SLAIM: Robust Dense Neural SLAM for Online Tracking and Mapping

Vincent Cartillier[1], Grant Schindler[2], Irfan Essa[1,2]
[1]Georgia Tech [2] Google Research
vcartillier3@gatech.edu
vincentcartillier.github.io/slaim.html

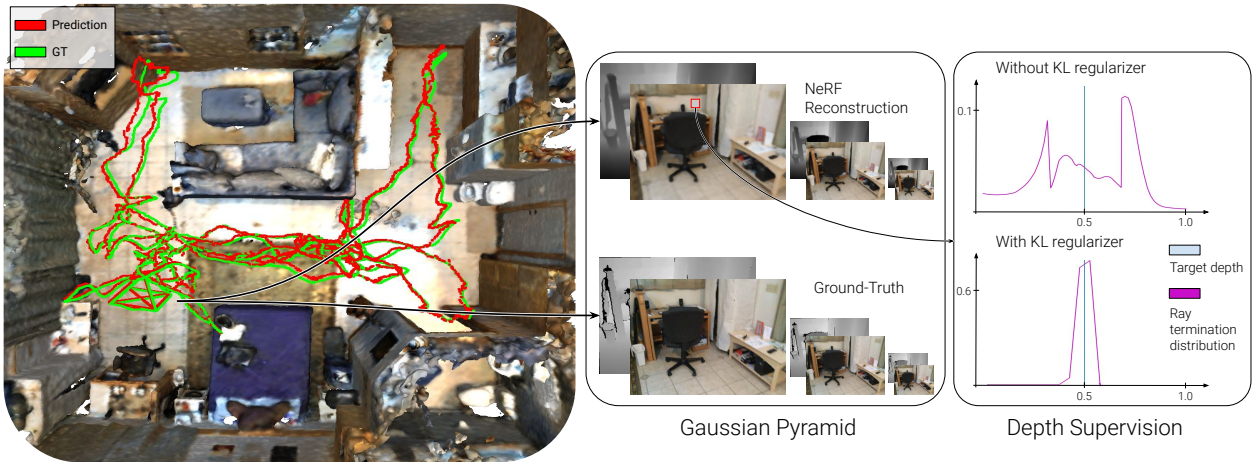

Figure 1. We present SLAIM, a robust dense neural RGB-D SLAM system that performs online tracking and mapping in real time. SLAIM implements a Gaussian Pyramid filter on top of NeRF to perform coarse-to-fine tracking and mapping. We also introduce a new target ray termination distribution that we use in a KL regularizer to supervise the network to converge towards the right geometry. SLAIM reaches state-of-the-art results in both tracking and 3D reconstruction accuracy.

Abstract

We present SLAIM - Simultaneous Localization and Implicit Mapping. We propose a novel coarse-to-fine tracking model tailored for Neural Radiance Field SLAM (NeRF-SLAM) to achieve state-of-the-art tracking performance. Notably, existing NeRF-SLAM systems consistently exhibit inferior tracking performance compared to traditional SLAM algorithms [21]. NeRF-SLAM methods solve camera tracking via image alignment and photometric bundle-adjustment. Such optimization processes are difficult to optimize due to the narrow basin of attraction of the optimization loss in image space (local minima) and the lack of initial correspondences. We mitigate these limitations by implementing a Gaussian pyramid filter on top of NeRF, facilitating a coarse-to-fine tracking optimization strategy. Furthermore, NeRF systems encounter challenges in converging to the right geometry with limited input views. While prior approaches use a Signed-Distance Function (SDF)-based NeRF and directly supervise SDF values by approximating ground truth SDF through depth measurements, this often results in suboptimal geometry. In contrast, our method employs a volume density representation and introduces a novel KL regularizer on the ray termination distribution, constraining scene geometry to consist of empty space and opaque surfaces. Our solution implements both local and global bundle-adjustment to produce a robust (coarse-to-fine) and accurate (KL regularizer) SLAM solution. We conduct experiments on multiple datasets (ScanNet, TUM, Replica) showing state-of-the-art results in tracking and in reconstruction accuracy.

1. Introduction

Dense visual Simultaneous Localization and Mapping (SLAM) is a long-standing problem in 3D computer vision with many applications, including autonomous driving, indoor and outdoor robotic navigation, virtual reality, and augmented reality. In this work, we present state-of-the-art tracking results using implicit maps to improve SLAM.

Traditional SLAM systems start by estimating image correspondences, which can be sparse, in the form of matched

keypoints [20] between frames, or dense [34] via estimation of optical flow for instance. These correspondences are then further used in a bundle adjustment process to predict and refine camera poses. The ability to find such correspondences is a large prior condition and assumption for traditional SLAM systems to work. This assumption may fail under some circumstances. In the sparse case, it is not always easy to detect keypoints – for instance texture-less surfaces are difficult to track. In the dense case, a deep pre-trained neural network is usually used, which limits the applications to scenes covering similar statistics as the ones used in the training set.

An emerging direction of research has presented new methods to solve SLAM using implicit maps (NeRFs), thus removing the requirement of correspondences in the pipeline. This new class of methods is best described as *NeRF-SLAM*. These NeRF-SLAM methods are interesting because they define SLAM as a full optimization problem without the need of any external pre-computed information (i.e., keypoints or a pre-trained network). The general idea is to (1) map past frames by building a NeRF model and (2) use the view synthesis capabilities of the NeRF model to track new frames via image alignment and photometric bundle-adjustment. iMAP [31] was the first implementation to demonstrate the feasibility of this approach. NICE-SLAM [41] improved the results by using a more accurate NeRF model producing better synthetic views. Both iMAP and NICE-SLAM represent the geometry as a volume density in NeRF [18].

More recently, ESLAM [12] and Co-SLAM [35] represent the geometry using a signed-distance function (SDF) and show better tracking and 3D reconstruction performances. Both solutions directly apply supervision to the SDF predicted values from NeRF which enforces the ray termination distribution to be unimodal and centered on the depth measurement. This leads to a more efficient and performant solution. However, the SDF supervision is done by computing an approximation of the ground-truth SDF values using the depth measurement. This approximation leads to sub-optimal geometry convergence.

In our proposed work, we keep the original volume density definition of NeRF but we apply a KL regularizer over the ray termination distribution to constraint them to be a narrow unimodal distribution. This effectively improves efficiencies and performances while not restricting the underlying geometry to a sub-optimal solution. In addition, ESLAM and Co-SLAM solve SLAM with local and global bundle-adjustment respectively. Instead we present a solution that combines the best of both worlds by performing local and global bundle adjustment.

A common observation on all previous NeRF-SLAM baselines is the gap in terms of tracking performances when compared to more traditional SLAM systems [20]. A key problem with these baselines is that any high spatial frequencies contained in accurate NeRF renderings actually make the image alignment optimization step difficult and inefficient to solve. The high frequencies induce a narrower basin of convergence during the image alignment process.

In this paper we present a new coarse-to-fine NeRF-SLAM pipeline to overcome this image alignment problem, achieving state-of-the-art results. The core idea is to use a Gaussian filter on the output image signal to enlarge the basin of attraction during image alignment optimization and photometric bundle-adjustment, making the tracking more robust and efficient. This is a similar idea to the classical hierarchical Lucas-Kanade optical flow algorithm [17] that solves optical flow using image pyramids [2]. In the NeRF-SLAM context, applying this central idea requires overcoming key technical hurdles that we present below.

Overall our contributions are as follows:

- We present SLAIM, a robust, NeRF-SLAM system using implicit mapping and a coarse-to-fine improved tracking.
- We present a new KL regularizer over the ray termination distribution which leads to better tracking and mapping.
- We present a new NeRF-SLAM pipeline that performs both local and global bundle-adjustment.
- We report state-of-the-art results on camera pose predictions and 3D reconstructions on several datasets.

2. Related Work

2.1. Visual SLAM

The goal of visual SLAM is to estimate camera poses along with a global 3D reconstruction given a stream of input frames [4]. Most solutions to this problem implement a mapping and tracking module working in parallel [14]. Traditionally, tracking is solved by iteratively finding correspondences between frames and updating the poses accordingly [5, 20, 21, 26, 34]. These correspondences can be found by image keypoints matching [5, 20, 21, 26]. More recently Droid-SLAM [34] estimates correspondences from dense deep correlation of features. All of the previous work are indirect SLAM methods and performances rely on good keypoints features [5, 20, 21, 26] or pre-trained networks [34]. Our proposed pipeline, a direct SLAM solution, does not require any pretraining or pre-processing.

We focus on dense reconstruction mapping (i.e., dense-SLAM) where a dense 3D map is maintained during the entire tracking process [22, 23]. Prior work used a fixed or a hierarchy of resolutions [8, 22, 28, 33]. These approaches are memory inefficient and are limited in the level of details represented. Other works [6, 9, 25] use training methods to improve the 3D reconstruction. However, these methods require extra training data that is not available in our problem. Closer to our work we find approaches that store information in a world-centric map representation using surfels [28, 38] or voxels [3, 8, 13, 22, 41]. Our work stores implicit latent

features in a hashmap of voxels.

2.2. Neural Radiance Fields (NeRFs)

Scene representations and graphics have recently made substantial progress by using a neural network MLP combined with volumetric rendering to allow for novel view synthesis [18]. NeRF [18] has triggered a large number of subsequent papers that proposed improvements to the initial model. [11, 19, 32, 41] drastically improved the training time by using a smaller MLP combined with a set of optimizable spatial grid features. Inspired by this performance, we build our work upon Instant-NGP [19] which uses a hash-map of hierarchical grid representations of the scene allowing real-time training. In another line of work, we find papers that alleviate the dependency of NeRF on having known camera poses. [37] adds camera poses as parameters in the overall optimization pipeline. The overall process is very slow.

BARF [16] shares a similar idea to our work as it solves bundle adjustment using a coarse-to-fine image alignment optimization setup. The system modulates the NeRF's positional encoding using a low-pass filter which effectively reduces high frequencies and widens the basin of convergence. Since we are using grid features as positional encoding we cannot apply the same technique. Moreover, BARF assumes all frames with initial noisy poses given at the beginning of the optimization which is not compatible with an on-line SLAM formulation. In terms of scene representation, the initial NeRF [18] characterizes scene geometry through volume density. Alternatively, some researchers have explored the use of a signed-distance function (SDF) to encode geometry [1, 24, 36, 39, 40]. While this approach can allow direct supervision over SDF values [12, 35], it necessitates intricate techniques for unbiased conversion of SDF to ray termination probability [36]. In our investigation, we encounter DS-NeRF [10], which uses the original density definition and imposes a Gaussian KL regularization on the ray termination distribution to ensure unimodality. In our work we apply regularization differently by modeling a skewed Gaussian-like distribution which is shown to be more accurate.

2.3. SLAM with NeRF

In the NeRF-SLAM domain we start by finding [27] that combines Droid-SLAM [34] and Instant-NGP [19]. However, [27] solves SLAM without NeRF. Closer to our work, we find SLAM implementations using implicit mapping, including ESLAM, Co-SLAM and others [12, 15, 31, 35, 41]. ESLAM and Co-SLAM are using an SDF-based NeRF and add supervision over the SDF values directly. Supervision is done via computing an approximate ground-truth SDF value given a ray depth measurement. This approach provides improved control over scene geometry and accelerates convergence. Nevertheless, this approximation may result in suboptimal geometry. In contrast, our proposed approach incorporates a KL regularizer on the ray termination distribution, achieving an optimal and rapid convergence. None of the previous techniques study coarse-to-fine approaches.

3. Methodology

We provide an overview of our method in Fig. 2. In this section we detail SLAIM, a novel approach for dense mapping and tracking of an RGB-D input $\{I_t\}_{t=1}^{N}$ stream using known camera intrinsics $K \in \mathbf{R}^{3\times3}$ and a neural scene representation f_Ψ. The camera poses $\{P_t\}_{t=1}^{N}$ and implicit scene representation are jointly optimized in a coarse-to-fine manner.

3.1. NeRF pre-requisites

The geometry is encoded in a multiresolution hash-based feature grid [19], with parameters β, that maps a 3D input coordinate $x \in \mathbf{R}^3$ into a dense feature vector $y = h_\beta^L(x)$. The feature space is divided into L grids of resolution ranging from R_{min} to R_{max}. Feature vectors at each level $l \in [1, L]$, $h_\beta^l(x)$ are queried via trilinear interpolation and further concatenated to form the final positional encoding $y = \left[h_\beta^1(x), ..., h_\beta^L(x)\right]$. We further employ two shallow MLP decoders to estimate the density and color at the given 3D input location. Specifically, the geometry decoder, with parameters τ, outputs a density value σ and a feature vector g. The color decoder, with parameters ψ, takes as input the feature vector g and predicts RGB values c.

$$f_\tau^g(y) \mapsto (\sigma, g); \quad f_\psi^c(g) \mapsto c \tag{1}$$

Note that in traditional NeRF systems [18] the color decoder also takes as input the direction embedding. Here we intentionally remove that input to reduce the optimization complexity. This choice is driven by our focus on camera tracking and mapping, with no intention of modeling specularities. During mapping we optimize the set of NeRF parameters $\Psi = \{\beta, \tau, \psi\}$ along with the camera poses parameters. Following [18, 19] we render color and depth pixels by alpha compositing the values along a ray. Specifically, given the camera origin $\vec{o} \in \mathbf{R}^3$ and ray direction $\vec{r} \in \mathbf{R}^3$, we sample M points $x_i = \vec{o} + d_i.\vec{r}$, with $i \in \{1, ..., M\}$ and depths values d_i. We employ the importance sampling strategy implemented in [19] to sample points along a ray. We bound the scene to the unit cube $[0, 1]^3$ and we start by uniformly sampling with a fixed ray marching step size of $\Delta_r = \sqrt{3}/1024$. In addition, we maintain an occupancy grid to avoid sampling points in free spaces, and we stop sampling once a surface is found. Similar to [18], we model the ray termination distribution $w(r)$ as follow:

$$w(r) = T(r)\sigma(r) \tag{2}$$

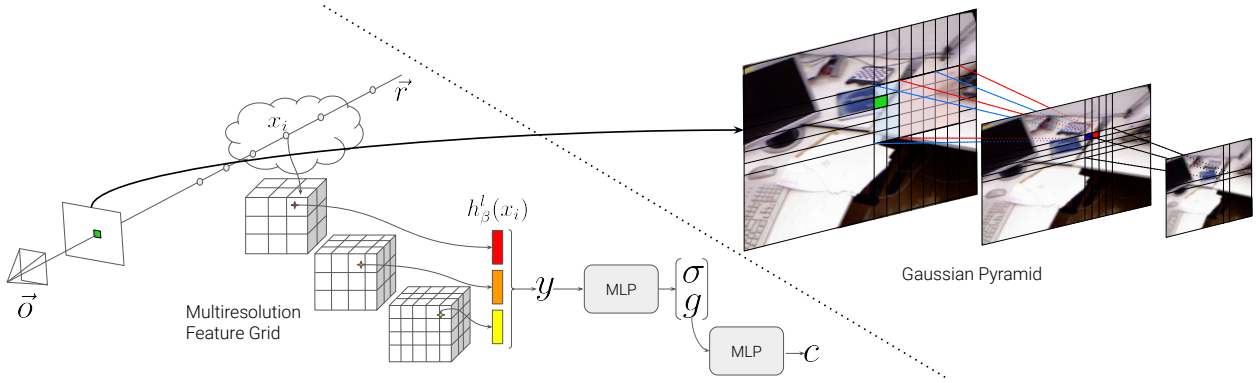

Figure 2. **Overview of SLAIM.** For a given input ray with center $\vec{o}$ and direction $\vec{r}$ we start by rendering its corresponding color pixel via ray tracing and volume rendering. For each sample x_i along the ray we query the multiresolution feature grid to form an input embedding y. We then make successive calls to two shallow MLP networks to predict a density σ and color c for that sample. After reconstructing the image we apply a Gaussian Pyramid filter to perform coarse-to-fine tracking and mapping.

with $T(r) = exp(-\int_0^r \sigma(s)ds)$. We approximate this integral using a sampling-based method and express the discretized w_i at point x_i as $w_i = \alpha_i.T_i$ with $\alpha_i = (1-e^{-\sigma_i\delta_t})$ and $T_i = \prod_{j=1}^{i-1}(1-\alpha_i)$. The final pixel color and depth values are computed as follows:

$$\hat{c} = \sum_{i=1}^{M} w_i.c_i; \quad \hat{d} = \sum_{i=1}^{M} w_i.d_i \tag{3}$$

3.2. Depth Supervision

Using depth supervision is crucial for a NeRF-SLAM system to produce the right geometry. Using a loss (e.g L1) over the depth values directly can lead to reconstruction artifacts in regions with only limited views. This is because the volumetric rendering does not adhere to the restriction that the majority of the geometry consists of empty space and opaque surfaces [10]. In other words, the ray termination distribution (i.e. the probability of a ray to hit an obstacle) should be unimodal and centered on the depth measurement. We implement a new KL regularizer over these ray termination distributions to enforce them to be unimodal. The ideal distribution is a delta function centered at the depth measurement d: $\delta(r-d)$. DSNeRF [10] approximates this distribution with a narrow Gaussian distribution. Nevertheless, we observe that the distribution arising from Eq. 2 closely resembles a skewed normal distribution, as illustrated in Fig. 3. This resemblance is attributed to the fact that a depth measurement implies a spike, or a narrow bell-shaped, *density* response $\sigma(r)$ centered at the depth. Consequently, Eq. 2 yields a skewed Gaussian-like ray termination distribution. Therefore, we approximate an ideal density function $\tilde{\sigma}$ and compute the ray termination distribution $\tilde{w}$ using Eq. 2. We define $\tilde{\sigma}(r) = S \times sech^2(\frac{(r-d')}{\sigma_d})$ with d' and σ_d the mean and variance and S a scale factor. This function is a narrow bell-shape function centered on d'. We choose the $sech^2$ function instead of a Gaussian for its mathematical simplicity when computing integrals. We set the d' parameter such that the expected depth $\hat{d}$ matches the depth measurement d, $E_x(\tilde{w}) = d$. The final KL regularizer function is expressed as follows:

$$\mathcal{L}_{KL} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k} log(w(r_k))\tilde{w}(r_k)\Delta_r \tag{4}$$

We show in the supplement how to compute the expectation and how to derive such KL loss.

3.3. Coarse-to-fine Tracking and Mapping.

Coarse-to-fine formulation. Our system, similar to previous ones [31, 35, 41], tracks cameras via image alignment and photometric bundle adjustment. Achieving good camera pose estimations given this approach can be difficult due to the narrow basin of attraction [16, 23] of the optimization function (i.e., there are many local minima in image space) and the lack of initial correspondences. To overcome this problem we use a coarse-to-fine approach using a Gaussian Pyramid filter [2] to effectively smooth the input signal in the early iterations in order to widen the basin of attraction and thus avoid getting stuck in local minima.

To render an image at a level l_G in the Gaussian Pyramid we perform multiple reduce operations [2] which consists of applying a convolution filter with a 5-tap kernel $w_g = [1,4,6,4,1] \times [1,4,6,4,1]^T/256$ and downsampling the results by selecting every two pixels. We apply a similar coarse-to-fine approach on the depth frame, but instead of a 5-tap kernel we apply a median filter. We apply this Gaussian pyramid filter on both reconstructed and ground-truth images. Reconstructing an entire image using NeRF during optimization would be impractical due to memory and time complexity constraints. Instead we sample a set of pixels at the Gaussian Pyramid level l_G and compute the

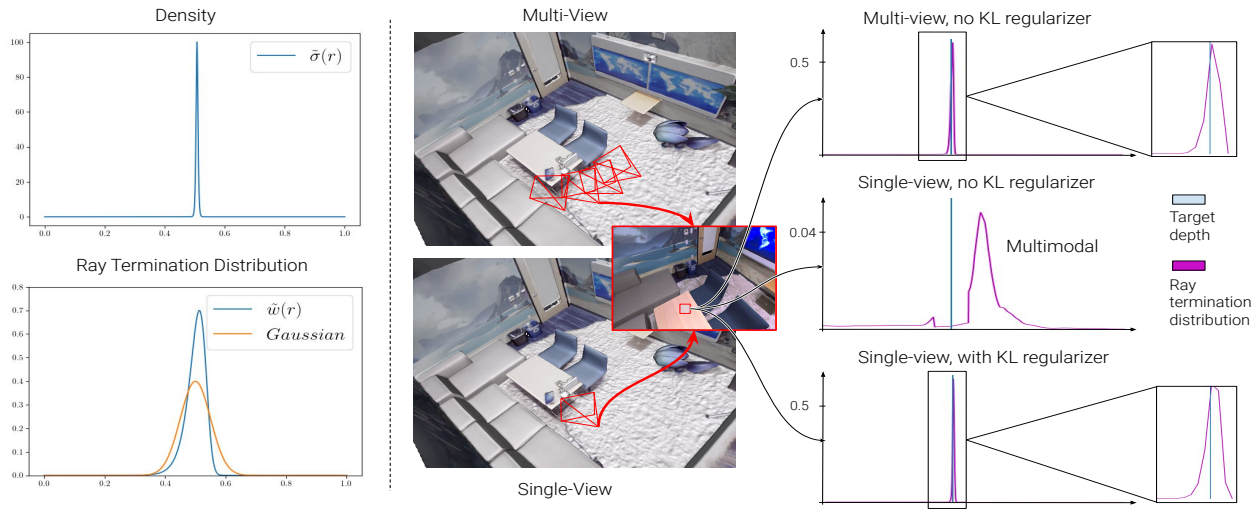

Figure 3. Illustration of the custom ray termination distribution. (left-top) shows the estimation of the density response $\tilde{\sigma} \sim sech^2$ as a narrow bell-shape function. (left-bottom) compares the resulting ray termination distribution $\tilde{w}$ computed from $\tilde{\sigma}$ to the Gaussian distribution described in DS-NeRF [10]. On the right, we display results from an experiment showcasing the ray termination distribution' shape under different conditions. (middle) compares mapping setups: one with multiple training views (over-constrained geometry) and another with only one view (under-constrained geometry). (right-top) shows multi-view training without KL regularization (right-middle) depicts single-view training without KL regularization, and (right-bottom) exhibits single-view training with our custom regularizer. We observe similarity between the ray termination distributions in the (right-top) and (right-bottom), supporting the use of our custom distribution.

corresponding receptive field on the original image. We then restrict the NeRF pixel reconstruction to the pixels within the receptive field. We can derive the steepest descent image formulation of the smooth image $I^{(l_G)}$ after applying the Gaussian Pyramid filter with l_G levels.

$$J(I^{(l_G)}, P) = \prod_{k=1}^{l_G} \frac{\partial I^k}{\partial I^{k-1}} \frac{\partial I^0}{\partial P} \tag{5}$$

Where $\partial I^0 / \partial P$ is the partial derivative of the full resolution image given the camera pose parameters P, and $\partial I^k / \partial I^{k-1}$ is the Jacobian matrix between images at levels k and $k-1$ in the Gaussian Pyramid. These matrices are populated with the kernel weights w_g. From this formulation this is akin to applying a weighted average on image gradients $\partial I^0 / \partial P$. This has the effect to smooth and cohere the gradients for a more robust optimization.

Camera Tracking. Similar to previous works [31, 35, 41] our tracking strategy is defined as an image alignment problem between the current frame and the underlying implicit map. Optimization is performed via minimizing a photometric and geometrical loss along with our KL regularizer.

$$\mathcal{L} = \mathcal{L}_{rgb} + \lambda_d \mathcal{L}_d + \lambda_{KL} \mathcal{L}_{KL} \tag{6}$$

The color loss $\mathcal{L}_{rgb}$ is an l_2 loss over the RGB pixel values and the depth loss $\mathcal{L}_d$ is an l_1 loss over depth values.

$$\mathcal{L}_{rgb} = \frac{1}{N} \sum_{n=1}^{N} (\hat{c}_n - c_n)^2; \mathcal{L}_d = \frac{1}{|\mathcal{I}_d|} \sum_{r \in \mathcal{I}_d} |\hat{d}_r - d_r| \tag{7}$$

Where $\mathcal{I}_d$ is the subset of sampled rays with a valid depth measurement. Using the defined loss we track the camera-to-world pose matrix $P_t \in SE(3)$ for each timestep t. We initialize the pose of the current frame using constant speed assumption $P_t = P_{t-1} P_{t-2}^{-1} P_{t-1}$. The tracking optimization consists of doing multiple iterations of selecting $N = N_t$ pixels within the current frame and optimizing the pose by minimizing the tracking loss via stochastic gradient descent.

Mapping. We continuously optimize the scene representation with a set of selected keyframes. Similarly to [35] we select keyframes at fixed intervals. Given the set of growing keyframes $\mathcal{K}$ we jointly optimize the scene parameters Ψ and camera poses $P_k, k \in \mathcal{K}$ using the same loss $\mathcal{L}$ from Eq. 6. The joint optimization occurs through alternating steps. In each iteration, we optimize the scene representation Ψ, however the gradients of camera poses are accumulated over k_p iterations before updating these parameters. Mapping is performed every X frames and that process is split into two phases. We start by running local bundle adjustment over the $M_{\mathcal{K}} - 1$ most recent keyframes and the current frame, and then continue by running additional global bundle adjustment iterations over all keyframes $\mathcal{K}$. We observed that when only doing global bundle adjustment, the quality of image reconstruction decreases as the number of keyframe

increases. This is because the number of sampled rays per frame then reduces. Adding, local bundle adjustment helps maintaining a good reconstruction quality throughout the video which is crucial for tracking performances.

3.4. Implementation details

The proposed approach follows a two-fold process of tracking and mapping. To initialize the system, a few mapping iterations are executed on the first frame. Subsequently, for each new frame, we run the tracking and mapping processes in a sequential manner. The entire algorithm is built upon Instant-NGP [19] and is written in C++ and CUDA kernels leading to fast computation. We conduct experiments on a single NVIDIA A40 GPU and 2.35GHz AMD EPYC 7452 32-Core CPU. For experiments with default settings (Ours) we sample $N_t = 1024$ and $N_m = 2048$ pixels for tracking and mapping over 12 and 30 iterations respectively. Refer to the supplementary material for additional details.

4. Experiments

4.1. Experimental setup.

Datasets. We evaluate SLAIM on different scenes from three different datasets. Similar to previous work, we compare reconstruction performance on 8 synthetic scenes of the Replica dataset [29]. We evaluate the tracking performance of SLAIM on 6 scenes of the ScanNet [7] dataset and 3 scenes from TUM-RGBD [30].

Metrics. Following previous baselines [12, 35, 41], we measure the reconstruction quality on observed regions (within camera FoV) using Accuracy (cm), Completion (cm), Depth L1 (cm), and Completion ratio (%) with a threshold of 5cm. We evaluate camera tracking using absolute trajectory error (ATE) RMSE [30] (cm).

Baselines. We consider iMAP [31], NICE-SLAM [41], Co-SLAM [35] and ESLAM [12] as baselines for comparison of reconstruction quality and camera tracking. We also compare to another version of SLAIM, that we refer to as $SLAIM_{MG}$ (MG stands for max-grid), which renders images using different grid-resolution features to perform coarse-to-fine rendering instead of using a Gaussian Pyramid. Specifically, we use a max-grid-level parameter $mgl \leq L$ that we use to set a max resolution the network is allowed to use to construct the position embedding y. Grid features for higher levels are set to 0, and the final positional encoding can be written as $y = \left[h_\beta^1(x), ..., h_\beta^{mgl}(x), 0\right]$. This is a very simple trick technique to effectively render blurry images. See supplementary material for visualizations.

Coarse-to-fine settings. In all of our experiments we apply the coarse-to-fine strategy to both tracking and mapping. We set an initial Gaussian Pyramid level (GPL) and gradually reduce that GPL throughout a mapping or tracking phase. For a given number of iterations and initial GPL, we split evenly the number of iterations per pyramid levels. For example, in Replica we run 20 mapping iterations with an initial GPL of 1. This means we run 10 iterations at levels 1 and 0 respectively (level 0 means no blurring).

4.2. Tracking and Reconstruction performance.

Replica dataset. We evaluate 3D reconstruction performance on the same simulated RGB-D sequences as iMAP. We report numbers in Tab. 1 and qualitative examples in Fig. 4. Prior to running the evaluation we follow the implementation in Co-SLAM [35] and perform mesh culling to remove unobserved regions outside of any camera frustum. As shown in Tab. 1, our method achieves the best results in terms of accuracy with an improvement of close to 5% compared to Co-SLAM [35] and ESLAM [12]. On the Depth-L1 and completion metrics, SLAIM performs second best. SLAIM struggles in estimating completely unobserved regions. This is not surprising given that NeRF cannot hallucinate large unobserved regions. Mesh holes in missing regions will lead to large Depth-L1 and completion scores. We additionally evaluate against two baselines: $SLAIM_{noKL}$, which omits the KL regularizer, and $SLAIM_G$, leveraging a Gaussian KL regularizer, as per DSNerf [10]. We observe that $SLAIM_{noKL}$ performs worse than NICE-SLAM [41]. While both approaches employ a density-based NeRF with multi-resolution feature grids, NICE-SLAM utilizes pretrained decoders that incorporate learned geometrical priors. In contrast, our method is trained from scratch. Consequently, NICE-SLAM achieves superior reconstruction results. Comparing SLAIM to $SLAIM_G$, we consistently observe performance improvement by employing our custom ray termination distribution instead of a Gaussian. This suggests that $\tilde{w}(r)$ helps at better represent the geometry.

	Depth L1(cm) ↓	Acc. (cm) ↓	Comp. (cm) ↓	Comp. ratio (cm) ↑
iMAP[31]	4.64	3.62	4.93	80.51
NICE-SLAM [41]	1.90	2.37	2.64	91.13
Co-SLAM [35]	1.51	2.10	2.08	93.44
ESLAM [12]	**0.94**	2.18	**1.75**	**96.46**
$SLAIM_{noKL}$	2.13	2.45	2.78	89.17
$SLAIM_G$	1.46	2.11	1.98	94.87
$SLAIM$ (*Ours*)	1.37	**1.82**	1.90	95.61

Table 1. Reconstruction results on the Replica dataset [29]. SLAIM is best on accuracy and second best on all other metrics.

ScanNet dataset. We evaluate the camera tracking accuracy of SLAIM on 6 real-world sequences from ScanNet [7]. We compute the absolute trajectory error (ATE) between the aligned prediction and ground-truth trajectories. Tab. 2 shows that quantitatively, our method achieves the best tracking results across the board on all scenes with an increase of close to 15% on average.

TUM dataset. We also evaluate the tracking performances on the TUM dataset [30]. As depicted in Tab. 3, our

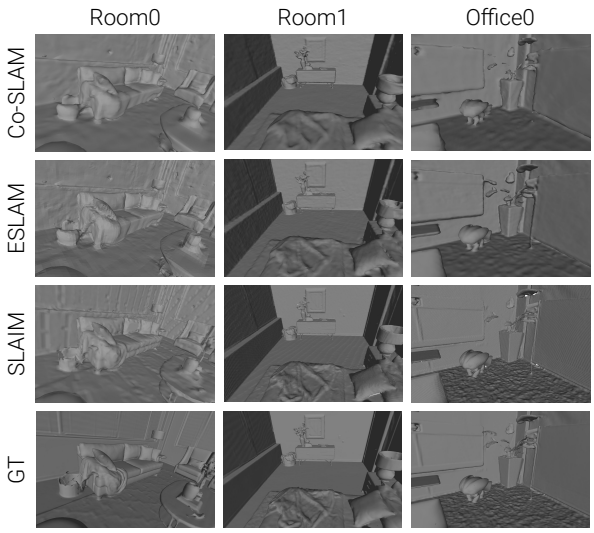

Figure 4. Reconstruction results on Replica [29]. Our method can retrieve thin structures where other baselines tend to oversmooth the geometry.

scene-ID	0000	0059	0106	0169	0181	0207	Avg.
iMAP*[31]	55.95	32.06	17.50	70.51	32.10	11.91	36.67
NICE-SLAM [41]	8.64	12.25	8.09	10.28	12.93	5.59	9.63
Co-SLAM [35]	7.13	11.14	9.36	5.90	11.81	7.14	8.75
ESLAM [12]	7.32	8.55	7.51	6.51	9.21	5.71	7.42
SLAIM (*Ours*)	**4.56**	**6.12**	**6.9**	**5.82**	**8.88**	**5.69**	**6.32**

Table 2. Tracking results, ATE RMSE(cm) ↓, on the ScanNet dataset [7]. Our method gets the best tracking performances across the board.

method achieves best tracking results on two of the three scenes, and second best on the third one. We notice that across the board our method outperforms the $SLAIM_{MG}$ baseline which highlights the benefits of performing proper coarse-to-fine with a Gaussian Pyramid filter compared to rendering views with different grid feature resolution. The coarse views rendered by $SLAIM_{MG}$ look blurry at first glance but have artifacts caused by the MLP decoders trying to over-compensate the lack of high-resolution features (see visualizations in supplementary material), which weakens the training signal during the early tracking iterations.

4.3. Ablation studies.

Effect of the coarse-to-fine strategy on Mapping and Tracking. We report tracking accuracy (ATE) numbers on the ScanNet dataset [7] in Tab. 4 showing a significant performance increase when using the coarse-to-fine strategy on both mapping and tracking. We compare to a baseline where no coarse-to-fine is applied ($SLAIM_{noc2f}$) and observe an 8% increase in performances when using coarse-to-fine (lines 1 and 5). The $SLAIM_{track}$ and $SLAIM_{map}$ are baselines where we run the coarse-to-fine strategy on the tracker or mapper only. We notice that coarse-to-fine tracking has a bigger impact than coarse-to-fine mapping (lines 2-3).

	fr1/desk	fr2/xyz	fr3/office
iMAP[31]	4.9	2.0	5.8
NICE-SLAM [41]	2.7	1.8	3.0
Co-SLAM [35]	2.4	1.7	2.4
ESLAM [12]	2.5	**1.1**	2.4
$SLAIM(Ours)$	**2.1**	1.5	**2.3**
$SLAIM_{MG}$	2.5	1.8	2.5
ORB-SLAM2 [20]	1.6	0.4	1.0

Table 3. RMSE ATE(cm) ↓ tracking results on the TUM-RGBD dataset [30]. SLAIM gets the best performances on two of the three scenes and second best on the third scene.

	Map. c2f	Track. c2f	GPL	AVG ATE(cm)
$SLAIM_{noc2f}$			N/A	6.91
$SLAIM_{track}$		✓	2	6.56
$SLAIM_{map}$	✓		2	6.90
$SLAIM$	✓	✓	1	6.44
$SLAIM$	✓	✓	2	**6.32**
$SLAIM$	✓	✓	3	6.47

Table 4. Results on ScanNet [7] showing the impact of the coarse-to-fine (c2f) strategy along with the number of Gaussian Pyramid levels. We observe best performances when doing c2f with a level-2 Gaussian Pyramid compared to no c2f ($noc2f$) and baselines that only do c2f during tracking ($track$) and mapping (map).

Impact of the number of levels in the Gaussian Pyramid We compare the tracking accuracy on the ScanNet dataset in Tab. 4 when testing with different GPL. From Tab. 4 we observe that the optimal GPL is 2 (lines 4-6). Increasing the GPL to 3 decreases the performances. We explain this phenomenon by the fact that the receptive field of the Gaussian Pyramid increases as the GPL increases. Therefore, with a fixed set of rendered rays per batch, the number of pixels at the GPL on which the loss is computed will reduce as the GPL increases.

Geometry is important. Tab. 5 reports ablation tracking accuracy results on the ScanNet [7] dataset. We observe that our approach with the custom KL regularizer (line 3) yields the best results with an increase of close to 5% in average ATE compared to $SLAIM_G$ which uses the Gaussian regularizer implemented in DSNerf [10]. We also observe that the results drop drastically when not using any KL regularizer ($SLAIM_{noKL}$ - line 5).

Local and global bundle adjustments. From Tab. 5 we measure the impact of the local and global bundle-adjustment. We observe that we obtain the best results when combining both global and local bundle-adjustment (line 3). We notice a large drop in performances when doing local (LBA) or global (GBA) bundle-adjustment only (lines 1,2).

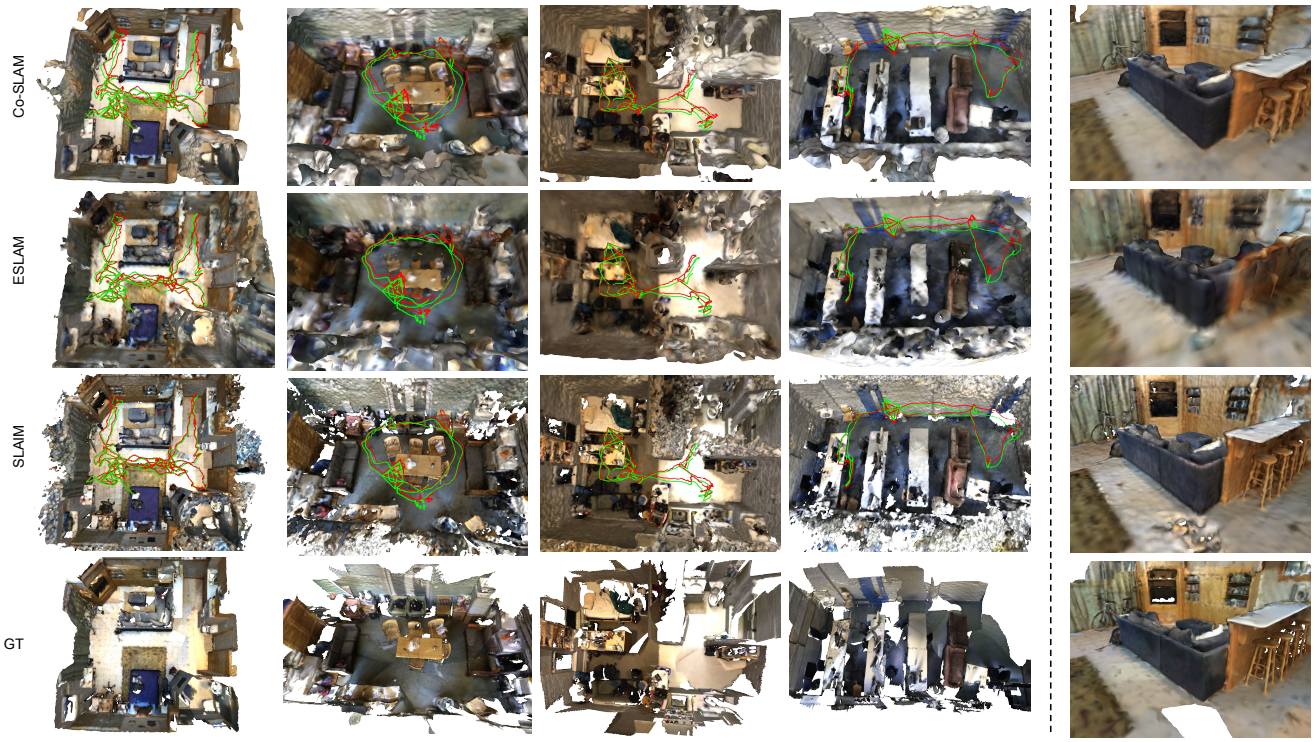

Figure 5. SLAIM qualitative results on the ScanNet dataset [7]. The ground truth camera trajectory is shown in green, and the estimated one in red. In comparison to other baselines [12, 35] our method achieves more accurate tracking results and produces finer 3D reconstruction (right column).

	BA	KL reg.	AVG ATE(cm)
LBA	L	*Ours*	9.45
GBA	G	*Ours*	11.32
$SLAIM$	G+L	*Ours*	**6.32**
$SLAIM_G$	G+L	Gaussian	6.69
$SLAIM_{noKL}$	G+L	N/A	9.21

Table 5. Ablation study showing the impact of local (L) and global (G) bundle adjustment (BA) and the choice of the KL regularizer. We get the best performances when doing both global and local bundle-adjustment and using our custom KL regularizer.

		Track. (ms) ↓	Map. (ms) ↓	FPS ↑	#Params ↓
Replica	iMAP	21.1×6	46.1×10	4.5	0.26 M
	NICE-SLAM	7.9×10	91.2×60	0.98	17.4 M
	Co-SLAM	5.9×10	10.1×10	12.6	0.26 M
	ESLAM	7.5×8	19.9×15	6.4	9.29 M
	$SLAIM$ (Ours)	7.4×10	9.8×20	8.8	0.25 M
ScanNet	iMAP	30.4×50	44.9×300	0.37	0.2 M
	NICE-SLAM	12.3×50	125.3×60	0.68	10.3 M
	Co-SLAM	7.8×10	20.2×10	8.4	0.8 M
	ESLAM	7.4×30	22.4×30	2.81	10.5 M
	$SLAIM$ (Ours)	8.4×15	$23.1 \times (15 + 15)$	3.8	0.8 M

Table 6. Run time and memory footprint comparison across baselines on Replica and ScanNet datasets. Track.(ms) and Map. (ms) are the tracking and mapping time shown as the time per iteration multiplied by the number of iterations. For SLAIM the number of mapping iterations comprises the local and global bundle-adjustment iterations (we do only global BA on Replica).

4.4. Run time comparison.

Tab. 6 reports the run times and memory footprint of the different baselines on Replica and ScanNet. Our method has similar FPS compared to ESLAM while using significantly less memory - close to ten times less. In addition, SLAIM presents better tracking and 3D reconstruction results compared to Co-SLAM at the price of a lower FPS.

5. Conclusion

We introduced SLAIM, a dense real-time RGB-D NeRF-SLAM system with state-of-the-art camera tracking and mapping. We show that using coarse-to-fine tracking and photometric bundle-adjustment along with proper depth supervision, SLAIM achieves both accurate camera tracking and high-quality 3D reconstruction while staying memory efficient. We show from extensive experiments, comparisons, and ablations that our Gaussian Pyramid filter and our new KL regularizer lead to SOTA results in camera tracking and 3D reconstruction.

References

[1] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6290–6301, 2022. 3

[2] Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pages 671–679. Elsevier, 1987. 2, 4

[3] Erik Bylow, Jürgen Sturm, Christian Kerl, Fredrik Kahl, and Daniel Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *Robotics: Science and Systems*, page 2, 2013. 2

[4] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016. 2

[5] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 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 *European Conference on Computer Vision*, pages 608–625. Springer, 2020. 2

[7] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 6, 7, 8

[8] 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)*, 36(4):1, 2017. 2

[9] Angela Dai, Christian Diller, and Matthias Nießner. Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2020. 2

[10] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022. 3, 4, 5, 6, 7

[11] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 3

[12] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hybrid representation of signed distance fields. *arXiv preprint arXiv:2211.11704*, 2022. 2, 3, 6, 7, 8

[13] Olaf Kähler, Victor A Prisacariu, and David W Murray. Real-time large-scale dense 3d reconstruction with loop closure. In *European Conference on Computer Vision*, pages 500–516. Springer, 2016. 2

[14] Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 83–86. IEEE, 2009. 2

[15] Xin Kong, Shikun Liu, Marwan Taher, and Andrew J Davison. vmap: Vectorised object mapping for neural field slam. *arXiv preprint arXiv:2302.01838*, 2023. 3

[16] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5741–5751, 2021. 3, 4

[17] Bruce D Lucas, Takeo Kanade, et al. *An iterative image registration technique with an application to stereo vision*. Vancouver, 1981. 2

[18] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2, 3

[19] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 3, 6

[20] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017. 2, 7

[21] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015. 1, 2

[22] 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 *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. Ieee, 2011. 2

[23] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011. 2, 4

[24] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. Stylesdf: High-resolution 3d-consistent image and geometry generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13503–13513, 2022. 3

[25] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020. 2

[26] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696. IEEE, 2020. 2

[27] Antoni Rosinol, John J Leonard, and Luca Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. *arXiv preprint arXiv:2210.13641*, 2022. 3

[28] Thomas Schops, Torsten Sattler, and Marc Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 134–144, 2019. 2

[29] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 6, 7

[30] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE, 2012. 6, 7

[31] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6229–6238, 2021. 2, 3, 4, 5, 6, 7

[32] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11358–11367, 2021. 3

[33] Danhang Tang, Saurabh Singh, Philip A Chou, Christian Hane, Mingsong Dou, Sean Fanello, Jonathan Taylor, Philip Davidson, Onur G Guleryuz, Yinda Zhang, et al. Deep implicit volume compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1293–1303, 2020. 2

[34] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in Neural Information Processing Systems*, 34:16558–16569, 2021. 2, 3

[35] Hengyi Wang, Jingwen Wang, and Lourdes Agapito. Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam. *arXiv preprint arXiv:2304.14377*, 2023. 2, 3, 4, 5, 6, 7, 8

[36] 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. *arXiv preprint arXiv:2106.10689*, 2021. 3

[37] 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. 3

[38] Thomas Whelan, Stefan Leutenegger, Renato Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015. 2

[39] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021. 3

[40] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P Srinivasan, Richard Szeliski, Jonathan T Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. *arXiv preprint arXiv:2302.14859*, 2023. 3

[41] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12786–12796, 2022. 2, 3, 4, 5, 6, 7