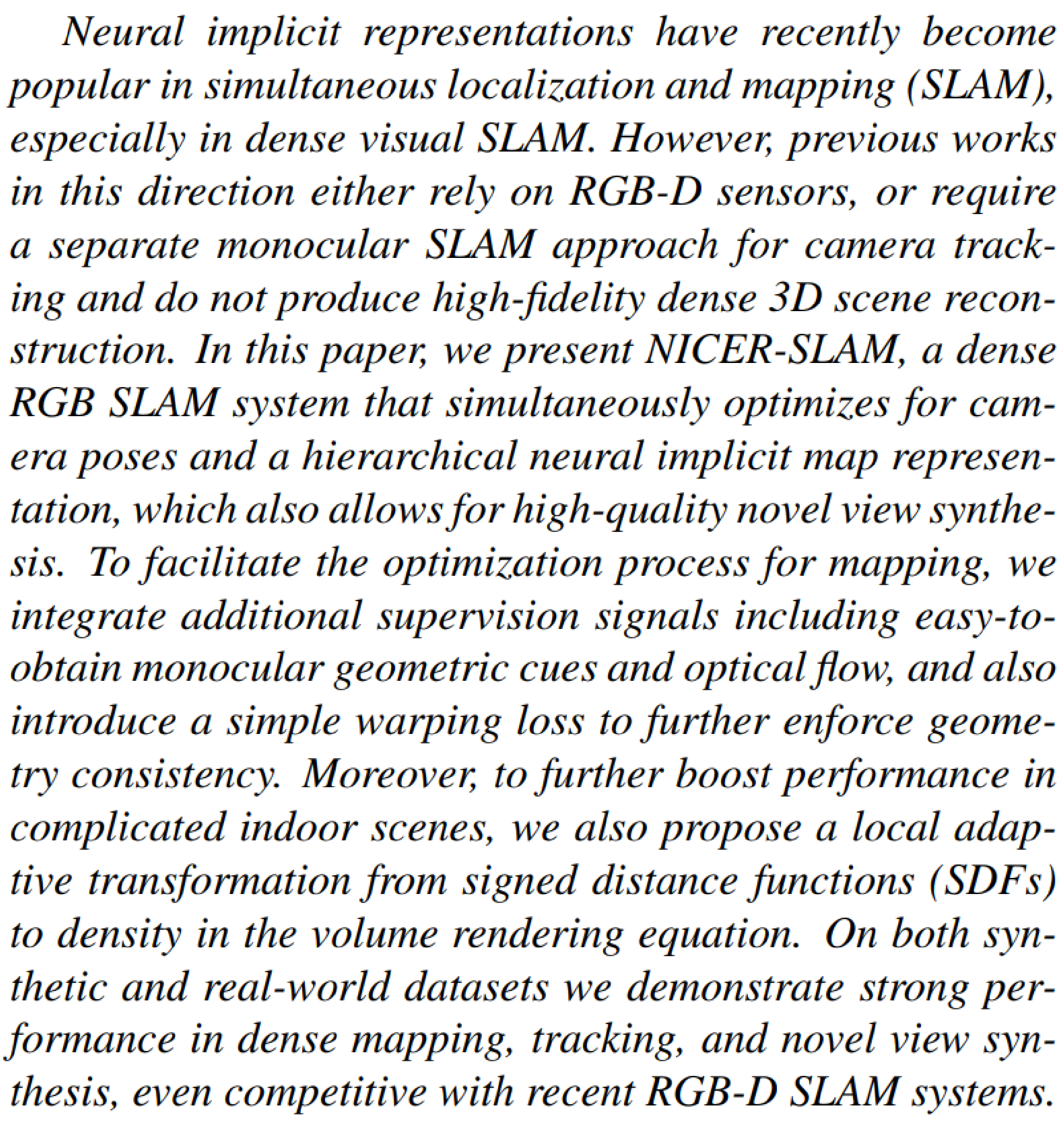
小组汇报NICER-SLAM阅读笔记

ABSTRACT



Neural implicit representations have recently become popular in simultaneous localization and mapping (SLAM), especially in dense visual SLAM. However, previous works in this direction either rely on RGB-D sensors, or require a separate monocular SLAM approach for camera tracking and do not produce high-fidelity dense 3D scene reconstruction. In this paper, we present NICER-SLAM, a dense RGB SLAM system that simultaneously optimizes for camera poses and a hierarchical neural implicit map representation, which also allows for high-quality novel view synthesis. To facilitate the optimization process for mapping, we integrate additional supervision signals including easy-to-obtain monocular geometric cues and optical flow, and also introduce a simple warping loss to further enforce geometry consistency. Moreover, to further boost performance in complicated indoor scenes, we also propose a local adaptive transformation from signed distance functions (SDFs) to density in the volume rendering equation. On both synthetic and real-world datasets we demonstrate strong performance in dense mapping, tracking, and novel view synthesis, even competitive with recent RGB-D SLAM systems.

CONTRIBUTION

In summary, we make the following contributions:

- We present NICER-SLAM, one of the first dense RGB-only SLAM that is end-to-end optimizable for both tracking and mapping, which also allows for high-quality novel view synthesis.
- We introduce a hierarchical neural implicit encoding for SDF representations, different geometric and motion regularizations, as well as a locally adaptive SDF to volume density transformation.
- We demonstrate strong performances in mapping, tracking, and novel view synthesis on both synthetic and real-world datasets, even competitive with recent RGB-D SLAM methods.

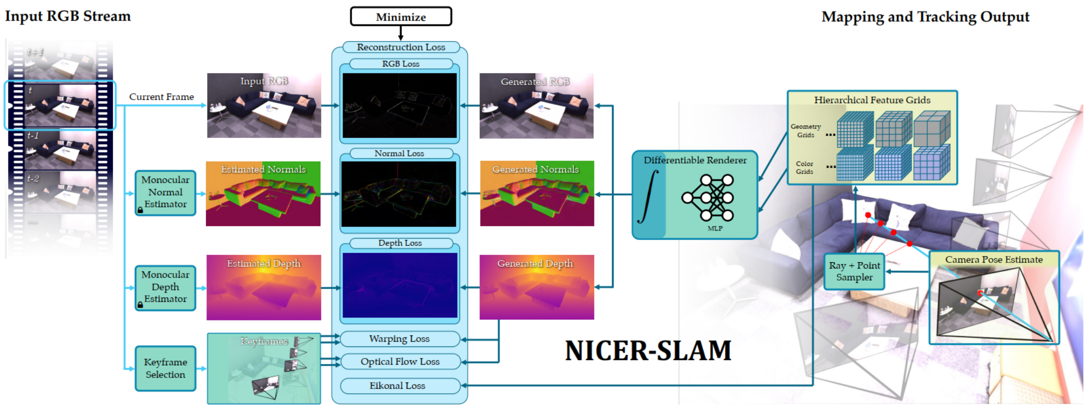
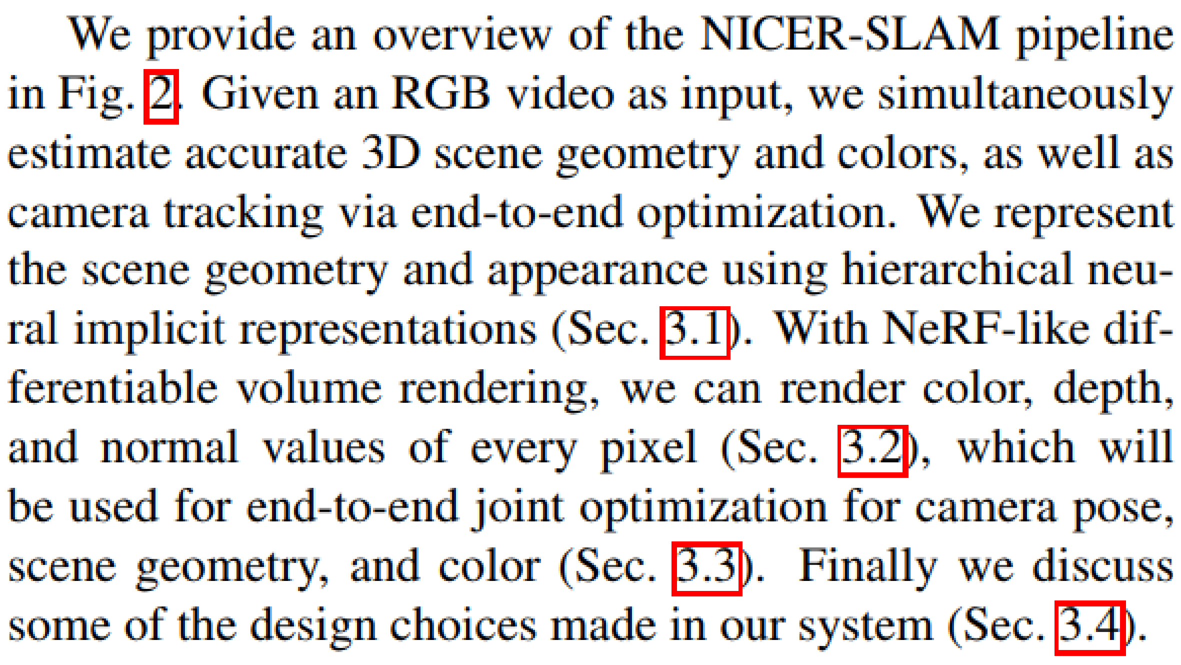


Figure 2: **System Overview.** Our method takes only an RGB stream as input and outputs both the camera poses as well as a learned hierarchical scene representation for geometry and colors. To realize an end-to-end joint mapping and tracking, we render predicted colors, depths, normals and optimize wrt. the input RGB and monocular cues. Moreover, we further enforce the geometric consistency with an RGB warping loss and an optical flow loss.

METHOD


We provide an overview of the NICER-SLAM pipeline in Fig. 2. Given an RGB video as input, we simultaneously estimate accurate 3D scene geometry and colors, as well as camera tracking via end-to-end optimization. We represent the scene geometry and appearance using hierarchical neural implicit representations (Sec. 3.1). With NeRF-like differentiable volume rendering, we can render color, depth, and normal values of every pixel (Sec. 3.2), which will be used for end-to-end joint optimization for camera pose, scene geometry, and color (Sec. 3.3). Finally we discuss some of the design choices made in our system (Sec. 3.4).

3.1. Hierarchical Neural Implicit Representations

3.1. Hierarchical Neural Implicit Representations

We first introduce our optimizable hierarchical scene representations that combine multi-level grid features with MLP decoders for SDF and color predictions.

Coarse-level Geometric Representation. The goal of the coarse-level geometric representation is to efficiently model the coarse scene geometry (objects without capturing geometric details) and the scene layout (e.g. walls, floors) even with only partial observations. To this end, we represent the normalized scene with a dense voxel grid with a resolution of $32 \times 32 \times 32$, and keep a 32-dim feature in each voxel. For any point $\mathbf{x} \in \mathbb{R}^3$ in the space, we use a small MLP f^{coarse} with a single 64-dim hidden layer to obtain its base SDF value $s^{\text{coarse}} \in \mathbb{R}$ and a geometric feature $\mathbf{z}^{\text{coarse}} \in \mathbb{R}^{32}$ as:

$$s^{\text{coarse}}, \ \mathbf{z}^{\text{coarse}} = f^{\text{coarse}}\big(\gamma(\mathbf{x}), \Phi^{\text{coarse}}(\mathbf{x})\big), \tag{1}$$

where γ corresponds to a fixed positional encoding [29, 54] mapping the coordinate to higher dimension. Following [71, 69, 68], we set the level for positional encoding to 6. $\Phi^{\text{coarse}}(\mathbf{x})$ denotes that the feature grid Φ^{coarse} is tri-linearly interpolated at the point $\mathbf{x}$.

Fine-level Geometric Representation. While the coarse geometry can be obtained by our coarse-level shape representation, it is important to capture high-frequency geometric details in a scene. To realize it, we further model the high-frequency geometric details as residual SDF values with multi-resolution feature grids and an MLP decoder [5, 31, 76, 53]. Specifically, we employ multi-resolution dense feature grids $\{\Phi_l^{\text{fine}}\}_1^L$ with resolutions R_l. The resolutions are sampled in geometric space [31] to combine features at different frequencies:

$$R_l := \lfloor R_{\min} b^l \rfloor \quad b := \exp\left(\frac{\ln R_{\max} - \ln R_{\min}}{L-1}\right) \ , \quad (2)$$

where $R_{\min}, R_{\max}$ correspond to the lowest and highest resolution, respectively. Here we consider $R_{\min} = 32$, $R_{\max} = 128$, and in total $L = 8$ levels. The feature dimension is 4 for each level.

Now, to model the residual SDF values for a point $\mathbf{x}$, we extract and concatenate the tri-linearly interpolated features at each level, and input them to an MLP f^{fine} with 3 hidden layers of size 64:

$$(\Delta s, \ \mathbf{z}^{\text{fine}}) = f^{\text{fine}}\left(\gamma(\mathbf{x}), \ \{\Phi_l^{\text{fine}}(\mathbf{x})\}\right) \ , \quad (3)$$

where $\mathbf{z}^{\text{fine}} \in \mathbb{R}^{32}$ is the geometric feature for $\mathbf{x}$ at the fine level.

With the coarse-level base SDF value s^{coarse} and the fine-level residual SDF Δs, the final predicted SDF value $\hat{s}$ for $\mathbf{x}$ is simply the sum between two:

$$\hat{s} = s^{\text{coarse}} + \Delta s \ . \quad (4)$$

Color Representation. Besides 3D geometry, we also predict color values such that our mapping and camera tracking can be optimized also with color losses. Moreover, as an additional application, we can also render images from novel views on the fly. Inspired by [31], we encode the color using another multi-resolution feature grid $\{\Phi_l^{\text{color}}\}_1^L$ and a decoder f^{color} parameterized with a 2-layer MLP of size 64. The number of feature grid levels is now $L = 16$, the features dimension is 2 at each level. The minimum and maximum resolution now become $R_{\min} = 16$ and $R_{\max} = 2048$, respectively. We predict per-point color values as:

$$\hat{\mathbf{c}} = f^{\text{color}}\left(\mathbf{x}, \hat{\mathbf{n}}, \gamma(\mathbf{v}), \mathbf{z}^{\text{coarse}}, \mathbf{z}^{\text{fine}}, \{\Phi_l^{\text{color}}(\mathbf{x})\}\right) \ . \quad (5)$$

where $\hat{\mathbf{n}}$ corresponds to the normal at point $\mathbf{x}$ calculated from $\hat{s}$ in Eq. (4), and $\gamma(\mathbf{v})$ is the viewing direction with positional encoding with a level of 4, following [68, 71].

3.2. Volume Rendering

Following recent works on implicit-based 3D reconstruction [38, 68, 71, 59] and dense visual SLAM [51, 76], we optimize our scene representation from Sec. 3.1 using differentiable volume rendering. More specifically, in order to render a pixel, we cast a ray $\mathbf{r}$ from the camera center $\mathbf{o}$ through the pixel along its normalized view direction $\mathbf{v}$. N points are then sampled along the ray, denoted as $\mathbf{x}_i = \mathbf{o} + t_i\mathbf{v}$, and their predicted SDFs and color values are $\hat{s}_i$ and $\hat{\mathbf{c}}_i$, respectively. For volume rendering, we follow [68] to transform the SDFs $\hat{s}_i$ to density values σ_i:

$$\sigma_\beta(s) = \begin{cases} \frac{1}{2\beta}\exp\left(\frac{s}{\beta}\right) & \text{if } s \leq 0 \\ \frac{1}{\beta}\left(1 - \frac{1}{2}\exp\left(-\frac{s}{\beta}\right)\right) & \text{if } s > 0 \end{cases}, \tag{6}$$

where $\beta \in \mathbb{R}$ is a parameter controlling the transformation from SDF to volume density. As in [29], the color $\hat{C}$ for the current ray $\mathbf{r}$ is calculated as:

$$\hat{C} = \sum_{i=1}^{N} T_i\, \alpha_i\, \hat{\mathbf{c}}_i \qquad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \\ \alpha_i = 1 - \exp\left(-\sigma_i \delta_i\right) \quad , \tag{7}$$

where T_i and α_i correspond to transmittance and alpha value of sample point i along ray $\mathbf{r}$, respectively, and δ_i is the distance between neighboring sample points. In a similar manner, we can also compute the depth $\hat{D}$ and normal $\hat{N}$ of the surface intersecting the current ray $\mathbf{r}$ as:

$$\hat{D} = \sum_{i=1}^{N} T_i\, \alpha_i\, t_i \qquad \hat{N} = \sum_{i=1}^{N} T_i\, \alpha_i\, \hat{\mathbf{n}}_i \quad . \tag{8}$$

Locally Adaptive Transformation. The β parameter in Eq. (6) models the smoothing amount near the object's surface. The value of β gradually decreases during the optimization process as the network becomes more certain about the object surface. Therefore, this optimization scheme results in faster and sharper reconstructions.

In VolSDF [68], they model β as a single global parameter. This way of modeling the transformation intrinsically assumes that the degree of optimization is the same across different regions of the scene, which is sufficient for small object-level scenes. However, for our sequential input setting within a complicated indoor scene, a global optimizable β is sub-optimal (cf. ablation study in Sec. 4.2). Therefore, we propose to assign β values *locally* so the SDF-density transformation in Eq. (6) is also locally adaptive.

In detail, we maintain a voxel counter across the scene and count the number of point samples within every voxel during the mapping process. We empirically choose the voxel size of 64^3 (see ablation study in Sec. 4.2). Next, we heuristically design a transformation from local point samples counts T_p to the β value:

$$\beta = c_0 \cdot \exp(-c_1 \cdot T_p) + c_2 \quad . \tag{9}$$

We come up with the transformation by plotting β decreasing curve with respect to the voxel count under the global input setting, and fitting a function on the curve. We empirically find that the exponential curve is the best fit.

3.3. End-to-End Joint Mapping and Tracking

Purely from an RGB sequential input, it is very difficult to jointly optimize 3D scene geometry and color together with camera poses, due to the high degree of ambiguity, especially for large complex scenes with many textureless and sparsely covered regions. Therefore, to enable end-to-end joint mapping and tracking under our neural scene

representation, we propose to use the following loss functions, including geometric and prior constraints, single- and multi-view constraints, as well as both global and local constraints.

***Mapping*:** To optimize the scene representation mentioned in Sec. 3.1, we uniformly sample M pixels/rays in total from the current frame and selected keyframes. Next, we perform a 3-stage optimization similar to [76] but use the following loss:

$$\begin{aligned}\mathcal{L} = \mathcal{L}_{\text{rgb}} + 0.5\mathcal{L}_{\text{warp}} + 0.001\mathcal{L}_{\text{flow}} \\ + 0.1\mathcal{L}_{\text{depth}} + 0.05\mathcal{L}_{\text{normal}} + 0.1\mathcal{L}_{\text{eikonal}}\end{aligned} \quad (16)$$

At the first stage, we treat the coarse-level base SDF value s^{coarse} in Eq. (1) as the final SDF value $\hat{s}$, and optimize the coarse feature grid Φ^{coarse}, coarse MLP parameters of f^{coarse}, and color MLP parameters of f^{color} with Eq. (16). Next, after 25% of the total number of iterations, we start using Eq. (4) as the final SDF value so the fine-level feature grids $\{\Phi_l^{\text{fine}}\}$ and fine-level MLP f^{fine} are also jointly optimized. Finally, after 75% of the total number of iterations, we conduct a local bundle adjustment (BA) with Eq. (16), where we also include the optimization of color feature grids $\{\Phi_l^{\text{color}}\}$ as well as the extrinsic parameters of K selected mapping frames.

***Camera Tracking*:** We run in parallel camera tracking to optimize the camera pose (rotation and translation) of the current frame, while keeping the hierarchical scene representation fixed. Straightforwardly, we sample M_t pixels from the current frame and use purely the RGB rendering loss in Eq. (10) for 100 iterations.