Point-SLAM: Dense Neural Point Cloud-based SLAM

Erik Sandström[1]* Yue Li[1]* Luc Van Gool[1,2] Martin R. Oswald[1,3]
[1]ETH Zürich, Switzerland [2]KU Leuven, Belgium [3]University of Amsterdam, Netherlands

Abstract

We propose a dense neural simultaneous localization and mapping (SLAM) approach for monocular RGBD input which anchors the features of a neural scene representation in a point cloud that is iteratively generated in an input-dependent data-driven manner. We demonstrate that both tracking and mapping can be performed with the same point-based neural scene representation by minimizing an RGBD-based re-rendering loss. In contrast to recent dense neural SLAM methods which anchor the scene features in a sparse grid, our point-based approach allows dynamically adapting the anchor point density to the information density of the input. This strategy reduces runtime and memory usage in regions with fewer details and dedicates higher point density to resolve fine details. Our approach performs either better or competitive to existing dense neural RGBD SLAM methods in tracking, mapping and rendering accuracy on the Replica, TUM-RGBD and ScanNet datasets. The source code is available at https://github.com/eriksandstroem/Point-SLAM.

1. Introduction

Dense visual simultaneous localization and mapping (SLAM) is a long-standing problem in computer vision where dense maps have widespread applications in augmented and virtual reality (AR, VR), robot navigation and planning tasks [17], collision detection [7], detailed occlusion reasoning [46], and interpretation [72] of scene content which is vital for scene understanding and perception.

To estimate a dense map via SLAM, tracking and mapping steps have traditionally been employed with different scene representations which creates undesirable data redundancy and independence since the tracking is then often performed independently of the estimated dense map. Camera **tracking** is frequently done with sparse point clouds or depth maps, *e.g.* via frame-to-model tracking [36, 66, 6, 38, 21] and with incorporated loop closures [15, 77, 5]. For dense **mapping** the most common scene representations

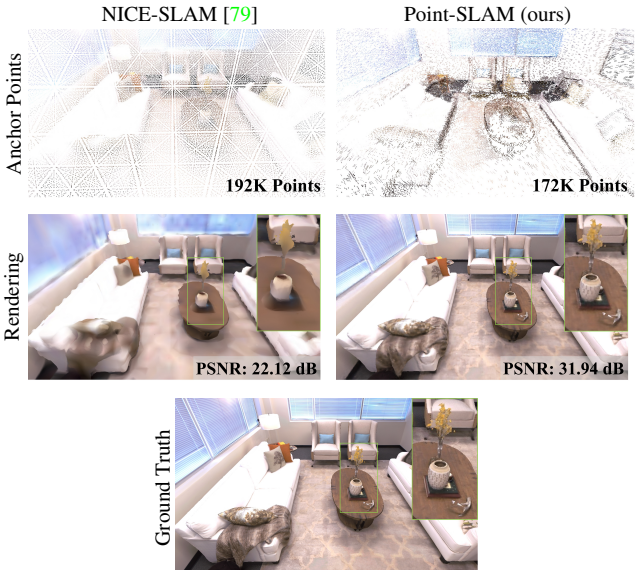


Figure 1: **Point-SLAM Benefits.** Due to the spatially adaptive anchoring of neural features, Point-SLAM can encode high-frequency details more effectively than NICE-SLAM which leads to superior performance in rendering, reconstruction and tracking accuracy while attaining competitive runtime and memory usage. The **first row** shows the feature anchor points. For NICE-SLAM we show the centers of non-empty voxels located on a regular grid, while the density of anchor points for Point-SLAM depends on depth and image gradients. The row below depicts resulting renderings showing substantial differences on areas with high-frequency textures like the vase, blinds, floor or blanket.

are voxel grids [36, 37], voxel hashing [38, 15, 21, 20], octrees [16, 49, 29], or point/surfel clouds [77, 5, 48]. The introduction of learned scene representations [42, 30, 8, 32] has led to rapid progress for learning-based online mapping methods [63, 64, 31, 18, 24, 41] and offline methods [43, 1, 57, 73]. However, most of these methods require ground truth depth or 3D for model training and may not generalize to unseen real-world scenarios at test time. To eliminate the potential domain gap between train and test time, recent SLAM methods rely on test time optimization via volume rendering [53, 69, 79]. Compared to tradi-

*Equal contribution.

tional approaches, neural scene representations have attractive properties for mapping like improved noise and outlier handling [64], better hole filling and inpainting capabilities for unobserved scene parts [69, 79], and data compression [42, 58]. Like DTAM [37] or BAD-SLAM [48] recent neural SLAM methods [79, 69, 53] only use a single scene representation for both tracking and mapping but they rely either on a regular grid structure [79, 69] or a single MLP [53]. Inspired by BAD-SLAM [48], NICE-SLAM [79] and Point-NeRF [67], the research question we tackle in this work is:

Can point-based neural scene representations be used for tracking and mapping for real-time capable SLAM?

To this end, we introduce Point-SLAM, a point-based solution to dense RGBD SLAM, which allows for a data-adaptive scene encoding. The key ideas of our method are as follows: Instead of anchoring the feature points on a regular grid, our approach populates points adaptively depending on information density in the input data which allows for a better memory vs. accuracy trade-off. For rendering, we depart from the classical splatting technique used for surfels and instead aggregate neural point features in a ray-marching fashion. MLP decoders translate these features into scene geometry and color estimates. Tracking and mapping are performed alternatingly by minimizing an RGBD-based re-rendering loss. Different from grid-based approaches, we do not model free space and encode only little information around the surface. We evaluate our proposed method on a selection of indoor RGBD datasets and demonstrate state-of-the-art performance on dense neural RGBD SLAM in terms of tracking, rendering, and mapping - see Fig. 1 for exemplary results. In summary, our **contributions** include:

- We present Point-SLAM, a real-time capable dense RGBD SLAM approach which anchors neural features in a point cloud that grows iteratively in a data-driven manner during scene exploration. We demonstrate that the proposed neural point-based scene representation can be effectively used for both mapping and tracking.
- We propose a dynamic point density strategy which allows for computational and memory efficiency gains and trade reconstruction accuracy against speed and memory.
- Our approach shows clear benefits on a variety of datasets in terms of tracking, rendering and mapping accuracy.

2. Related Work

Dense Visual SLAM and Mapping. Curless and Levoy [13] laid the groundwork for many 3D reconstruction strategies that employ truncated signed distance functions (TSDF). Subsequent developments include KinectFusion [36] and more scalable techniques with voxel hashing [38, 21, 40], octrees [49], and pose robustness via sparse image features [4]. Further extensions involve tracking for SLAM [37, 48, 53, 79, 5, 70] which can also handle loop closures, like BundleFusion [15]. To address the issue of noisy depth maps, RoutedFusion [63] learns a fusion network that outputs the TSDF update of the volumetric grid. NeuralFusion [64] and DI-Fusion [18] extend this concept by learning the scene representation implicitly, resulting in better outlier handling. A number of recent works do not need depth input and accomplish dense online reconstruction from RGB cameras only [35, 10, 3, 50, 54, 47, 23]. Lately, methods relying on test time optimization have become popular due to their adaptability to test time constraints. For example, Continuous Neural Mapping [68] learns a representation of the scene by means of continually mapping from a sequence of depth maps. Neural Radiance Fields [32] inspired works for dense surface reconstruction [39, 59] and pose estimation [45, 25, 62, 2]. These works have led to full dense SLAM pipelines [69, 79, 53, 28], which represent the current most promising trend towards accurate and robust visual SLAM. See [80] for a survey on online RGBD reconstruction. In contrast to our work, none of the neural SLAM approaches supports an input-adaptive scene encoding with high fidelity.

Concurrent to our work, ESLAM [28] tackles RGBD SLAM with axis aligned feature planes and NICER-SLAM [78], NeRF-SLAM [45] and Orbeez-SLAM [12] focus on RGB-only SLAM.

Scene Representations. Most dense 3D reconstruction works can be separated into three categories: (1) *grid-based*, (2) *point-based*, (3) *network-based*. The *grid-based* representation is perhaps the most explored one and can be further split into methods using dense grids [79, 36, 63, 64, 13, 54, 3, 24, 11, 77, 76, 66, 81], hierarchical octrees [69, 49, 29, 6, 26] and voxel hashing [38, 21, 15, 60, 33] to save memory. One advantage of grids is that neighborhood look ups and context aggregations are fast and straightforward. As their main limitation, the grid resolution needs to be specified beforehand and cannot be trivially adapted during reconstruction, even for octrees. This can lead to a suboptimal resolution strategy where memory is wasted in areas with little complexity while not being able to resolve details beyond the resolution choice. *Point-based* representations offer a solution to the issues facing grids and have successfully been applied to 3D reconstruction [65, 48, 5, 12, 21, 22, 9, 74]. For example, analogous to the resolution in grids, the point density does not need to be specified beforehand and can inherently vary across the scene. Further, point sets can be trivially focused around the surface in order not to waste memory on modeling free space. The penalty for this flexibility is a more difficult neighborhood search problem as point sets lack connectivity structure. For dense SLAM,

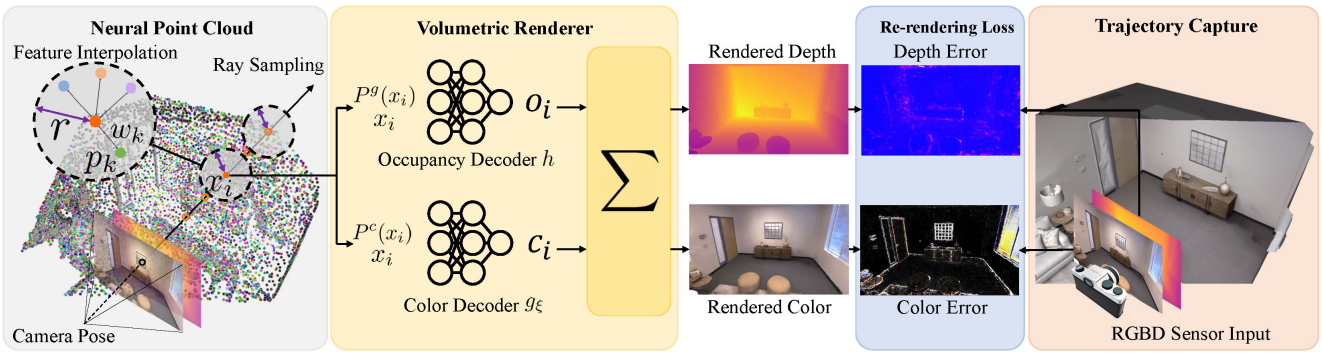


Figure 2: **Point-SLAM Architecture.** Given an estimated camera pose, mapping is performed as follows. We first add a sparse set of neural points to the neural point cloud, and then render depth and color images via volume rendering along the ray. For each sampled pixel we sample a set of points x_i along the ray and extract the geometric and color features ($P^g(x_i)$ and $P^c(x_i)$ resp.) at x_i, using feature interpolation within the spherical search radius r. Each neural point location p_k is weighted by the distance w_k to the sampled point x_i. The features are passed to the occupancy and color decoders (h and g_ξ resp.) along with the point coordinate x_i to extract the occupancy $\mathbf{o}_i$ and color $\mathbf{c_i}$. By imposing a depth and color re-rendering loss to the sensor input RGBD frame, the neural point features are optimized during mapping. Alternating to the mapping step, we perform tracking by optimizing the camera extrinsics while keeping the map fixed.

neighborhood search can be accelerated by converting the 3D search problem into a 2D one by projecting the point set into a set of keyframes [65, 48]. A more elegant and faster solution is to register each point within a grid structure [67]. In this work, we argue that points provide a flexible representation that can benefit from a grid structure for fast neighborhood search. Contrary to previous point- or surfel-based SLAM approaches [65, 48, 5], we benefit from neural implicit features from which rendering is performed through volumetric alpha compositing. *Network-based* methods for dense 3D reconstruction offer a continuous representation by modeling the global scene implicitly through coordinate-MLPs [1, 53, 59, 45, 41, 68, 71, 42, 30]. Benefiting from a simple formulation that is continuous and compressed, network-based methods can recover maps and textures of high quality, but are not suitable for online scene reconstruction for two main reasons: 1) they do not allow for local scene updates, 2) for growing scene size the network capacity cannot be increased at runtime. In this work, we adopt neural implicit representations popularized by network-based methods, but allow for scalability and local updates by anchoring neural point features in 3D space.

Outside the domain of the aforementioned three groups, a few works have studied other representations such as parameterized surface elements [56] and axis aligned feature planes [28, 43]. Parameterized surface elements generally struggle with formulating a flexible shape template while feature planes struggle with scene reconstructions containing multiple surfaces, due to their overly compressed representation. Therefore, we believe that these approaches are not suitable for dense SLAM. Instead we look to model our scene space as a collection of unordered points with corresponding optimizable features.

3. Method

This section details how our neural point cloud is deployed as the sole representation for dense RGBD SLAM. Given an estimated camera pose, points are iteratively added to the scene as new areas are explored (Section 3.1). We make use of per-pixel image gradients to achieve a dynamic point density which aids in resolving fine details while compressing the representation elsewhere. We further detail how depth and color rendering is performed (Section 3.2), with which we minimize a re-rendering loss for both mapping and tracking (Section 3.3). An overview of our method is provided in Fig. 2.

3.1. Neural Point Cloud Representation

We define our neural point cloud as a set of N neural points

$$P = \{(p_i, f_i^g, f_i^c) \,|\, i = 1, ..., N\} \;, \tag{1}$$

each anchored at location $p_i \in \mathbb{R}^3$ and with a geometric and color feature descriptor $f_i^g \in \mathbb{R}^{32}$ and $f_i^c \in \mathbb{R}^{32}$.

Point Adding Strategy. For every mapping phase and a given estimated camera pose, we sample X pixels uniformly across the image plane and Y pixels among the top $5Y$ pixels with the highest color gradient magnitude. Using the available depth information, the pixels are unprojected into 3D where we search for neighbors within a radius r. If no neighbors are found, we add three neural points along the ray, centered at the depth reading D and then offset by $(1-\rho)D$ and $(1+\rho)D$ with $\rho \in (0, 1)$ being a hyperparameter accounting for the expected depth noise. If neighbors are found, no points are added. We use a normally distributed initialization of the feature vectors. The three points act as a limited update band that is depth dependent in order to

model the common noise characteristic of depth cameras. As more frames are processed, our neural point cloud grows progressively to represent the exploration of the scene, but converges to a bounded set of points when no new scene parts are visited. Contrary to many voxel-based representations, it is not required to specify any scene bounds before the reconstruction.

Dynamic Resolution. For computational and memory efficiency, we employ a dynamic point density across the scene. This allows Point-SLAM to efficiently model regions with few details while high point densities are imposed where it is needed to resolve fine details. We implement this by allowing the nearest neighbor search radius r to vary according to the color gradient observed from the sensor. We use a clamped linear mapping to define the search radius r based on the color gradient:

$$r(u,v) = \begin{cases} r_l & \text{if } \nabla I(u,v) \geq g_u \\ \beta_1 \nabla I(u,v) + \beta_2 & \text{if } g_l \leq \nabla I(u,v) \leq g_u \\ r_u & \text{if } \nabla I(u,v) \leq g_l \end{cases} , \quad (2)$$

where $\nabla I(u,v)$ denotes the gradient magnitude at the pixel location (u,v). We use a lower and upper bound (r_l, r_u) for the search radius to control the compression level and memory usage. For more details about parameter choices, we refer to the supplementary material.

3.2. Rendering

To render depth and color, we adopt a volume rendering strategy. Given a camera pose with origin $\mathbf{O}$, we sample a set of points x_i as

$$x_i = \mathbf{O} + z_i \mathbf{d}, \quad i \in \{1, ..., M\} , \quad (3)$$

where $z_i \in \mathbb{R}$ is the point depth and $d \in \mathbb{R}^3$ the ray direction. Specifically, we sample 5 points spread evenly between $(1-\rho)D$ and $(1+\rho)D$, where D is the sensor depth at the pixel to be rendered. This is in contrast to voxel-based frameworks [79, 69] which need to carve the empty space between the camera and the surface, thus requiring significantly more samples. For example, NICE-SLAM [79] uses 48 samples (16 around the surface and 32 between the camera and the surface). With fewer samples along the ray, we achieve a computational speed-up during rendering. After the points x_i have been sampled, the occupancies o_i and colors $\mathbf{c}_i$ are decoded using MLPs following [79] as

$$\mathrm{o}_i = h\big(x_i, P^g(x_i)\big) \qquad \mathbf{c}_i = g_\xi\big(x_i, P^c(x_i)\big) . \quad (4)$$

We denote the geometry and color decoder MLPs by h and g_ξ, respectively, where ξ are the trainable parameters of g. We use the same architecture for h and g as [79] and use their provided pretrained and fixed middle geometric decoder h. The decoder input is the 3D point x_i, to which we apply a learnable Gaussian positional encoding [55] to mitigate the limited band-width of MLPs, and the associated feature. We further denote $P^g(x_i)$ and $P^c(x_i)$ as the geometric and color features extracted at point x_i respectively. For each point x_i we use the corresponding per-pixel query radius $2r$, where r is computed according to Eq. (2). Within the radius $2r$, we require to find at least two neighbors. Otherwise, the point is given zero occupancy. We use the closest eight neighbors and use inverse squared distance weighting for the geometric features, *i.e.*

$$P^g(x_i) = \sum_k \frac{w_k}{\sum_k w_k} f_k^g \quad \text{with } w_k = \frac{1}{||p_k - x_i||^2} . \quad (5)$$

For the color features, inspired by [67], we impose a non-linear preprocessing on the extracted neighbor features f_k^c such that

$$f_{k,x_i}^c = F_\theta(f_k^c, p_k - x_i) , \quad (6)$$

where F is a one-layer MLP parameterized by θ, with 128 neurons and softplus activations. We use the same Gaussian positional encoding for the relative point vector $(p_k - x_i)$ as used by the geometry and color decoders. This yields

$$P^c(x_i) = \sum_k \frac{w_k}{\sum_k w_k} f_{k,x_i}^c . \quad (7)$$

For pixels without depth observation, we render by marching along the ray from the depth $30cm$ to $1.2D_{max}$, where D_{max} is the maximum frame depth. We use 25 samples within this interval. This technique acts as a hole filling technique, but does not fill in arbitrarily large holes, which can cause large completion errors. Next, we describe how the per-point occupancies o_i and colors $\mathbf{c}_i$ are used to render the per-pixel depth and color using volume rendering. We construct a weighting function, α_i as described in Eq. (8). This weight represents the discretized probability that the ray terminates at point x_i.

$$\alpha_i = \mathrm{o}_{\mathbf{p}_i} \prod_{j=1}^{i-1} (1 - \mathrm{o}_{\mathbf{p}_j}) . \quad (8)$$

The rendered depth is computed as the weighted average of the depth values along each ray, and equivalently for the color according to Eq. (9).

$$\hat{D} = \sum_{i=1}^{N} \alpha_i z_i, \quad \hat{I} = \sum_{i=1}^{N} \alpha_i \mathbf{c}_i \quad (9)$$

We also compute the variance along the ray as

$$\hat{S}_D = \sum_{i=1}^{N} \alpha_i \big(\hat{D} - z_i\big)^2 \quad (10)$$

For more details, we refer to [79].

3.3. Mapping and Tracking

Mapping. During mapping, we render M pixels uniformly across the RGBD frame and minimize the re-rendering loss to the sensor reading D and I as

$$\mathcal{L}_{map} = \sum_{m=1}^{M} |D_m - \hat{D}_m|_1 + \lambda_m |I_m - \hat{I}_m|_1 \ , \qquad (11)$$

which combines a geometric L_1 depth loss and a color L_1 loss with hyperparameter λ_m for given ground truth values $\hat{D}_m, \hat{I}_m$. The loss optimizes the geometric and color features f^g and f^c as well as the parameters ξ and θ of the color decoder g and interpolation decoder F respectively. For each mapping phase, we first optimize using only the depth term in order to initialize the color optimization well. We then add the color loss for the remaining 60 % of iterations. Following the same strategy as [79], we make use of a database of keyframes to regularize the mapping loss. We sample a set of keyframes which have a significant overlap with the viewing frustum of the current frame and add pixel samples from the keyframes. More details are provided in the supplementary material.

Tracking. In a separate process to mapping, we perform tracking by optimizing the camera extrinsics $\{\mathbf{R}, \mathbf{t}\}$ at each frame. We sample M_t pixels across the frame and initialize the new pose with a simple constant speed assumption that transforms the last known pose with the relative transformation between the second last pose and the last pose. The tracking loss $\mathcal{L}_{\text{track}}$ combines a color term weighted by λ_t and a geometric term weighted by the standard deviation of the depth prediction:

$$\mathcal{L}_{\text{track}} = \sum_{m=1}^{M_t} \frac{|D_m - \hat{D}_m|_1}{\sqrt{\hat{S}_D}} + \lambda_t |I_m - \hat{I}_m|_1 \qquad (12)$$

3.4. Exposure Compensation

For scenes with significant exposure changes between frames, we use an additional module to reduce color differences between corresponding pixels. Inspired by [44], we learn a per-image latent vector which is fed as input to an exposure MLP G_ϕ with parameters ϕ. The network G is shared between frames and optimized at runtime. It outputs an affine transformation (3×3 matrix and 3×1 translation) which is used to transform the color prediction from Eq. (9) before being fed to the tracking or mapping loss. For more details see the supplementary material.

4. Experiments

We first describe our experimental setup and then evaluate our method against state-of-the-art dense neural RGBD SLAM methods on Replica [51] as well as the real world TUM-RGBD [52] and the ScanNet [14] datasets. Further experiments and details are in the supplementary material.

Implementation Details. For efficient nearest neighborhood search, we use the FAISS library [19] which supports GPU processing. We use $\rho = 0.02$ on Replica and TUM-RGBD and $\rho = 0.04$ on ScanNet. We set $r_l = 0.02$, $r_u = 0.08$, $g_u = 0.15$, $g_l = 0.01$ and $\beta_1 = -\frac{2}{3}$, $\beta_2 = \frac{13}{150}$. For all datasets, $X = 6000$. For Replica $Y = 1000$ and for ScanNet and TUM-RGBD $Y = 0$. For tracking, we sample $M_t = 1.5K$ pixels uniformly on Replica. On TUM-RGBD and ScanNet, we first compute the top $75K$ pixels based on the image gradient magnitude and sample $M_t = 5K$ out of this set. For mapping, we sample uniformly $M = 5K$ pixels for Replica and $10K$ pixels for TUM-RGBD and ScanNet. Although we specify a number of mapping iterations, we use an adaptive scheme which takes the number of newly added points into account. The number of mapping iterations is computed as $m_i = m_i^d n/300$, where m_i^d is the default mapping iterations and n is the number of added points. We clip m_i to lie within $[0.95m_i^d, 2m_i^d]$. This strategy speeds up mapping when few points are added and helps optimize frames with many new points. To mesh the scene, we render depth and color every fifth frame over the estimated trajectory and use TSDF Fusion [13] with voxel size 1 cm. See the supplementary material for more details.

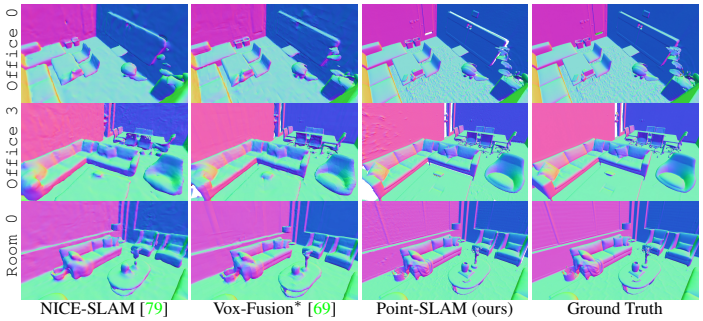
Evaluation Metrics. The meshes, produced by marching cubes [27], are evaluated using the F-score which is the harmonic mean of the Precision (P) and Recall (R). We use a distance threshold of 1 cm for all evaluations. We further provide the depth L1 metric as in [79]. For tracking accuracy, we use ATE RMSE [52] and for rendering we provide the peak signal-to-noise ratio (PSNR), SSIM [61] and LPIPS [75]. Our rendering metrics are evaluated by rendering the full resolution image along the estimated trajectory every 5th frame. Unless otherwise written, we report the average metric of three runs on seeds 0, 1 and 2.

Datasets. The Replica dataset [51] comprises high-quality 3D reconstructions of a variety of indoor scenes. We utilize the publicly available dataset collected by Sucar *et al.* [53], which provides trajectories from an RGBD sensor. Further, we demonstrate that our framework can handle real-world data by using the TUM-RGBD dataset [52], as well as the ScanNet dataset [14]. The poses for TUM-RGBD were captured using an external motion capture system while ScanNet uses poses from BundleFusion [15].

Baseline Methods. We primarily compare our method to existing state-of-the-art dense neural RGBD SLAM methods such as NICE-SLAM [79], Vox-Fusion [69] and ESLAM [28]. We reproduce the results from [69] using the open source code and report the results as Vox-Fusion*. For NICE-SLAM, we use 40 tracking iterations on Replica and mesh the scene at resolution $1cm$ for a fair comparison.

Method	Metric	Rm 0	Rm 1	Rm 2	Off 0	Off 1	Off 2	Off 3	Off 4	Avg.
NICE-SLAM [79]	Depth L1 [cm] ↓	1.81	1.44	2.04	1.39	1.76	8.33	4.99	2.01	2.97
	Precision [%] ↑	45.86	43.76	44.38	51.40	50.80	38.37	40.85	37.35	44.10
	Recall [%] ↑	44.10	46.12	42.78	48.66	53.08	39.98	39.04	35.77	43.69
	F1 [%] ↑	44.96	44.84	43.56	49.99	51.91	39.16	39.92	36.54	43.86
Vox-Fusion* [69]	Depth L1 [cm] ↓	1.09	1.90	2.21	2.32	3.40	4.19	2.96	1.61	2.46
	Precision [%] ↑	75.83	35.88	63.10	48.51	43.50	54.48	69.11	55.40	55.73
	Recall [%] ↑	64.89	33.07	56.62	44.76	38.44	47.85	60.61	46.79	49.13
	F1 [%] ↑	69.93	34.38	59.67	46.54	40.81	50.95	64.56	50.72	52.20
ESLAM [28]	Depth L1 [cm] ↓	0.97	1.07	1.28	0.86	1.26	1.71	1.43	1.06	1.18
Ours	Depth L1 [cm] ↓	**0.53**	**0.22**	**0.46**	**0.30**	**0.57**	**0.49**	**0.51**	**0.46**	**0.44**
	Precision [%] ↑	**91.95**	**99.04**	**97.89**	**99.00**	**99.37**	**98.05**	**96.61**	**93.98**	**96.99**
	Recall [%] ↑	**82.48**	**86.43**	**84.64**	**89.06**	**84.99**	**81.44**	**81.17**	**78.51**	**83.59**
	F1 [%] ↑	**86.90**	**92.31**	**90.78**	**93.77**	**91.62**	**88.98**	**88.22**	**85.55**	**89.77**

(a)



(b)

Figure 3: **Reconstruction Performance on Replica [51].** Fig. 3a: Our method is able to outperform all existing methods. Best results are highlighted as **first**, second, and third. Fig. 3b: Point-SLAM yields on average more precise reconstructions than existing methods, e.g. note the fidelity of the rough carpet reconstruction on Office 0.

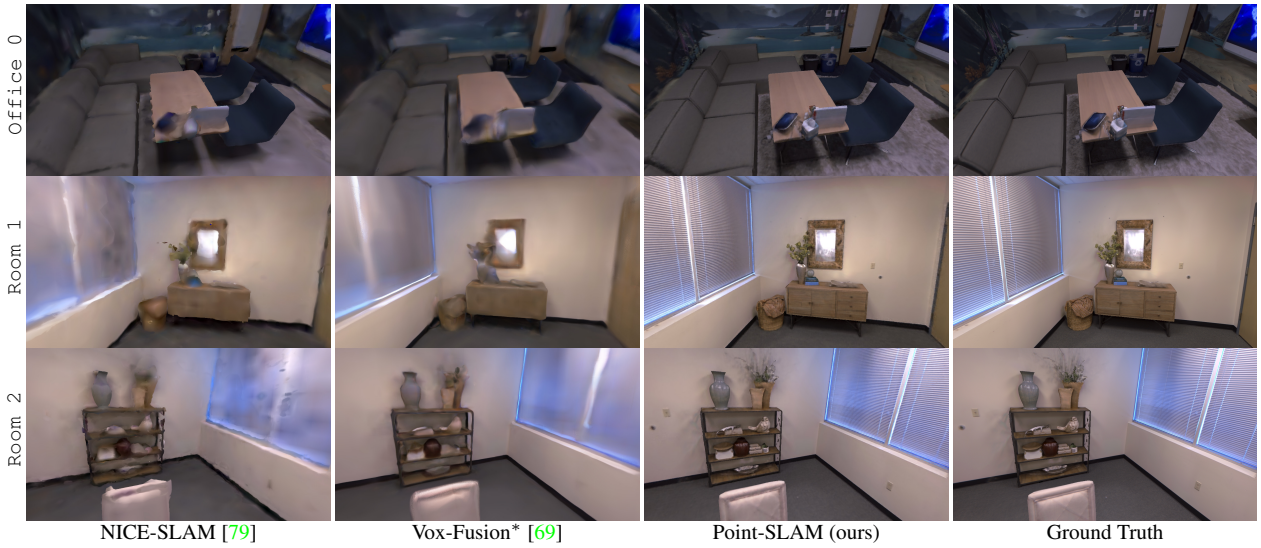


Figure 4: **Rendering Performance on Replica [51].** Thanks to the adaptive density of the neural point cloud, Point-SLAM is able encode more high-frequency details and to substantially increase the fidelity of the renderings. This is also supported by the quantitative results in Table 2.

Method	Rm 0	Rm 1	Rm 2	Off 0	Off 1	Off 2	Off 3	Off 4	Avg.
NICE-SLAM [79]	0.97	1.31	1.07	0.88	1.00	1.06	1.10	1.13	1.06
Vox-Fusion [69]	0.40	0.54	0.54	0.50	0.46	0.75	0.50	0.60	0.54
Vox-Fusion* [69]	1.37	4.70	1.47	8.48	2.04	2.58	1.11	2.94	3.09
ESLAM [28]	0.71	0.70	0.52	0.57	0.55	0.58	0.72	**0.63**	0.63
Point-SLAM (ours)	**0.61**	**0.41**	**0.37**	**0.38**	**0.48**	**0.54**	**0.69**	0.72	**0.52**

Table 1: **Tracking Performance on Replica [51]** (ATE RMSE ↓ [cm]). On average, we achieve better tracking than existing methods. The grayed numbers of [69] are from the paper that come from a single run which we could not reproduce. We report an average of 3 runs for all other methods in this table. Vox-Fusion* indicates recreated results.

4.1. Reconstruction

Fig. 3a compares our method to NICE-SLAM [79], Vox-Fusion [69] and ESLAM [28] in terms of the geometric reconstruction accuracy. We outperform all methods on all metrics and report an average improvement of 85 %, 82 % and 63 % on the depth L1 metric over NICE-SLAM, Vox-Fusion and ESLAM respectively. Fig. 3b compares the mesh reconstructions of NICE-SLAM [79], Vox-Fusion [69] and our method to the ground truth mesh. We find that our method is able to resolve fine details to a significantly greater extent than previous approaches. We attribute this to our neural point cloud which adapts the point density where it is needed (i.e. close to the surface and around fine details) and conserves memory in other areas.

4.2. Tracking

We report the tracking performance on the Replica dataset in Table 1. On average we outperform the existing methods. We believe this is due to the more accu-

Method	Metric	Room 0	Room 1	Room 2	Office 0	Office 1	Office 2	Office 3	Office 4	Avg.
NICE-SLAM [79]	PSNR [dB] ↑	22.12	22.47	24.52	29.07	30.34	19.66	22.23	24.94	24.42
	SSIM ↑	0.689	0.757	0.814	0.874	0.886	0.797	0.801	0.856	0.809
	LPIPS ↓	0.330	0.271	0.208	0.229	0.181	0.235	0.209	0.198	0.233
Vox-Fusion* [69]	PSNR [dB] ↑	22.39	22.36	23.92	27.79	29.83	20.33	23.47	25.21	24.41
	SSIM ↑	0.683	0.751	0.798	0.857	0.876	0.794	0.803	0.847	0.801
	LPIPS ↓	0.303	0.269	0.234	0.241	0.184	0.243	0.213	0.199	0.236
Ours	PSNR [dB] ↑	**32.40**	**34.08**	**35.50**	**38.26**	**39.16**	**33.99**	**33.48**	**33.49**	**35.17**
	SSIM ↑	**0.974**	**0.977**	**0.982**	**0.983**	**0.986**	**0.960**	**0.960**	**0.979**	**0.975**
	LPIPS ↓	**0.113**	**0.116**	**0.111**	**0.100**	**0.118**	**0.156**	**0.132**	**0.142**	**0.124**

Table 2: **Rendering Performance on Replica [51].** We outperform existing dense neural RGBD methods on the commonly reported rendering metrics. For NICE-SLAM [79] and Vox-Fusion [69] we take the numbers from [78]. For qualitative results, see Fig. 4.

Method	fr1/ desk	fr1/ desk2	fr1/ room	fr2/ xyz	fr3/ office	Avg.
DI-Fusion [18]	4.4	N/A	N/A	2.0	5.8	N/A
NICE-SLAM [79]	4.26	4.99	34.49	31.73 (6.19)	3.87	15.87 (10.76)
Vox-Fusion* [69]	3.52	6.00	19.53	1.49	26.01	11.31
Point-SLAM (Ours)	4.34	4.54	30.92	1.31	3.48	8.92
BAD-SLAM [48]	1.7	N/A	N/A	1.1	1.7	N/A
Kintinuous [66]	3.7	7.1	7.5	2.9	3.0	4.84
ORB-SLAM2 [34]	**1.6**	**2.2**	**4.7**	**0.4**	**1.0**	**1.98**
ElasticFusion [65]	2.53	6.83	21.49	1.17	2.52	6.91

Table 3: **Tracking Performance on TUM-RGBD [52]** (ATE RMSE ↓ [cm]). Point-SLAM consistently outperforms existing dense neural RGBD methods (top part), and is reducing the gap to sparse tracking methods (bottom part). In parenthesis we report the average over only the successful runs.

Method	0000	0059	0106	0169	0181	0207	Avg.
DI-Fusion [18]	62.99	128.00	18.50	75.80	87.88	100.19	78.89
NICE-SLAM [79]	12.00	14.00	**7.90**	10.90	**13.40**	**6.20**	**10.70**
Vox-Fusion [69]	8.39	N/A	7.44	6.53	12.20	5.57	N/A
Vox-Fusion* [69]	68.84	24.18	8.41	27.28	23.30	9.41	26.90
	(16.55)						(18.52)
Point-SLAM (Ours)	**10.24**	**7.81**	8.65	22.16	14.77	9.54	12.19

Table 4: **Tracking Performance on ScanNet [14]** (ATE RMSE ↓ [cm]). All scenes are evaluated on the 00 trajectory. We take the numbers from [28] for NICE-SLAM. Tracking failed for one run on Vox-Fusion on scene 0000. In parenthesis we report the average over only the successful runs.

rate scene representation that the neural point cloud provides. We show that the performance of Point-SLAM transfers to real-world data by evaluating on the TUM-RGBD dataset in Table 3. We outperform all existing dense neural RGBD methods. Nevertheless, there is still a gap to traditional methods which employ more sophisticated tracking schemes including loop closures. Finally, Table 4 shows our tracking performance on some selected ScanNet scenes, where we activate the exposure compensation module. We achieve competitive performance on ScanNet, but find that this dataset is generally more complex due to motion blur

Without F_θ. PSNR: 27.41 With F_θ. PSNR: 32.09

Figure 5: **Non-Linear Appearance Space.** A non-linear preprocessing via F_θ of the appearance features helps resolve high frequency textures like the blinds, the pot on the table and the tree print on the pillow.

and specularities. We believe our model is more sensitive to these effects if not modeled properly compared to *e.g.* NICE-SLAM [79] and Vox-Fusion [69] which employ a large voxel size that leads to more averaging and a reduced sensitivity to specularities. We added a more detailed discussion to the supplementary material.

4.3. Rendering

Table 2 compares rendering performance and shows improvements over existing dense neural RGBD SLAM methods. Fig. 4 shows examplary full resolution renderings where Point-SLAM yields more accurate details.

4.4. Further Statistical Evaluation

Non-Linear Appearance Space. We evaluate Point-SLAM on the Room 0 scene of the Replica dataset with and without the non-linear preprocessing network F_θ. Fig. 5 shows that a simple linear weighting of the features cannot resolve high frequency textures like the blinds while this can successfully be done when F_θ is optimized during runtime. Quantitatively, we evaluate the PSNR over the entire trajectory and show a gain of 17% (32.09 vs. 27.41). We find that for higher tracking errors *e.g.* on TUM-RGBD [52] or ScanNet [14], the MLP F_θ is not helpful and we disable it. High-frequency appearance can only be resolved with pixel accurate poses that align the frames correctly.

Color Ablation. We investigate the performance of our

Mapping RGB	Tracking RGB	ATE RMSE [cm]↓	Depth L1 [cm]↓	F1 [%]↑	PSNR [dB]↑
✗	✗	0.59	0.38	91.37	-
✓	✗	0.67	0.38	**91.49**	30.43
✓	✓	**0.36**	**0.35**	91.29	**32.15**

Table 5: **Color Ablation.** The experiment shows that color information is valuable for tracking and marginally for reconstruction.

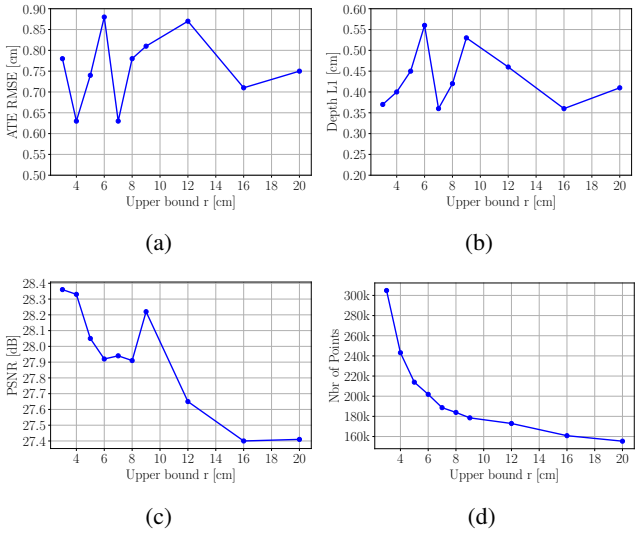


Figure 6: **Dynamic Resolution Ablation**. We show the performance metrics for varying upper bounds r_u of the search radius on the `Room 0` scene. Our method is robust to compression regarding the tracking and mapping accuracy ((a) and (b) resp.). The rendering quality gradually degrades (c) while the memory usage starts to bottom out around $r_u = 8$ cm. We thus choose $r_u = 8$ cm for all experiments.

pipeline when the RGB input is not used for different settings. Table 5 reports performance metrics on `Room 0`. When no RGB is used for tracking, we find that the tracking performance degrades, which negatively affects the depth L1 metric and the rendering quality. The reconstruction performance is mainly determined by the depth input given good camera poses, but since RGB is useful in attaining better poses, we find that RGB information is helpful for both tracking and reconstruction.

Dynamic Resolution Ablation. We show that our method is quite robust to the value of r_u, the upper bound for the search radius. Figs. 6a to 6c display the ATE RMSE, depth L1 and the PSNR respectively as r_u is varied. The tracking and reconstruction metrics are quite robust to r_u while we see a gradual decrease in terms of the PSNR. Fig. 6d shows the total number of neural points at the end of frame capture, for each r_u. We find that the curve bottoms out around $r_u = 8$ cm, which is what we use for all experiments.

Memory and Runtime Analysis. We report runtime and memory usage on the Replica `office 0` scene in Table 6. The tracking and mapping time is reported per iteration and frame. The decoder size denotes the memory footprint of all MLP networks and includes the networks G_ϕ and F_θ. The embedding size is the total memory footprint of the scene representation. The memory usage of Point-SLAM falls between NICE-SLAM an Vox-Fusion while the runtime is competitive. The runtimes were profiled on a single Nvidia RTX 2080 Ti while Vox-Fusion used an RTX 3090.

Method	Tracking /Iteration	Mapping /Iteration	Tracking /Frame	Mapping /Frame	Decoder Size	Embedding Size
NICE-SLAM [79]	32 ms	182 ms	1.32 s	10.92 s	**0.47 MB**	95.86 MB
Vox-Fusion [69]	**12 ms**	55 ms	**0.36 s**	**0.55 s**	1.04 MB	**0.149 MB**
Point-SLAM (ours)	21 ms	**33 ms**	0.85 s	9.85 s	0.51 MB	27.23 MB

Table 6: **Runtime and Memory Usage on Replica** `office 0`. The decoder size is the memory of all MLP networks. The embedding size is the total memory of the scene representation. Our memory usage and runtime are competitive.

Limitations. While our framework demonstrates competitive tracking performance on TUM-RGBD and ScanNet, we believe that a more robust system can be built to handle depth noise, by allowing the point locations to be optimized on the fly. The local adaptation of point densities follows a simple heuristic and should ideally also be learned. We also think that many of our empirical hyperparameters can be made test time adaptive *e.g.* the keyframe selection strategy as well as the color gradient upper and lower bounds to determine the search radius. Finally, while our framework is able to substantially increase the rendering and reconstruction performance over the current state of the art, our system seems more sensitive to motion blur and specularities which we hope to address in future work.

5. Conclusion

We proposed Point-SLAM, a dense SLAM system which utilizes a neural point cloud for both mapping and tracking. The data-driven anchoring of features allows to better align them with actual surface locations and the proposed dynamic resolution strategy populates features depending on the input information density. Overall, this leads to a better balance of memory and compute resource usage and the accuracy of the estimated 3D scene representation. Our experiments demonstrate that Point-SLAM substantially outperforms existing solutions regarding the reconstruction and rendering accuracy while being competitive with respect to tracking as well as runtime and memory usage.

Acknowledgements. This work was supported by a VIVO collaboration project on real-time scene reconstruction and research grants from FIFA. We thank Danda Pani Paudel and Suryansh Kumar for fruitful discussions.

References

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

[2] Wenjing Bian, Zirui Wang, Kejie Li, Jia-Wang Bian, and Victor Adrian Prisacariu. Nope-nerf: Optimising neural radiance field with no pose prior. *arXiv preprint arXiv:2212.07388*, 2022. 2

[3] Aljaž Božič, Pablo Palafox, Justus Thies, Angela Dai, and Matthias Nießner. Transformerfusion: Monocular rgb scene reconstruction using transformers. *arXiv preprint arXiv:2107.02191*, 2021. 2

[4] E. Bylow, C. Olsson, and F. Kahl. Robust online 3d reconstruction combining a depth sensor and sparse feature points. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3709–3714, 2016. 2

[5] Yan-Pei Cao, Leif Kobbelt, and Shi-Min Hu. Real-time high-accuracy three-dimensional reconstruction with consumer rgb-d cameras. *ACM Transactions on Graphics (TOG)*, 37(5):1–16, 2018. 1, 2, 3

[6] Jiawen Chen, Dennis Bautembach, and Shahram Izadi. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(4):1–16, 2013. 1, 2

[7] Timothy Chen, Preston Culbertson, and Mac Schwager. Catnips: Collision avoidance through neural implicit probabilistic scenes, 2023. 1

[8] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *IEEE/CVF conference on computer vision and pattern recognition*, pages 5939–5948, 2019. 1

[9] Hae Min Cho, HyungGi Jo, and Euntai Kim. Sp-slam: Surfel-point simultaneous localization and mapping. *IEEE/ASME Transactions on Mechatronics*, 27(5):2568–2579, 2021. 2

[10] Jaesung Choe, Sunghoon Im, Francois Rameau, Minjun Kang, and In So Kweon. Volumefusion: Deep depth fusion for 3d scene reconstruction. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16086–16095, October 2021. 2

[11] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5556–5565, 2015. 2

[12] Chi-Ming Chung, Yang-Che Tseng, Ya-Ching Hsu, Xiang-Qian Shi, Yun-Hung Hua, Jia-Fong Yeh, Wen-Chin Chen, Yi-Ting Chen, and Winston H Hsu. Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping. *arXiv preprint arXiv:2209.13274*, 2022. 2

[13] Brian Curless and Marc Levoy. Volumetric method for building complex models from range images. In *SIGGRAPH Conference on Computer Graphics*. ACM, 1996. 2, 5

[14] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE/CVF, 2017. 5, 7

[15] 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. 1, 2, 5

[16] Simon Fuhrmann and Michael Goesele. Fusion of depth maps with multiple scales. *ACM Trans. Graph.*, 30(6):148:1–148:8, 2011. 1

[17] Christian Häne, Christopher Zach, Jongwoo Lim, Ananth Ranganathan, and Marc Pollefeys. Stereo depth map fusion for robot navigation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1618–1625. IEEE, 2011. 1

[18] Jiahui Huang, Shi-Sheng Huang, Haoxuan Song, and Shi-Min Hu. Di-fusion: Online implicit 3d reconstruction with deep priors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8932–8941, 2021. 1, 2, 7

[19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. 5

[20] Olaf Kähler, Victor Prisacariu, Julien Valentin, and David Murray. Hierarchical voxel block hashing for efficient integration of depth images. *IEEE Robotics and Automation Letters*, 1(1):192–197, 2015. 1

[21] Olaf Kähler, Victor Adrian Prisacariu, Carl Yuheng Ren, Xin Sun, Philip H. S. Torr, and David William Murray. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Trans. Vis. Comput. Graph.*, 21(11):1241–1250, 2015. 1, 2

[22] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *International Conference on 3D Vision (3DV)*, pages 1–8. IEEE, 2013. 2

[23] Heng Li, Xiaodong Gu, Weihao Yuan, Luwei Yang, Zilong Dong, and Ping Tan. Dense rgb slam with neural implicit maps. *arXiv preprint arXiv:2301.08930*, 2023. 2

[24] Kejie Li, Yansong Tang, Victor Adrian Prisacariu, and Philip HS Torr. Bnv-fusion: Dense 3d reconstruction using bi-level neural volume fusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6166–6175, 2022. 1, 2

[25] Chen Hsuan Lin, Wei Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: Bundle-Adjusting Neural Radiance Fields. In *International Conference on Computer Vision (ICCV)*. IEEE/CVF, 2021. 2

[26] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020. 2

[27] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 5

[28] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. Eslam: Efficient dense slam system based on hy-

brid representation of signed distance fields. *arXiv e-prints*, pages arXiv–2211, 2022. 2, 3, 5, 6, 7

[29] Nico Marniok, Ole Johannsen, and Bastian Goldluecke. An efficient octree design for local variational range image fusion. In *German Conference on Pattern Recognition (GCPR)*, pages 401–412. Springer, 2017. 1, 2

[30] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 1, 3

[31] Marko Mihajlovic, Silvan Weder, Marc Pollefeys, and Martin R Oswald. Deepsurfels: Learning online appearance fusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14524–14535, 2021. 1

[32] 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. In *European Conference on Computer Vision (ECCV)*. CVF, 2020. 1, 2

[33] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 2

[34] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. 7

[35] 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 *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 414–431. Springer, 2020. 2

[36] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, volume 11, pages 127–136, 2011. 1, 2

[37] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *International Conference on Computer Vision (ICCV)*, 2011. 1, 2

[38] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 32, 11 2013. 1, 2

[39] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In *International Conference on Computer Vision (ICCV)*. IEEE/CVF, 2021. 2

[40] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan I. Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board MAV planning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pages 1366–1373. IEEE, 2017. 2

[41] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception. *arXiv preprint arXiv:2204.02296*, 2022. 1, 3

[42] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 1, 2, 3

[43] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional Occupancy Networks. In *European Conference Computer Vision (ECCV)*. CVF, 2020. 1, 3

[44] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban Radiance Fields. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE/CVF, 2021. 5

[45] Antoni Rosinol, John J. Leonard, and Luca Carlone. NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields. *arXiv*, 2022. 2, 3

[46] James Ross, Oscar Mendez, Avishkar Saha, Mark Johnson, and Richard Bowden. Bev-slam: Building a globally-consistent world map using monocular vision. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3830–3836. IEEE, 2022. 1

[47] Mohamed Sayed, John Gibson, Jamie Watson, Victor Prisacariu, Michael Firman, and Clément Godard. Simplerecon: 3d reconstruction without 3d convolutions. In *European Conference on Computer Vision*, pages 1–19. Springer, 2022. 2

[48] Thomas Schops, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle adjusted direct RGB-D SLAM. In *CVF/IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3, 7

[49] Frank Steinbrucker, Christian Kerl, and Daniel Cremers. Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *IEEE International Conference on Computer Vision*, pages 3264–3271, 2013. 1, 2

[50] Noah Stier, Alexander Rich, Pradeep Sen, and Tobias Höllerer. Vortx: Volumetric 3d reconstruction with transformers for voxelwise view selection and fusion. In *2021 International Conference on 3D Vision (3DV)*, pages 320–330. IEEE, 2021. 2

[51] 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. 5, 6, 7

[52] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2012. 5, 7

[53] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J. Davison. iMAP: Implicit Mapping and Positioning in Real-Time. In *International Conference on Computer Vision (ICCV)*. IEEE/CVF, 2021. 1, 2, 3, 5

[54] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15598–15607, 2021. 2

[55] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020. 4

[56] Maria Vakalopoulou, Guillaume Chassagnon, Norbert Bus, Rafael Marini, Evangelia I Zacharaki, M-P Revel, and Nikos Paragios. Atlasnet: multi-atlas non-linear deep networks for medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 658–666. Springer, 2018. 3

[57] Jingwen Wang, Tymoteusz Bleja, and Lourdes Agapito. Go-surf: Neural feature grid optimization for fast, high-fidelity rgb-d surface reconstruction. In *International Conference on 3D Vision*, 2022. 1

[58] Jiepeng Wang, Peng Wang, Xiaoxiao Long, Christian Theobalt, Taku Komura, Lingjie Liu, and Wenping Wang. Neuris: Neural reconstruction of indoor scenes using normal priors. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 139–155. Springer, 2022. 2

[59] 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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2, 3

[60] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. *arXiv preprint arXiv:2212.05231*, 2022. 2

[61] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 5

[62] 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

[63] Silvan Weder, Johannes Schonberger, Marc Pollefeys, and Martin R Oswald. Routedfusion: Learning real-time depth map fusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4887–4897, 2020. 1, 2

[64] Silvan Weder, Johannes L Schonberger, Marc Pollefeys, and Martin R Oswald. Neuralfusion: Online depth fusion in latent space. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3162–3172, 2021. 1, 2

[65] Thomas Whelan, Stefan Leutenegger, Renato Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: Science and Systems (RSS)*, 2015. 2, 3, 7

[66] Thomas Whelan, John McDonald, Michael Kaess, Maurice Fallon, Hordur Johannsson, and John J. Leonard. Kintinuous: Spatially extended kinectfusion. In *Proceedings of RSS '12 Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012. 1, 2, 7

[67] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 2, 3, 4

[68] 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 *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15782–15792, October 2021. 2, 3

[69] Xingrui Yang, Hai Li, Hongjia Zhai, Yuhang Ming, Yuqian Liu, and Guofeng Zhang. Vox-fusion: Dense tracking and mapping with voxel-based neural implicit representation. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 499–507. IEEE, 2022. 1, 2, 4, 5, 6, 7, 8

[70] Xingrui Yang, Yuhang Ming, Zhaopeng Cui, and Andrew Calway. Fd-slam: 3-d reconstruction using features and dense matching. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8040–8046. IEEE, 2022. 2

[71] 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

[72] Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. Ds-slam: A semantic visual slam towards dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1168–1174. IEEE, 2018. 1

[73] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 1

[74] Heng Zhang, Guodong Chen, Zheng Wang, Zhenhua Wang, and Lining Sun. Dense 3d mapping for indoor environment based on feature-point slam method. In *2020 the 4th International Conference on Innovation in Artificial Intelligence*, pages 42–46, 2020. 2

[75] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 5

[76] Qian-Yi Zhou and Vladlen Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics (ToG)*, 32(4):1–8, 2013. 2

[77] Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. Elastic fragments for dense scene reconstruction. In *IEEE International Conference on Computer Vision*, pages 473–480, 2013. 1, 2

[78] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. *arXiv preprint arXiv:2302.03594*, 2023. 2, 7

[79] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12786–12796, 2022. 1, 2, 4, 5, 6, 7, 8

[80] Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Nießner, Reinhard Klein, and Andreas Kolb. State of the art on 3d reconstruction with rgb-d cameras. In *Computer graphics forum*, volume 37, pages 625–652. Wiley Online Library, 2018. 2

[81] Zi-Xin Zou, Shi-Sheng Huang, Yan-Pei Cao, Tai-Jiang Mu, Ying Shan, and Hongbo Fu. Mononeuralfusion: Online monocular neural 3d reconstruction with geometric priors. *arXiv preprint arXiv:2209.15153*, 2022. 2