IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

HI-SLAM: Monocular Real-time Dense Mapping with Hybrid Implicit Fields

Wei Zhang, *Student Member, IEEE,* Tiecheng Sun, *Member, IEEE,* Sen Wang, Qing Cheng, Norbert Haala

***Abstract*—In this letter, we present a neural field-based real-time monocular mapping framework for accurate and dense Simultaneous Localization and Mapping (SLAM). Recent neural mapping frameworks show promising results, but rely on RGB-D or pose inputs, or cannot run in real-time. To address these limitations, our approach integrates dense-SLAM with neural implicit fields. Specifically, our dense SLAM approach runs parallel tracking and global optimization, while a neural field-based map is constructed incrementally based on the latest SLAM estimates. For the efficient construction of neural fields, we employ multi-resolution grid encoding and signed distance function (SDF) representation. This allows us to keep the map always up-to-date and adapt instantly to global updates via loop closing. For global consistency, we propose an efficient $Sim(3)$-based pose graph bundle adjustment (PGBA) approach to run online loop closing and mitigate the pose and scale drift. To enhance depth accuracy further, we incorporate learned monocular depth priors. We propose a novel joint depth and scale adjustment (JDSA) module to solve the scale ambiguity inherent in depth priors. Extensive evaluations across synthetic and real-world datasets validate that our approach outperforms existing methods in accuracy and map completeness while preserving real-time performance.**

***Index Terms*—SLAM; Mapping; Deep Learning for Visual Perception**

I. Introduction

SIMULTANEOUS real-time dense mapping of the environment and camera tracking has long been a popular research topic, with vast applications in robot navigation, AR/VR, and autonomous driving. Classical SLAM approaches [1], [2], [3] can provide accurate camera poses, but typically yield sparse or up-to-semi-dense maps, which are insufficient for most robotic applications. Some works [4], [5] provide dense mapping, but their pose estimation is not accurate enough or they cannot generalize well to large-scale scenes.

With the growth of computing resources and advances in deep learning, real-time monocular dense SLAM is now becoming feasible. Moreover, the emergence of neural implicit fields [6] provides a new, flexible representation for dense SLAM, allowing for more complex and memory-efficient scene representation. iMAP [7] presents the concept of utilizing a single MLP [8] to jointly perform pose tracking and neural mapping. NICE-SLAM [8] introduces multi-resolution grids to improve efficiency. Follow-up works [9], [10] apply signed distance function (SDF) for better surface definition. However, most approaches face inherent limitations, such as reliance on RGB-D [11], [12] or fail to run in real-time [10]. Moreover, previous works lack a critical aspect of SLAM: loop closure, which is essential for robots to recognize previously visited places and correct pose drift [13]. The concurrent work [14] integrates a loop closure module, but it relies on computing all-pairs co-visibility and deploying computationally intensive full BA. This considerably slows down the system and increases the risk of lost tracking. To overcome these challenges, we propose a new set-up for our SLAM system: real-time dense monocular SLAM with online loop closing and map update.

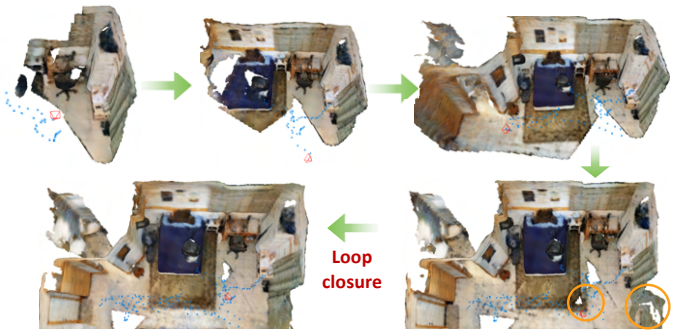


Fig. 1. Parallel pose tracking and dense mapping by the proposed system. In addition, our method performs on-the-fly map updates when loop closure is detected.

Implementing such a SLAM system poses numerous challenges should be considered: real-time and dense settings demand significant computing resources. Furthermore, monocular SLAM systems face challenges including depth ambiguity, scale drift, potential slow convergence, and the risk of falling into local minima [10]. Last but not least, in neural mapping systems, global updates through loop closures could be tricky to incorporate into the map as soon as possible. Delays can result in the accumulation of map artifacts due to the increasing number of processed frames and the quick collapsing of reconstruction. Therefore, it is crucial to address all of these challenges without compromising accuracy, robustness, or efficiency.

In this paper, we propose a novel approach that combines deep learning-based dense SLAM with neural implicit fields to generate dense maps in real-time without the reliance on RGB-D or pose input as in previous approaches. Our

Manuscript received: September, 28, 2023; Revised: December, 6, 2023; Accepted: December, 13, 2023. This paper was recommended for publication by Editor Sven Behnke upon evaluation of the Associate Editor and Reviewers' comments.

Wei Zhang is with the Institute for Photogrammetry, University of Stuttgart, Germany, and also with the Huawei Munich Research Center, Germany (email: wei.zhang@ifp.uni-stuttgart.de)

Tiecheng Sun is with Central Media Technology Institute, Huawei 2012 Laboratories, China (email: suntiecheng1@huawei.com)

Sen Wang and Qing Cheng are with the Technical University of Munich, Germany, and also with the Huawei Munich Research Center, Germany (email: sen.wang@tum.de; qing.cheng@tum.de)

Norbert Haala is with the Institute for Photogrammetry, University of Stuttgart, Germany (email: norbert.haala@ifp.uni-stuttgart.de)

Digital Object Identifier (DOI): see top of this page.

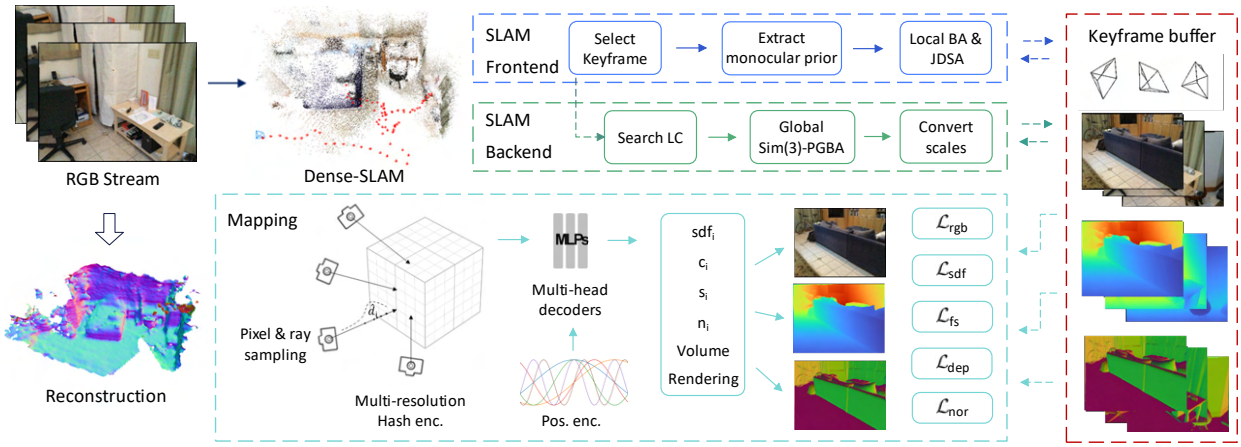


Fig. 2. **System overview**. Given an RGB image stream, our system runs parallel tracking and mapping. On tracking part, two processes, namely frontend and backend, are spawn for local and global consistent tracking respectively. Our SLAM frontend further leverages a pre-trained CV model to predict monocular geometric priors. The keyframe data, including estimated poses, depths, and monocular normal priors, are shared between processes. On the mapping side, the neural map is constructed incrementally based on the latest estimates from the shared buffer in an online manner.

method takes full advantage of the representational capability of deep learning and the adaptability of neural implicit fields, providing a robust, efficient and accurate solution for real-time monocular dense SLAM. Furthermore, we incorporate easy-to-obtain monocular priors into our framework to recover more geometric details and maintain surface smoothness. To ensure global consistency, we run a SLAM backend in parallel to frontend tracking. This backend searches for potential loop closures and performs the proposed efficient $Sim(3)$-based pose graph bundle adjustment (PGBA). On the global update, the neural map is updated instantly according to the updated states. Fig. 1 shows the incremental map reconstruction process by our method, including online adaptations to loop closure updates. Through extensive experiments on both synthetic and real-world datasets, we prove that our proposed hybrid implicit dense SLAM framework can not only run in real-time but also achieve favorable accuracy and higher completeness compared to the offline methods.

The key contributions of the proposed approach are summarized as follows:

- A novel hybrid dense SLAM framework that combines the complementary features of deep learning-based dense SLAM and neural implicit fields for real-time monocular dense mapping.
- A joint depth and scale adjustment (JDSA) approach that solves the scale ambiguity of monocular depth priors and improves the quality of depth estimation.
- An efficient SLAM backend utilizing the $Sim(3)$ pose representation and pose graph BA that can correct both pose and scale drift to enable global consistent mapping.
- An effective neural map optimization scheme, which can be updated on-the-fly with rapid camera motion, and also adapt instantly with global changes by successful loop closures.

II. RELATED WORKS

Monocular dense SLAM. Over past decades, the monocular dense SLAM technique has seen significant development. DTAM [15] pioneered one of the first real-time dense SLAM systems by parallelizing depth computing on GPU. To balance computational cost and accuracy, there are semi-dense methods [16], which however do not capture texture-poor regions. In the deep learning era, many works [4], [5] stride to push the density limit by estimating dense depth maps of keyframes along with poses, but their tracking accuracy lags behind the traditional sparse landmark-based approaches. DROID-SLAM [17] proposes to apply an optical flow network to establish dense pixel correspondences and achieve excellent trajectory estimation. Another line of works [18], [19] combines real-time VIO/SLAM systems with MVS methods for parallel tracking and dense depth estimation. The truncated signed distance function (TSDF) is then used to fuse depth maps and extract meshes. In our work, we also adopt voxel representation but store feature encodings instead of direct SDF values. This allows us to refine our map with photometric terms and regularization terms through SLAM estimations and geometric priors.

Neural SLAM. Recent advances in Neural Radiance Field (NeRF) have shown strong ability in scene representation. iMAP [7] begins to incorporate this representation into the SLAM system to perform joint neural scene and pose optimization. Later, many follow-up methods [8], [9], [11], [12] emerged, and neural SLAM performance was rapidly improved and reached comparable accuracy to classical methods. Nevertheless, these methods require accurate depth inputs to overcome the shortcomings of NeRF. Recently, researchers made strides to make monocular neural SLAM possible. [20], [10] introduce more sophisticated loss functions, such as warping loss and optical flow loss, to address the depth ambiguity problem but they cannot run in real-time and do not include loop closing.

Hybrid dense neural SLAM. Recently, instead of jointly optimizing poses and neural fields, a few methods have made efforts to merge neural SLAM with classical or dense learning-based SLAM methods. Orbeez-slam [21] resorts to ORB-SLAM [3] to obtain poses and sparse point clouds, which are leveraged to regularize neural field learning. NeRF-SLAM [22] combines DROID-SLAM [17] with InstantNGP [23] to build real-time NeRF representation and can synthesize photo-

realistic novel views. GO-SLAM [14], concurrent to us, presents a hybrid dense SLAM system with similar spirit to ours. While it uses expensive full BA to achieve global consistency, our system can run more efficient loop closing with pose graph BA and correct scale drift. Moreover, we integrate valuable monocular priors through scale adjustment and surface regularization to boost scene geometry estimation. We achieve both higher accuracy and completeness than GO-SLAM while running faster.

III. METHOD

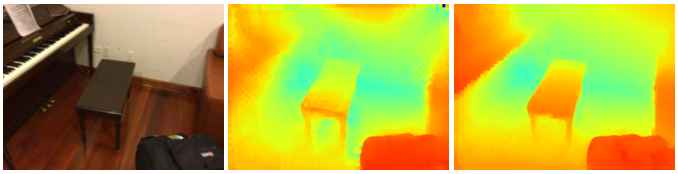
Given an RGB image stream, the goal of our framework is to simultaneously track camera poses and reconstruct high-quality and globally consistent scene geometry in real-time. Fig. 2 provides an overview of our system. To achieve this, we design a multi-process pipeline to run parallel tracking and mapping. Specifically, in the tracking part, we spawn two processes to perform robust tracking (Sec. III-A) and global optimization with loop closures (Sec. III-B). Concurrently, the mapping process reconstructs the scene incrementally using the continuously updated states estimated by the SLAM frontend and backend processes (Sec. III-C).

A. Robust Frontend Tracking

To robustly track the camera poses under challenging scenarios, such as low texture and rapid movement, we build our SLAM system on the foundation of DROID-SLAM [17], which adopts optical flow network to accurately predict dense pixel correspondences between nearby frames. Our system maintains a keyframe graph $(\mathcal{V}, \mathcal{E})$ representing the co-visibility of keyframes and a keyframe buffer storing keyframe information and their respective states. For each incoming frame, the mean flow distance to the last keyframe is determined by a single-pass through the optical flow network. If the distance surpasses a predefined threshold d_{flow}, the current frame is selected as a keyframe and added to the buffer. The edges between this new keyframe and its neighbors are inserted into the keyframe graph. Besides, nearby keyframes with high co-visibility, namely those with small flow distance, are also linked to the latest keyframe. These edges extend the tracking duration of each view. Local BA is then performed based on a co-visibility graph within a sliding window. We employ the flow predictions by the network as targets, denoted as $\check{\mathbf{p}}_{ij}$, and refine the poses $\mathbf{T}$ and depth maps $\mathbf{d}$ of the keyframes involved. This BA optimization is solved iteratively using a damped Gauss-Newton algorithm with the following objective:

$$\underset{\mathbf{T},\mathbf{d}}{\arg\min} \sum_{(i,j)\in\mathcal{E}} \|\check{\mathbf{p}}_{ij} - \Pi(\mathbf{T}_{ij}\Pi^{-1}(\mathbf{p}_i, \mathbf{d}_i))\|^2_{\Sigma_{ij}} \tag{1}$$

where $\mathbf{d}_i$ refers to the depth map of keyframe i in inverse depth parametrization, Π and Π^{-1} are the projection and back-projection functions, Σ_{ij} is a diagonal matrix composed of the prediction confidences by the network. This matrix serves to weigh down the influence of occluded and hard-to-match pixels. However, in this way, the pixels with low confidences attain high depth variances in occluded or texture-poor regions and can not achieve accurate depth estimation, as depicted in Fig. 3. To address this, we extend the system by incorporating monocular depth priors.



Input image BA depth BA+JDSA

Fig. 3. Comparison of depths with and without incorporating depth prior by JDSA module.

Incorporate monocular depth prior. The depth map estimation is largely dependent on the matching ability of the network, which is highly based on image texture. In regions with low-texture, the network often struggles to find robust correspondences. As shown in Fig. 4, depth certainties (inverse of variances) are low on texture poor regions such as white tables and walls. To this end, we leverage the off-the-shelf monocular depth network [24] to generate depth priors. This can then be incorporated into our BA optimization process. Although this network is versatile and can generalize to unseen scenes, the predicted depths prior are relative with varying scales across different viewpoints.

To address this, for each depth prior $\check{\mathbf{d}}_i$, we estimate a scale s_i and offset o_i. We then attempt to incorporate them as variables within the BA optimization. The depth prior factor can be formulated as:

$$r_d = \|(\check{\mathbf{d}}_i \cdot s_i + o_i) - \mathbf{d_i}\|^2 \tag{2}$$

We observe that the prior scales may not necessarily converge when jointly optimized with camera poses. The scales of camera poses could be misdirected by scale-varying priors to shift away. Thus, we explore alternative methods to separate the monocular scale estimation from the BA problem. Specifically, we introduce a joint depth and scale adjustment (JDSA) module. This module minimizes both reprojection factors and depth prior factors, while keeping camera poses fixed. Fixing the poses in this JDSA module prevents scale drift which can arise from the scale-varying priors, and the prior scales can be properly aligned to the system scale. The objective of the JDSA problem can be formulated as:

$$\begin{aligned}\underset{\mathbf{d},\mathbf{s},\mathbf{o}}{\arg\min} \sum_{(i,j)\in\mathcal{E}} \|\check{\mathbf{p}}_{ij} - \Pi(\mathbf{T}_{ij}\Pi^{-1}(\mathbf{p}_i, \mathbf{d}_i))\|^2_{\Sigma_{ij}} + \\ \sum_{i\in\mathcal{V}} \|(\check{\mathbf{d}}_i \cdot s_i + o_i) - \mathbf{d}_i\|^2\end{aligned} \tag{3}$$

In each iteration, we alternate between the BA and JDSA optimizations, which can complement to each other. The poses estimated by BA ensure the consistent scales of depth priors in JDSA optimization. In return, the JDSA provides refined depths that facilitate easier flow updates. This updates are then converted into poses and depths via BA. Fig. 3 shows the enhanced depth accuracy, especially in low texture areas such as white wall and black bench.

B. Backend with Global Consistent Optimization

While the SLAM frontend can reliably track accurate camera poses, pose drift can accumulate inevitably over long distances. Finding loop closing and performing global optimization is an effective way to minimize this drift error. Moreover, due to inherent scale ambiguity, monocular SLAM methods also face scale drift. We first introduce how we detect loop closures and then present our proposed $Sim(3)$-based pose graph BA, designed for efficient loop closing in an online system.

Loop closure detection. Loop closure detection runs in parallel to the tracking process. For each new keyframe, we compute the flow distances d_{of} between the new keyframe and previous keyframes. Three criteria are defined for selecting loop closure candidates. Firstly, d_{of} should fall below a predefined threshold τ_{flow}, ensuring adequate co-visibility for successful convergence of recurrent flow updates. Secondly, orientation differences based on current pose estimation should remain below a threshold τ_{ori}. Lastly, the difference in frame indices should be at least τ_{temp} indices. If all criteria are satisfied, we add edges between the selected keyframe pairs bidirectionally into our keyframe graph.

$Sim(3)$-based pose graph BA. Upon a few loop closure candidates are detected, inspired by [25], we opt for pose graph BA over full BA to enhance efficiency while maintaining accuracy. To tackle scale drift, we use $Sim(3)$ to represent keyframe poses allowing for scale updates. Before each run, we convert the latest pose estimates from $SE(3)$ to $Sim(3)$ and initialize the scales with ones. The pixel warping step follows Eq. 1, but the $SE(3)$ transformation is replaced by $Sim(3)$ transformation.

Another aspect of pose graph BA is to construct a pose graph connected by relative pose edges. Follows [25], we compute relative poses from the dense correspondences of inactive reprojection edges. These come into play when their associated keyframes exit the sliding window of the frontend. Having been refined multiple times while active in the sliding window, these dense correspondences offer a solid foundation for computing relative poses. The same reprojection error term in Eq. 1 is used but only optimizing for the relative poses $\check{\mathbf{T}}_{ij}$ under the assumption the depths are already accurately estimated. Along with the estimated relative poses, the associated variances Σ_{ij}^{rel} are estimated based on the adjustment theory [26] as:

$$\Sigma_{ij}^{rel} = (\mathbf{J}\Delta\mathbf{T_{ij}} - \mathbf{r})^T \Sigma_{ij} (\mathbf{J}\Delta\mathbf{T_{ij}} - \mathbf{r})(\mathbf{J}^T \Sigma_{ij} \mathbf{J})^{-1} \tag{4}$$

where $\mathbf{J}$, $\mathbf{r}$ and $\Delta\mathbf{T_{ij}}$ are the Jacobian, the reprojection residuals, and the relative pose update from the last iteration. The relative pose variances serve as weights for pose graph BA. Finally, the objective PGBA problem is to minimize the sum of the relative pose factors and reprojection factors as:

$$\begin{aligned} \arg\min_{\mathbf{T},\mathbf{d}} \sum_{(i,j)\in\mathcal{E}*} \|\check{\mathbf{p}}_{ij} - \Pi(\mathbf{T}_{ij}\Pi^{-1}(\mathbf{p}_i, \mathbf{d}_i))\|^2_{\Sigma_{ij}} + \\ \sum_{(i,j)\in\mathcal{E}^+} \|\log(\check{\mathbf{T}}_{ij} \cdot \mathbf{T}_i \cdot \mathbf{T}_j^{-1})\|^2_{\Sigma_{ij}^{rel}} \end{aligned} \tag{5}$$

where $\mathcal{E}*$ and $\mathcal{E}^+$ are the set of detected loop closures and the set of relative pose factors respectively.

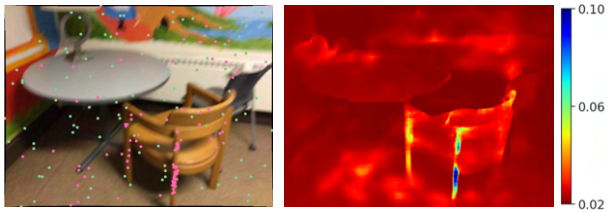


Fig. 4. Left: example sample pixels of certainty-guided, uniform-based; Right: depth certainties.

C. Hybrid Scene Representation

For our scene representation, we first leverage multi-resolution hash feature grid [23] denoted as $h_{\theta_{hash}}$ with optimizable parameters θ_{hash}. For a sample point $\mathbf{x}$, features at every resolution are looked up via tri-linear interpolation and concatenated together. This yields a coarse-to-fine feature encoding. To enhance the scene completion capability, inspired by [6], [11], we also adopt the positional encoding $\gamma(\mathbf{x})$ to map the coordinate to a encoding in high-dimensional spaces. Both the hash and positional encodings serve both spatial and geometric purposes and are fed into our SDF network.

Our SDF network, denoted as $f_{\theta_{sdf}}$, serves as a geometry decoder. It predicts a SDF value s and a geometric feature vector h expressed as:

$$s, \mathbf{h} = f_{\theta_{sdf}}(\gamma(\mathbf{x}), h_{\theta_{hash}}(\mathbf{x})) \tag{6}$$

where θ_{sdf} represents the learnable parameters of our SDF network, which is a shallow MLP with two 32-dimension hidden layers. Following [27] and given the differentiability of both the hash feature grid and SDF network, we can compute the analytical gradient of the SDF function $\nabla f_{\theta_{sdf}}$. After normalization, this gradient becomes the surface normal n.

Similar to our SDF decoder, we also employ a color network to predict the color value c as:

$$\mathbf{c} = f_{\theta_{color}}(\gamma(\mathbf{x}), \mathbf{h}) \tag{7}$$

where θ_{color} denotes the learnable parameters of the color network, which shares the same MLP architecture as our SDF network.

Depth-guided pixel and ray sampling. From the keyframe buffer, the latest depth estimations with associated variances provides readily the guidance of neural map optimization. During each map optimization step, we sample N_{pixel} pixels from the current keyframe buffer. While NeRF and many follow-up methods use straightforward uniform pixel sampling, we observe that they tend to poorly reconstruct small objects or smooth out boundaries, due to insufficient sampling in these areas. To address this, we propose a depth certainty-guided importance sampling strategy. Specifically, we sample $N_{pixel}/2$ pixels based on the depth variance of each pixel. Pixels with lower variance (means higher certainty) are sampled more frequently. As illustrated in Fig. 4, pixels on object borders typically have higher certainties than those on the flat surfaces. For the remaining $N_{pixel}/2$ pixels, we combine uniform sampling to ensure coverage on low-texture areas, such as floors and walls.

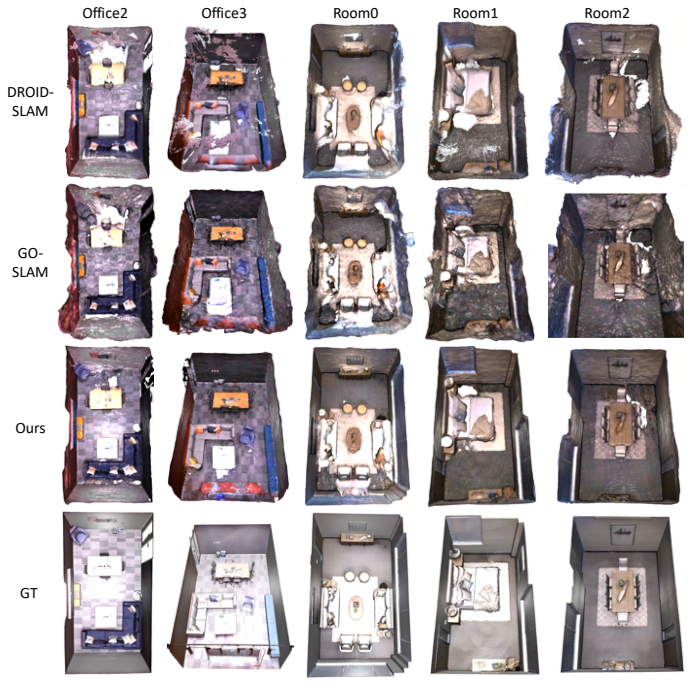


Fig. 5. Reconstruction results on Replica dataset [30]. Our results show smoother surfaces with more detailed geometric compared to DROID-SLAM [17] and GO-SLAM [14].

After the pixel sampling step, rays are cast from the optical center through the sampled pixels, and we sample query points along these rays. Inspired by [11], we first sample M_d points centered around the estimated depth. In addition, M_u points are uniformly sampled between the predefined near and far bounds. For each ray, we sample a total of $M = M_u + M_d$ query points.

Following the bell-shaped formulation in [28], we can re-render the depth and color value of a pixel. We first compute the weights of each sample point from its predicted signed distance values as:

$$w_i = \sigma(\frac{s_i}{tr}) \cdot \sigma(-\frac{s_i}{tr}) \tag{8}$$

where $tr = 10\,cm$ denotes the truncation distance. We then normalize the weights by the sum of all values on each ray. Using this weights, we can calculate the rendered depth, color, and normal by accumulating the values along the ray [29] as:

$$\hat{\mathbf{C}} = \sum_{i=1}^{M} w_i \mathbf{c_i}, \quad \hat{D} = \sum_{i=1}^{M} w_i d_i, \quad \hat{\mathbf{N}} = \sum_{i=1}^{M} w_i \mathbf{n_i} \tag{9}$$

Optimization losses. Our neural map optimization is performed based on various objective functions with respect to the learnable parameters $\theta = \{\theta_{hash}, \theta_{sdf}, \theta_{color}\}$. Following [6], we first apply the color rendering loss, which computes errors between the rendered colors and input image colors as:

$$\mathcal{L}_c = \frac{1}{N} \sum_{n=1}^{N} (\hat{\mathbf{C}}_\mathbf{n} - \mathbf{C}_n)^2 \tag{10}$$

Likewise, the rendered depths are supervised by estimated depths from dense SLAM:

$$\mathcal{L}_d = \frac{1}{N} \sum_{n=1}^{N} \frac{(\hat{D}_n - D_n)^2}{\sigma_n^2} \tag{11}$$

where σ_n^2 is the associated depth variance derived from the Hessian matrix of the BA problem [31]. This effectively reduces the influence of uncertain noisy depths. Furthermore, we supervise the surface normal prediction by the monocular normal prior $\mathbf{N}_n$ by pre-trained Omnidata network [24] as:

$$\mathcal{L}_n = \frac{1}{N} \sum_{n=1}^{N} (\hat{\mathbf{N}}_n - \mathbf{N}_n) \tag{12}$$

where $\mathbf{N}_n$ is transformed from the local coordinate frame to the global one using the currently estimated poses. This ensures consistency with the SDF gradient that is in the global frame.

To accelerate training, following [9], [11], we also directly supervise the SDF predictions. For those points within the truncation bounds, namely the point set S^{tr} where $|D - d_i| \leqslant tr$, we ensure that the neural fields learn to approximate a surface distribution. This is achieved using the pseudo-ground-truth SDF values derived from the estimated depths:

$$\mathcal{L}_{sdf} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{|S^{tr}|} \sum_{p \in S^{tr}} (s_p - (D - d_i))^2 \tag{13}$$

For the sampled points outside the truncation bounds, denoted as S^{fs}, we enforce the SDF prediction to match the truncation distance tr in order to encourage the prediction in free space:

$$\mathcal{L}_{fs} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{|S^{fs}|} \sum_{p \in S^{fs}} (s_p - tr)^2 \tag{14}$$

Finally, our total loss can be formulated as:

$$\mathcal{L} = \lambda_c \mathcal{L}_c + \lambda_d \mathcal{L}_d + \lambda_n \mathcal{L}_n + \lambda_{sdf} \mathcal{L}_{sdf} + \lambda_{fs} \mathcal{L}_{fs} \tag{15}$$

where we assign the loss weights $\lambda_c, \lambda_d, \lambda_n, \lambda_{sdf}$ and λ_{fs} to 10, 0.1, 1, 1000, and 2 respectively. This ensures a balance between the photometric and various geometric supervisions. For each newly created keyframe, we optimize our neural map representation for 10 iterations. For global updates via loop closures, we extend the optimization process to 50 iterations accounting for greater changes. While this is generally effective for major map changes, it might fall short in refining the finer details. However, subsequent map updates on new keyframes can continually enhance the details of the updated regions, progressively improving the overall scene quality.

IV. EXPERIMENTS

We evaluate our proposed system on both synthetic and real-world datasets, including Replica [30], ScanNet [32], and the larger-scale Apartment dataset from NICE-SLAM [8]. Both tracking accuracy and reconstruction quality metrics are reported and compared with prior works. Additionally, we conduct an ablation study to validate the effectiveness of the proposed components. Finally, a runtime analysis is provided.

A. Implementation Details

We build our BA and neural map optimization using the PyTorch library and employ LieTorch library [33] for pose

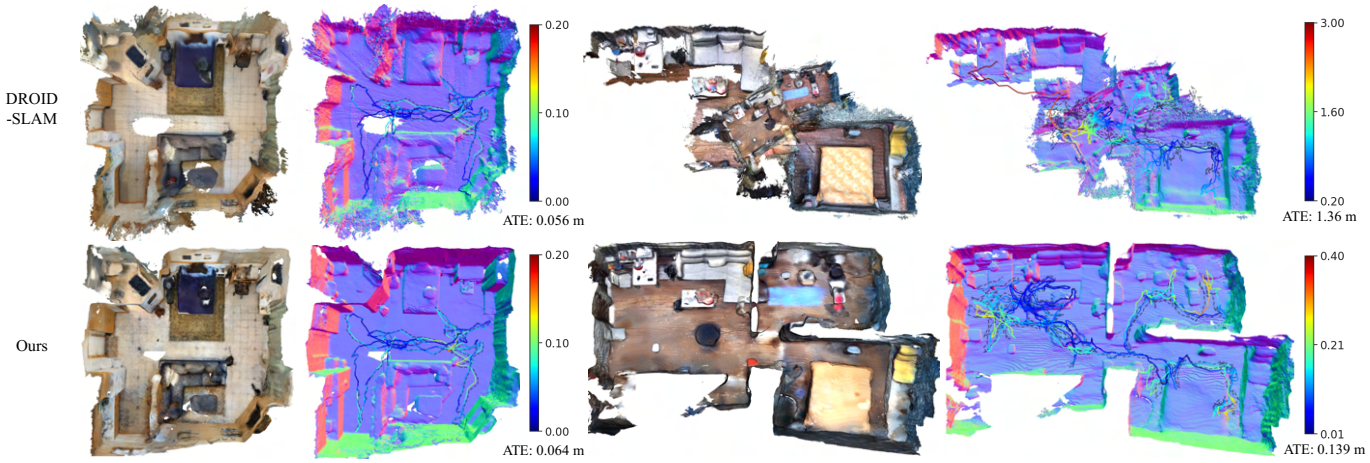


Fig. 6. Qualitative results of reconstructed maps and estimated trajectories colored by ATE. Our method can produce more complete maps while using less memory footprint, object surfaces are smoother with finer details. On the apartment dataset, DROID-SLAM fails to detect all loop closures and suffers from severe scale drift. Please check our attached demo videos[1] for the incremental mapping process.

TABLE I
RECONSTRUCTION EVALUATIONS ON REPLICA DATASET. BEST RESULTS ARE HIGHLIGHTED AS FIRST, SECOND, AND THIRD

		ro-0	ro-1	ro-2	of-0	of-1	of-2	of-3	of-4	Avg.
iMAP	Acc. [cm]↓	3.58	3.69	4.68	5.87	3.71	4.81	4.27	4.83	4.43
	Comp. [cm]↓	5.06	4.87	5.51	6.11	5.26	5.65	5.45	6.59	5.56
	Comp. Ratio [%]	83.90	83.40	75.50	77.70	79.60	77.20	77.30	77.60	79.00
iMODE	Acc. [cm]↓	7.40	6.40	9.30	6.60	11.80	11.40	9.40	8.00	8.78
	Comp. [cm]↓	13.50	10.10	19.20	9.70	17.00	14.50	11.80	15.40	13.90
	Comp. Ratio [%]	38.70	46.10	36.10	49.30	30.10	29.80	36.00	31.00	37.10
DROID	Acc. [cm]↓	12.18	8.35	3.26	3.01	2.39	5.66	4.49	4.65	5.50
	Comp. [cm]↓	8.96	6.07	16.01	16.19	16.20	15.56	9.73	9.63	12.29
	Comp. Ratio [%]	60.07	76.20	61.62	64.19	60.63	56.78	61.95	67.51	63.60
NICER	Acc. [cm]↓	2.53	3.93	3.40	5.49	3.45	4.02	3.34	3.03	3.65
	Comp. [cm]↓	3.04	4.10	3.42	6.09	4.42	4.29	4.03	3.87	4.16
	Comp. Ratio [%]	88.75	76.61	86.10	65.19	77.84	74.51	82.01	83.98	79.37
GO-SLAM	Depth L1 [cm]↓	-	-	-	-	-	-	-	-	4.39
	Acc. [cm]↓	4.60	3.31	3.97	3.05	2.74	4.61	4.32	3.91	3.81
	Comp. [cm]↓	5.56	3.48	6.90	3.31	3.46	5.16	5.40	5.01	4.79
	Comp. Ratio [%]	73.35	82.86	74.23	82.56	86.19	75.76	72.63	76.61	78.00
Ours	Depth L1 [cm]↓	3.61	2.07	4.63	3.66	1.91	3.39	5.07	4.68	3.63
	Acc. [cm]↓	3.21	3.74	3.16	3.87	2.60	4.62	4.25	3.53	3.62
	Comp. [cm]↓	3.25	3.08	4.09	5.29	8.83	4.42	4.06	3.72	4.59
	Comp. Ratio [%]	86.99	87.19	80.82	72.55	72.44	80.90	81.04	82.88	80.60

representation in $SE(3)$ and $Sim(3)$ groups. We use the pre-trained model from DROID-SLAM [17] and input images are resized to 400x536 to better align with the resolution of training images and enhance efficiency. For multiresolution features grids, we utilize the tiny-cuda-nn framework, which implements the fast fully-fused CUDA kernels to accelerate computing. We set 16 grid levels ranging from a base resolution 16 to a finest spacing of 4 cm. The hash table size is set to 2^{16} for all room-size datasets and increased to 2^{19} for Apartment data considering its larger dimensions. For map optimization, we sample a total of $N_{pixel} = 2048$ pixels in each iteration. Along each sampled pixel ray, we sample $M_d = 11$ plus $M_u = 32$ query points.

B. Evaluation on Replica dataset

Replica [30] comprises several high-quality reconstructions from real-world scenes. We use the synthesized sequences by [7] as the benchmark for comparison against prior works. We omit the trajectory evaluation for this dataset since both our method and previous works have already achieved excellent accuracy below 1 cm. Notably, neural field-based map has the scene completion capability. In alignment with [7] [10], we take the complete ground-truth (GT) models to also assess the reconstruction quality on unseen areas. Additionally, following the approach of [8], a convex hull is computed based on the keyframe poses and rendered depth maps, and mesh faces outside this are considered outliers.

As shown in Tab. I, our results exhibit superior performance in both accuracy and completeness. Notably, our results are comparable to the RGB-D approach iMAP [7] and significantly outperform the previous neural field-based RGB method iMODE [34]. We also compare with the dense-SLAM approach, specifically DROID-SLAM, where reconstruction is based on the TSDF integration of predicted depths. Furthermore, NICER-SLAM [10] and GO-SLAM [14], recent state-of-the-art methods, serve as another two strong baselines. While NICER-SLAM also employs monocular priors to facilitate neural scene reconstruction, it is unable to run in real-time nor achieve loop closure optimization. Fig. 5 shows qualitatively that our reconstruction are cleaner with more detailed geometry.

TABLE II
ATE [M] RESULTS ON SCANNET DATASET.

	Scene ID	0000	0059	0106	0169	0181	0207	Avg.
RGB-D	NICE-SLAM [8]	0.086	0.123	0.081	0.103	0.129	**0.056**	0.096
	Co-SLAM [11]	0.072	0.123	0.096	0.066	0.134	0.071	0.094
	ESLAM [12]	0.073	0.086	0.075	**0.065**	0.092	0.057	**0.074**
Mono.	GO-SLAM [14]	0.059	0.083	0.081	0.084	0.083	-	-
	DROID-SLAM [17] (VO)	0.145	0.281	0.088	0.180	0.089	0.102	0.148
	DROID-SLAM [17]	**0.056**	0.080	0.066	0.092	0.077	0.076	**0.074**
	Ours (VO)	0.144	0.267	0.100	0.155	0.093	0.099	0.143
	Ours	0.064	**0.072**	**0.065**	0.085	**0.076**	0.084	**0.074**

C. Evaluation on ScanNet dataset

We conduct further experiments on ScanNet [32] to verify our system with real-world datasets, which are notably more challenging due to their larger size and blurry images. We first evaluate the tracking performance by reporting the absolute

[1]https://youtu.be/lj4Ie1RBFBE?si=zB7XWqwS6egdEexL

TABLE III
QUANTITATIVE EVALUATION OF THE RECONSTRUCTION ON SCANNET. AVERAGED ON 4 SELECTED SEQUENCES. WE ALSO REPORT THE RUNTIME ON SCENE0059 BY EACH METHOD IN LAST COLUMN.

Method	Pose	Acc↓	Comp↓	Prec↑	Recall↑	F-score↑	Time[h]↓
ManhattanSDF [35]	GT	0.072	0.068	0.621	0.586	0.602	16.68
MonoSDF [27] (MLP)	GT	**0.031**	0.057	0.783	0.652	0.710	9.89
MonoSDF [27] (Grid)	GT	0.034	0.046	**0.796**	0.711	0.750	4.36
torch-ash [36]	GT	0.042	0.056	0.751	0.678	0.710	0.47
Ours	GT	0.042	**0.043**	0.776	**0.748**	**0.762**	**0.03**
DROID-SLAM [17]	SLAM	0.082	0.153	0.504	0.469	0.475	**0.02**
Ours	SLAM	**0.059**	**0.059**	**0.663**	**0.638**	**0.650**	0.03

trajectory error (ATE) metric. To ensure global consistency, both GO-SLAM [14] and DROID-SLAM [17] deploy expensive full BA to correct pose drift, whereas our can run the proposed $Sim(3)$-based pose graph BA efficiently in online manner. Tab. II shows that our approach achieves on-par accuracy to the strong baseline DROID-SLAM which employs offline full BA and even some RGB-D methods.

For reconstruction quality, both Tab. II and Fig. 6 show our system yields more accurate and more complete reconstructions than DROID-SLAM. Given the challenging nature of the ScanNet dataset, previous methods [27], [35], [36] have relied on GT poses and offline pipelines to circumvent the problem. For comparison, we run our proposed framework with the GT poses fixed in BA. Tab. II shows that our online system not only matches the accuracy of other offline methods but also achieves higher completeness.

D. Result on multi-room apartment scene

To test our approach quality on even larger scenes, we conduct experiments on the larger-scale apartment dataset [8] which has over 10k frames and traverses over multiple rooms. We calculate the ATE using the trajectory estimation by the offline RGB-D method [37] as reference. As a result, DROID-SLAM [17] fails to find all loop closures as the drift accumulates very fast and results in collapsed reconstruction (Fig. 6), whereas our approach can instantly close the loops once detected reaching a globally consistent map.

E. Ablation Study

Monocular priors. First, we investigate the effect of incorporating monocular priors into our system. Tab. IV shows that without either the normal or depth prior leads to degraded results. The normal prior loss plays a crucial role in improving both accuracy and completeness, whereas the depth prior primarily contributes to accuracy. We further assess the depth L1 metric in Tab. IV. Even after scale alignment, the monocular prior depth remains suboptimal due to its ill-posed nature. This is also evidenced by the experiment labeled 'w/o BA depth', which we replace the BA depth with the aligned prior depth during mapping. Nevertheless, the prior depth effectively boosts the BA depth estimation when combined with the proposed JDSA module. Finally, our neural map can render further improved depth with an averaged L1 error of 3.63 cm.

Furthermore, incorporating depth prior has a positive impact on tracking accuracy. As shown in Tab. II, our frontend (VO) achieves slightly better ATE than DROID-SLAM (VO). Arguably, this can be attributed to the improved depth estimation by the JDSA module, allowing the optical flow network to converge to more correspondences in later iterations.

Fig. 7. Ablation study on uniform sampling vs. depth certainty guided pixel sampling.

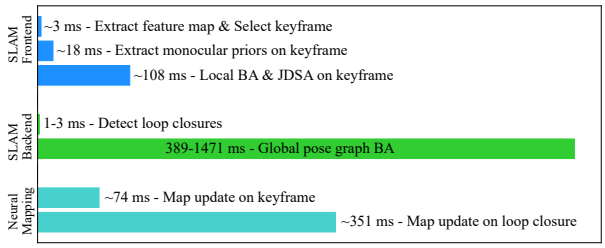


Fig. 8. Runtime analysis of the key components in each process.

TABLE IV
IMPACT OF DIFFERENT COMPONENTS BASED ON RECONSTRUCTION METRICS (LEFT) AND DEPTH L1 METRIC (RIGHT). NUMBERS ARE AVERAGED OVER 8 SEQUENCES OF REPLICA DATASET.

	Acc[cm]↓	Comp[cm]↓	Comp. Ratio[%]↑
w/o BA depth	8.23	7.83	64.32
w/o $\mathcal{L}_{normal}$	4.63	4.75	76.80
w/o depth prior	4.23	3.98	81.10
Ours	**3.56**	**3.60**	**82.95**

Depth type	L1[cm]↓
Aligned prior	9.36
BA only	6.11
BA+JDSA	4.81
Rendered	**3.63**

Depth guided sampling. Fig. 7 presents the reconstruction results using different pixel sampling methods. The uniform pixel sampling method as employed by prior works [14], [10] faces challenges to accurately reconstruct object boundaries and small objects. One potential cause is the high ratio of noisy depth estimations which can falsely carve out the surfaces, leading to conflicts between the SDF loss and free-space loss during map optimization. In contrast, as shown in Fig. 4, we observe that the pixels on small objects and object edges typically attain higher confidences from the optical flow network and corresponding higher certainties. Utilizing the depth certainty-guided pixel sampling method allows for more pixel sampling in these regions and helps the neural fields in distinguishing the contradictory supervisions between the SDF loss and free-space loss.

F. Runtime Analysis

All experiments are carried out on a desktop PC with an Intel i9 CPU and an Nvidia RTX 4090 GPU. We report the runtime of key components of the system. Fig. 8 shows that majority time is consumed to process keyframes. The frontend can handle up to 5 keyframes per second. The mapping process can rapidly update with camera movement, requiring only 74 ms for 10 optimization iterations. Notably the loop closing and global optimization run in parallel, taking a few hundreds milliseconds up to around 1.5 seconds. On Replica, our system operates at an average speed of 25 fps. On ScanNet, the speed

reduces to 15 fps due to the faster motion and more keyframes need to be processed.

V. CONCLUSIONS

In this letter, we present our integration of deep learning-based dense SLAM with neural field representation to reconstruct high-quality scene geometry in real-time. We jointly adjust depth maps and the scales of monocular priors to not only solve the scales of the priors but only enable accurate depth estimation. The estimated depth aids efficient ray sampling and optimization of the neural fields. Consequently, the neural map can be incrementally and continuously constructed in a live manner. To maintain global consistency, our system employs the proposed $Sim(3)$ based pose graph BA when loop closures are detected, correcting both pose and scale drifts. We show that our neural map can instantly adapt to these global updates by loop closures. Compared to previous methods, our approach achieves the state-of-the-art accuracy and completeness.

REFERENCES

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007, pp. 225–234.

[2] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[3] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[4] C. Tang and P. Tan, "Ba-net: Dense bundle adjustment network," *arXiv preprint arXiv:1806.04807*, 2018.

[5] Z. Teed and J. Deng, "Deepv2d: Video to depth with differentiable structure from motion," *arXiv preprint arXiv:1812.04605*, 2018.

[6] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[7] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "imap: Implicit mapping and positioning in real-time," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6229–6238.

[8] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 786–12 796.

[9] X. Yang, H. Li, H. Zhai, Y. Ming, Y. Liu, and G. Zhang, "Vox-fusion: Dense tracking and mapping with voxel-based neural implicit representation," in *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2022, pp. 499–507.

[10] Z. Zhu, S. Peng, V. Larsson, Z. Cui, M. R. Oswald, A. Geiger, and M. Pollefeys, "Nicer-slam: Neural implicit scene encoding for rgb slam," *arXiv preprint arXiv:2302.03594*, 2023.

[11] H. Wang, J. Wang, and L. Agapito, "Co-slam: Joint coordinate and sparse parametric encodings for neural real-time slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 293–13 302.

[12] M. M. Johari, C. Carta, and F. Fleuret, "Eslam: Efficient dense slam system based on hybrid representation of signed distance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 408–17 419.

[13] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005.

[14] Y. Zhang, F. Tosi, S. Mattoccia, and M. Poggi, "Go-slam: Global optimization for consistent 3d instant reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023.

[15] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 international conference on computer vision*. IEEE, 2011, pp. 2320–2327.

[16] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[17] Z. Teed and J. Deng, "DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 558–16 569, 2021.

[18] X. Yang, L. Zhou, H. Jiang, Z. Tang, Y. Wang, H. Bao, and G. Zhang, "Mobile3drecon: real-time monocular 3d reconstruction on a mobile phone," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 12, pp. 3446–3456, 2020.

[19] L. Koestler, N. Yang, N. Zeller, and D. Cremers, "Tandem: Tracking and dense mapping in real-time using deep multi-view stereo," in *Conference on Robot Learning*. PMLR, 2022, pp. 34–45.

[20] H. Li, X. Gu, W. Yuan, L. Yang, Z. Dong, and P. Tan, "Dense rgb slam with neural implicit maps," *arXiv preprint arXiv:2301.08930*, 2023.

[21] C.-M. Chung, Y.-C. Tseng, Y.-C. Hsu, X.-Q. Shi, Hua, and W. H. Hsu, "Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.

[22] A. Rosinol, J. J. Leonard, and L. Carlone, "Nerf-slam: Real-time dense monocular slam with neural radiance fields," *arXiv preprint arXiv:2210.13641*, 2022.

[23] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (ToG)*, vol. 41, no. 4, pp. 1–15, 2022.

[24] A. Eftekhar, A. Sax, J. Malik, and A. Zamir, "Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 786–10 796.

[25] W. Zhang, S. Wang, X. Dong, R. Guo, and N. Haala, "Bamf-slam: Bundle adjusted multi-fisheye visual-inertial slam using recurrent field transforms," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 6232–6238.

[26] W. Niemeier, "Ausgleichungsrechnung," in *Ausgleichungsrechnung*. de Gruyter, 2008.

[27] Z. Yu, S. Peng, M. Niemeyer, T. Sattler, and A. Geiger, "Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction," *Advances in neural information processing systems*, vol. 35, pp. 25 018–25 032, 2022.

[28] D. Azinović, R. Martin-Brualla, D. B. Goldman, M. Nießner, and J. Thies, "Neural rgb-d surface reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6290–6301.

[29] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume rendering of neural implicit surfaces," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4805–4815, 2021.

[30] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[31] A. Rosinol, J. J. Leonard, and L. Carlone, "Probabilistic volumetric fusion for dense monocular slam," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 3097–3105.

[32] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5828–5839.

[33] Z. Teed and J. Deng, "Tangent space backpropagation for 3d transformation groups," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 10 338–10 347.

[34] H. Matsuki, E. Sucar, T. Laidow, K. Wada, R. Scona, and A. J. Davison, "imode: Real-time incremental monocular dense mapping using neural field," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 4171–4177.

[35] H. Guo, S. Peng, H. Lin, Q. Wang, G. Zhang, H. Bao, and X. Zhou, "Neural 3d scene reconstruction with the manhattan-world assumption," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5511–5520.

[36] W. Dong, C. Choy, C. Loop, O. Litany, Y. Zhu, and A. Anandkumar, "Fast monocular scene reconstruction with global-sparse local-dense grids," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4263–4272.

[37] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5556–5565.