_p = f^1(p, \phi_\theta^1(p)), \quad (1)$$

where $\phi_\theta^1(p)$ denotes that the feature grid is tri-linearly interpolated at the point $p$. The relatively low-resolution allow us to efficiently optimize the grid features to fit the observations. To capture smaller high-resolution 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.,

$$\Delta o^1_p = f^2(p, \phi_\theta^1(p), \phi_\theta^2(p)), \quad (2)$$

where the final occupancy for a point is given by

$$o_p = o^1_p + \Delta o^1_p. \quad (3)$$

Note that we fix the pre-trained decoders $f^1$ and $f^2$, and only optimize the feature grids $\phi_\theta^1$ and $\phi_\theta^2$ 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.,

$$o_0^p = f^0(p, \phi_0^p(p)). \quad (4)$$

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

**Pre-training the 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 [33] 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 [33]. 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 $\phi_0^p(p)$ from the mid-level together with the fine-level feature $\phi_0^p(p)$ 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 signal for tracking. To encode the color in the scene, we apply another feature grid $\psi_0$ and decoder $g_0$:

$$c_p = g_0(p, \psi_0(p)), \quad (5)$$

where $\omega$ indicates learnable parameters during optimization. Different from the geometry that has strong prior knowledge, we empirically found that jointly optimizing the color features $\psi_0$ and decoder $g_0$ improves tracking performance. Note that this, similarly to iMAP [42], 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 [42, 45] to $p$ 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 [21], 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 $r$ of a pixel coordinate. We first sample along this ray $N_{\text{strat}}$ points for stratified sampling, and also uniformly sample $N_{\text{imp}}$ points near to the depth $\hat{D}$. In total we sample $N = N_{\text{strat}} + N_{\text{imp}}$ points for each ray. More formally, let $p_i = o + d_ir, i \in \{1, \cdots, N\}$ denote the sampling points on the ray $r$ given the camera origin $o$, and $d_i$ corresponds to the depth value of $p_i$ along this ray. For every point $p_i$, we can calculate their coarse-level occupancy probability $o_{0c}^i$, fine-level occupancy probability $o_{0f}^i$, and color value $c_i$ using Eq. (4), Eq. (3), and Eq. (5). Similar to [30], we model the ray termination probability at point $p_i$ as $w_i^c = o_{0c}^i \prod_{j=1}^{i-1}(1-o_{0c}^j)$ for coarse level, and $w_i^f = o_{0f}^i \prod_{j=1}^{i-1}(1-o_{0f}^j)$ for fine level.

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

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

Moreover, we also calculate depth variances along the ray:

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

### 3.3. Robust Alternating Optimization

In this section, we provide details on the optimization of the scene geometry $\theta$ and appearance $\omega$ parameters of our hierarchical scene representation, and of the camera poses.

**Optimization for Scene Representations.** 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 iterative optimization in a staged fashion to minimize the geometric and photometric losses.

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

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

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

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

At the first stage, we optimize only the mid-level feature grid $\phi_g^m$ using the geometric loss $\mathcal{L}_g^f$ in Eq. (8). Next, we

\footnote{We empirically define the sampling interval as $\pm 0.05D$, where $D$ is the depth value of the current ray.}
jointly optimize both the mid and fine-level \( \phi_g^1, \phi_g^2 \) features with the same fine-level depth loss \( L_g^f \). Finally, we jointly optimize feature grids at all levels and the color decoder using the loss with a weighting factor \( \lambda_p \):

\[
\min_{\theta, \omega} \left( L_g^c + L_g^f + \lambda_p L_p \right) .
\]

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.

**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 \( \{ R, t \} \). 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:

\[
L_{g, \text{var}} = \frac{1}{M_t} \sum_{m=1}^{M_t} \frac{|D_m - \hat{D}_m^c|}{\sqrt{\hat{D}_{\text{var}}^c}} + \frac{|D_m - \hat{D}_m^f|}{\sqrt{\hat{D}_{\text{var}}^f}} .
\]

The modified loss down-weights less certain regions in the reconstructed geometry \([42, 56]\), e.g., object edges.

The final loss for tracking is:

\[
\min_{R, t} \left( L_{g, \text{var}} + \lambda_{pt} L_p \right) .
\]

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. 7 shows an example where a dynamic object is ignored since it is not present in the rendered RGB and depth image.

### 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 \([42]\), where we incrementally add new keyframes based on the information gain. However, in contrast to iMAP \([42]\), 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 \([42]\). 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 random 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.

## 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 support our design choices.

### 4.1. Experimental Setup

**Datasets.** We consider 5 datasets ranging from a single small room to a larger apartment consisting of multiple rooms. To evaluate the scene reconstruction, we choose the Replica dataset \([40]\), which contains highly photo-realistic 3D indoor scene reconstruction at both room and flat scale. To evaluate the tracking, we use TUM RGB-D dataset \([41]\) since it provides ground truth camera trajectory. Moreover, to assess the scalability, we consider ScanNet \([12]\) in addition to one self-captured large-scale indoor dataset. Furthermore, we also run our system on the Co-Fusion dataset \([35]\) to show our ability of handling dynamic objects.

**Metrics.** Following \([42]\), we sample 200,000 points from both ground-truth and reconstructed meshes and consider Accuracy, Completion, and Completion Ratio for scene geometry evaluation. As for the evaluation of camera tracking, we use ATE RMSE \([41]\). See \([42]\) for details of all metrics.

**Our iMAP Re-implementation: iMAP\(^*\).** The authors of iMAP \([42]\) have not released its source code before the submission. To have a fair comparison, we faithfully re-implemented iMAP on our own, compare its performance separately and thus denote it as iMAP\(^*\). As reported in Table 1 and Table 2, our re-implementation shows similar performance compared to the original iMAP in both scene reconstruction and camera tracking. Whenever available, we also reported the results directly from \([42]\).

**Implementation Details.** We run our SLAM system on a desktop PC with a 3.80GHz Intel i7-10700K CPU and 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
The numbers for iMAP are directly taken from [42]. iMAP K D dataset, we use Net, we compare with our re-implemented iMAP different methods. Since iMAP does not evaluate on ScanNet [12] to benchmark the scalability of We select multiple large Evaluation on ScanNet [12]. nevertheless, our method significantly reduces the gap between explicit scene representations (iMAP [42] and ours). Nevertheless, our method mostly outperforms iMAP on various reconstruction metrics. NICE-SLAM reduces the gap between SLAM methods with neural implicit scene representations and traditional SLAM approaches, but cannot close it. ATE RMSE ↓ is used as the evaluation metric.

| Scene ID | room-0 | room-1 | room-2 | office-0 | office-1 | office-2 | office-3 | office-4 | Avg. |
|----------|--------|--------|--------|----------|----------|----------|----------|----------|------|
| iMAP [42] | 197.1  | 18.9  | 19.0  | 96.4  | 37.3  | 28.7  | 66.2  |
| NICE-SLAM* | 11.3  | 12.0  | 9.0  | 12.0  | 12.8  | 11.4  |

Table 3. Camera Tracking Results on ScanNet [12]. Our approach yields consistently better results on this dataset. ATE RMSE ↓ is used as the evaluation metric.

sharper and more detailed geometry over iMAP*. Moreover, in Fig. 5 we show comparison on more scenes. As can be observed, iMAP* either completely fails or introduces large drifting, while our method successfully reconstructs the entire scene. These results clearly indicates the effectiveness of our hierarchical scene representations especially on large scenes.

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. 6 show the reconstructions obtained using NICE-SLAM and iMAP*. For reference we also show the 3D reconstruction using the offline tool Redwood [9] in Open3D [63]. We can see that NICE-SLAM has comparable results with the offline method, while iMAP* 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.

Computation Complexity. First, we compare the num-

![Table 1. Reconstruction Results for the Replica Dataset [40]. NICE-SLAM mostly outperforms iMAP on various reconstruction metrics. The numbers for iMAP are directly taken from [42]. iMAP* indicates our re-implementation of iMAP.](image)

![Table 2. Camera Tracking Results on TUM RGB-D [41]. Our approach yields consistently better results on this dataset. ATE RMSE ↓ is used as the evaluation metric.](image)

![Table 3. Camera Tracking Results on ScanNet [12]. Our approach yields consistently better results on this dataset. ATE RMSE ↓ is used as the evaluation metric.](image)
number of floating point operations (FLOPs) needed for query-
ing 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× and 3× 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 [35] which contains dynamically moving objects. As illustrated in Fig. 7, our method correctly...
Table 4. Runtime comparison with iMAP. Our scene representation does not only improve the reconstruction and tracking quality, but is also faster. The runtimes for iMAP are taken from [42].

Table 5. Ablation on Color Representation. The evaluation metric is ATE RMSE(\(\downarrow\)). “F” indicates failed camera tracking.

Figure 7. 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 8. 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.

Figure 9. Hierarchical Architecture Ablation. Geometry optimization on a single depth image on Replica [40] with different architectures. The curves are smoothed for better visualization.
References

[1] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. arXiv preprint arXiv:2104.04532, 2021. 2

[2] 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

[3] Aljaž Božič, Pablo Palafao, Justus Thies, Angela Dai, and Matthias Nießner. Transformerfusion: Monocular rgb scene reconstruction using transformers. Proc. Neural Information Processing Systems (NeurIPS), 2021. 2

[4] 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

[5] 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 CVPR, 2020. 2

[6] 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

[7] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In CVPR, 2019. 2

[8] Jaesung Choe, Sunghoon Im, François Rameau, Minjun Kang, and In So Kweon. Volumefusion: Deep depth fusion for 3d scene reconstruction. In ICCV, 2021. 2

[9] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust local shapes: Learning local sdf priors for detailed 3d reconstruction. In ECCV, 2020. 2

[10] Michael Niemeyer and Andreas Geiger. Differentiable volumetric rendering: Learning scenes as compositional generative neural feature fields. In CVPR, 2017. 5, 6, 7

[11] 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

[12] Olaf Kähler, Victor Adrian Prisacariu, and David W. Murray. Real-time large-scale dense 3d reconstruction with loop closure. In ECCV, 2016. 2

[13] Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In ISMAR, 2009. 2

[14] 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

[15] Olaf Kähler, Victor Adrian Prisacariu, and David W. Murray. Real-time large-scale dense 3d reconstruction with loop closure. In ECCV, 2016. 2

[16] Chiyu Jiang, Avnesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In CVPR, 2020. 2

[17] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In ICCV, 2021. 2

[18] 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

[19] Ricardo Martin-Brualla, Nohea Radwan, Mehdi SM Saajadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In CVPR, 2021. 2

[20] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In CVPR, 2019. 2

[21] 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

[22] 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

[23] Zak Murez, Torrence 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

[24] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, 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

[25] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In ICCV, 2011. 1, 2

[26] Michael Niemeyer and Andreas Geiger. Campari: Camera-aware decomposed generative neural radiance fields. arXiv preprint arXiv:2103.17269, 2021. 2

[27] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In CVPR, 2021. 2

[28] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In CVPR, 2020. 2

[29] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. ACM Transactions on Graphics (ToG), 2017. 2

[30] Chiyu Jiang, Avnesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In CVPR, 2020. 2

[31] Olaf Kähler, Victor Adrian Prisacariu, and David W. Murray. Deep3d: Learning continuous signed distance functions for shape representation. In CVPR, 2019. 2

[32] 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
