

DI-Fusion: Online Implicit 3D Reconstruction with Deep Priors

Jiahui Huang Shi-Sheng Huang Haoxuan Song Shi-Min Hu*

BNRist, Department of Computer Science and Technology, Tsinghua University, Beijing

Abstract

Previous online 3D dense reconstruction methods often cost massive memory storage while achieving unsatisfactory surface quality mainly due to the usage of stagnant underlying geometry representation, such as TSDF (truncated signed distance functions) or surfels, without any knowledge of the scene priors. In this paper, we present DI-Fusion (Deep Implicit Fusion), based on a novel 3D representation, called Probabilistic Local Implicit Voxels (PLIVoxs), for online 3D reconstruction using a commodity RGB-D camera. Our PLIVox encodes scene priors considering both the local geometry and uncertainty parameterized by a deep neural network. With such deep priors, we demonstrate by extensive experiments that we are able to perform online implicit 3D reconstruction achieving state-of-the-art mapping quality and camera trajectory estimation accuracy, while taking much less storage compared with previous online 3D reconstruction approaches.

1. Introduction

Online 3D dense reconstruction has achieved great progress in the past ten years [3, 6, 11, 20, 31, 32, 42] along with the popularity of commercial depth sensors such as Kinect, Xtion Pro, Intel RealSense, etc. Most of the previous depth fusion approaches focus on the globally consistent 3D reconstruction with bundle adjustment [3, 11] or loop closure [42] techniques. However, the underlying representation for 3D scene itself has hardly made much development since the success of VoxelHashing [32] with Signed Distance Function (SDF) integration on sparse set of voxels [9]. This leads to a main drawback of previous depth fusion systems that often costs huge amount of memory storage even for moderate size of 3D scenes. Besides, the geometry quality could be unsatisfactory with non-complete regions or objects [12] due to geometric uncertainties caused by sensor noise or scan ambiguity such as view occlusion.

On the other hand, recent efforts in deep geometry learning community have demonstrated the power of im-

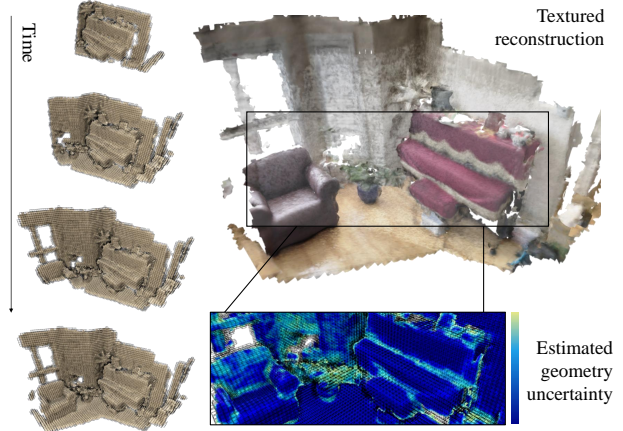


Figure 1. **DI-Fusion** incrementally builds up a continuous 3D scene from an RGB-D sequence. The tracking and mapping algorithm are fully based on our novel local deep implicit scene representation, where both the geometry and its uncertainty scene priors are estimated.

PLICIT geometric representation parameterized by neural networks [8, 28, 33]. By representing the geometry as a continuous implicit function, the underlying shape can be extracted at arbitrary resolution, which effectively enhances surface reconstruction quality. These methods are also efficient since the network structure used to regress implicit function only consists of simple fully connected layers. Another important feature of such deep implicit representation is the capability to encode geometric priors, which benefit many applications such as shape interpolation or reconstruction [4, 33]. This capability can be generalized to scene priors by decomposing and encoding the implicit fields in local voxels, leading to impressive scene completion or reconstruction [4, 19].

The power of such deep implicit representation motivates us to incorporate it into online 3D dense reconstruction systems. By encoding meaningful scene priors with a continuous function, we can achieve improved surface reconstruction as well as accurate camera trajectory. However, several challenges need to be overcome before the deep implicit representation is successfully applied in an online fusion scenario: (1) geometric uncertainty need to be

*corresponding author.

explicitly encoded in the deep implicit representation to against sensor noise or view occlusion; (2) An accurate camera tracking based on such implicit representation is essential for depth fusion, which remains unknown yet; (3) An efficient surface mapping which incrementally integrates new observations directly based on the deep implicit representation is also missing.

We hence respond with DI-Fusion, the first online 3D reconstruction system with tracking and mapping modules fully supported by deep implicit representations. To address the above challenges, we first extend the original local implicit grids [4, 19] and adapt it into a novel **Probabilistic Local Implicit Voxel (PLIVox)**, which encodes not only scene geometry but also the *uncertainty* with one deep neural network. We show that such an additional uncertainty encode is extremely useful during the online depth fusion. Based on our PLIVox representation, we devise an approximate gradient for solving the camera tracking problem efficiently. Besides, by formulating the deep implicit representation in a tailored encoder-decoder network, we perform geometry integration on the domain of latent vectors achieving high quality surface mapping than TSDF-based fusion approaches [9] in an efficient way. We extensively evaluated our approach on public 3D RGB-D dataset (ICL-NUIM [18], ScanNet dataset [10]), showing state-of-the-art or improved tracking and mapping quality compared to previous representations.

2. Related Works

Online 3D Reconstruction. The success of KinectFusion [31] inspired a lot of works on online 3D reconstruction. Early efforts mainly focus on efficient data structures to organize discrete voxels or surfels to enable large-scale 3D reconstruction, such as OctreeFusion [43], VoxelHashing [32], Scalable-VoxelHashing [6], ElasticFusion [21, 42]. Subsequent efforts turn to bundle adjustment [11, 20, 41] or loop closure [32] techniques for globally consistent 3D reconstruction. Despite the rapid development of this field, only a few works focus on the underlying 3D scenes representations. Compared to previous works, our PLIVox, a deep implicit representation that effectively encodes scene priors, can efficiently improve the surface reconstruction quality.

Deep Visual SLAM. Our work is also relevant to visual SLAM methods. The early techniques, such as MonoSLAM [13], LSD-SLAM [15], ORB-SLAM2 [30], and DSO [14], propose to perform camera tracking using sparse features or direct intensity RGB data. With the advantage of deep learning, recent visual SLAM techniques incorporate deep neural networks for more accurate representation of sparse features (DeepTrack [17]), monocular depth observation (CNN-SLAM [40], CodeSLAM [2]), or

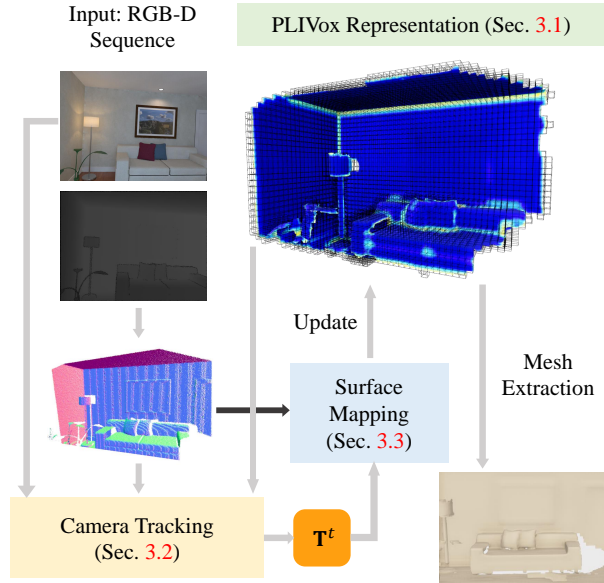


Figure 2. **Overview.** We represent the reconstructed 3D scene with PLIVoxs (Sec. 3.1) encoding the deep implicit representation. Given input RGB-D frames, we first estimate the camera pose T^t by finding the best alignment between the current depth point cloud and the map (Sec. 3.2), then the depth observations are efficiently integrated (Sec. 3.3) for surface mapping. Scene mesh can be extracted any time on demand at any resolution. Note that both the camera tracking and surface mapping are performed directly on the deep implicit representation.

object instance (Fusion++ [27], NodeSLAM [39]). Different from these works, our system integrates deep priors in the 3D map domain locally and is hence more robust and generalizable.

Implicit Representation. The use of implicit function for geometric reconstruction can be dated back to [9], where SDF values are stored in a set of occupied voxels describing the surface. However, even though the implicit function is continuous, the simple discretization [6, 32] introduce drawbacks in surface reconstruction quality and memory storage. As an effort to overcome such drawbacks, [23, 26] propose to model the map using Gaussian Process and perform Bayesian map updates incrementally. In contrast, the use of implicit representation has triggered much interest in deep learning community these years [28, 33]. Typical applications include: part segmentation [7], rendering [24, 29], non-linear fitting [38], meta-learning [37], offline reconstruction [4, 19] etc. Nevertheless, to the best of our knowledge, our DI-Fusion is among the first efforts to incorporate such an implicit representation with deep priors into an online 3D fusion framework.

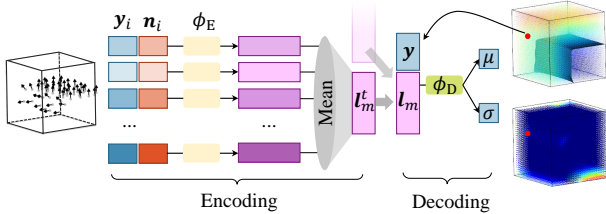


Figure 3. The structure of our encoder-decoder neural network Φ . The encoding and decoding process within one PLIVox is shown. The arrow between encoding and decoding denotes incremental latent vector update if the latent vector from last frame is available.

3. Method

Overview. Given a sequential RGB-D stream, our DI-Fusion incrementally builds up a 3D scene based on a novel **Probabilistic Local Implicit Voxels (PLIVox)** representation. Different from previous approaches using discrete voxels without any scene priors, our PLIVox is implicitly parameterized by a neural network and encodes useful local scene priors effectively (Sec. 3.1). Based on this representation, we introduce how to robustly perform camera tracking for each frame (Sec. 3.2) and how to perform efficient incremental surface mapping (Sec. 3.3). As a final step mesh can be extracted at arbitrary resolution thanks to the continuous representation, compared to previous approaches which can only reconstruct at a predefined resolution. This overview of our DI-Fusion is illustrated in Fig. 2.

3.1. PLIVox Representation

The PLIVox set for the reconstructed 3D scene is denoted as $\mathcal{V} = \{v_m = (c_m, l_m, w_m)\}$, with $c_m \in \mathbb{R}^3$ being voxel centroid, $l_m \in \mathbb{R}^L$ being the latent vector encoding the scene priors and $w_m \in \mathbb{N}$ being the observation weight. For an arbitrary point measurement $x \in \mathbb{R}^3$, we can efficiently query its corresponding PLIVox index $m(x)$ using simple division and rounding operations $m(x) : \mathbb{R}^3 \mapsto \mathbb{N}^+$. The local coordinate of x in $v_{m(x)}$ is calculated as $y = \frac{1}{a}(x - c_{m(x)}) \in [-\frac{1}{2}, \frac{1}{2}]^3$, with a being the voxel size.

Probabilistic Signed Distance Function. Different from the previous approaches representing the underlying 3D surface with signed distance function, we represent it using a *probabilistic* signed distance function, where the output at every position y is not a SDF but a SDF distribution $s \sim p(\cdot|y)$. In this way, the probabilistic signed distance function encodes the surface geometry and geometric uncertainty at the same time. Here we model the SDF distribution as a canonical Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with μ and σ being the mean and standard deviation respectively. For a more compact representation, we encode the probabilistic signed distance function with a latent vector l_m using an encoder-decoder deep neural network Φ .

Encoder-Decoder Neural Network. The encoder-decoder neural network $\Phi = \{\phi_E, \phi_D\}$ consists of encoder sub-network ϕ_E and decoder sub-network ϕ_D (Fig. 3) that share weights with all PLIVoxs.

The target of the encoder ϕ_E is to convert the measurements from each depth point observation at frame t to *observation* latent vectors l_m^t . Specifically, for all the RGB-D point measurements located in a PLIVox, ϕ_E takes in the point measurement’s local coordinate y and normal direction n , and transforms them to an L -dimensional feature vector $\phi_E(y, n)$ using only FC (Fully Connected) layers. Then the feature vectors from multiple points are aggregated to one latent vector l_m^t using a mean-pooling layer (Fig. 3). Here the normal direction n is required to eliminate the orientation ambiguity within each PLIVox so the sign of SDF can be inferred by the network.

For the decoder ϕ_D , the concatenation of the local coordinate y and the latent vector l_m are taken as input and the output is a 2-tuple $\{\mu_D, \sigma_D\}$, which represents the Gaussian parameters of the probabilistic signed distance function $p(\cdot|y) \sim \mathcal{N}(\mu_D, \sigma_D^2)$ at position y . Note that the two latent vectors l_m^t and l_m in ϕ_E and ϕ_D are different latent vectors. While the *observation* latent vector l_m^t encodes the RGB-D observations at frame t , the *geometry* latent vector l_m fuses the previous l_m^t and is stored in each PLIVox used for decoding. Both the observation latent vector and the geometry latent vector have the same dimension, and the geometry latent vector can be updated by the observation latent vector as described in Sec. 3.3.

Network Training. We train the ϕ_E and ϕ_D jointly in an end-to-end fashion, setting $l_m^t \equiv l_m$. We adopt the training strategy similar to Conditional Neural Process (CNP [16]) but extend it to 3D domain. Specifically, we first build up two training datasets: (1) $\mathcal{S} = \{\mathcal{S}_m\}$ for encoder, which is a set of tuples $\mathcal{S}_m = \{(y_i, n_i)\}$ for each PLIVox v_m with points y_i and n_i sampled from the scene surface; (2) $\mathcal{D} = \{\mathcal{D}_m\}$ for the decoder, which consists of tuples $\mathcal{D}_m = \{(y_i, s_{gt}^i)\}$ where points y_i are randomly sampled within a PLIVox using a strategy similar to [33] with s_{gt}^i being the SDF at point y_i . We give more details about how to build up such two training datasets in Sec. 4. During training, we feed \mathcal{S}_m to the encoder for latent vector l_m and concatenate the latent vector with each y_i in \mathcal{D}_m to obtain predicted SDF mean and standard deviation. The goal of training is to maximize the likelihood of the dataset \mathcal{D} for all training PLIVoxs. Specifically, the loss function \mathcal{L}_m for each PLIVox v_m is written as:

$$\mathcal{L}_m = - \sum_{(y_i, s_{gt}^i) \in \mathcal{D}_m} \log \mathcal{N}(s_{gt}^i; \mu_D(y_i, l_m), \sigma_D^2(y_i, l_m)), \quad (1)$$

$$l_m = \frac{1}{|\mathcal{S}_m|} \sum_{(y_i, n_i) \in \mathcal{S}_m} \phi_E(y_i, n_i). \quad (2)$$

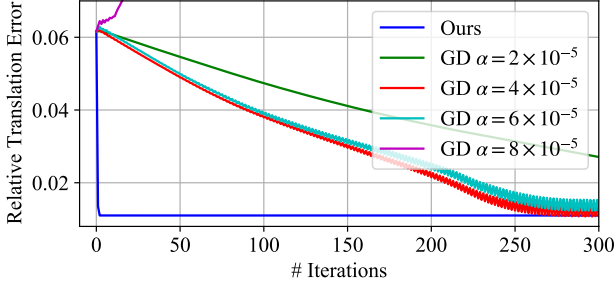


Figure 4. Comparison of converge between different optimization strategies for camera tracking. ‘GD’ is short for gradient descent on Eq (1), with α being the different learning rate. ‘Ours’ represents the approximate gradient we used in our camera tracking.

Additionally we regularize the norm of the latent vector with a l_2 -loss which reflects the prior distributions of \mathbf{l}_m . The final loss function \mathcal{L} we used for training is:

$$\mathcal{L} = \sum_{v_m \in \mathcal{V}} \mathcal{L}_m + \delta \|\mathbf{l}_m\|^2. \quad (3)$$

3.2. Camera Tracking

With our PLIVox encoding of the scene priors including both the scene geometry and uncertainty, we propose a frame-to-model [31] camera tracking method. We claim that the learnt deep priors have enough information of the 3D scene for an accurate camera pose estimation, without need of extra sparse features as in [3, 11]. We propose to formulate the probabilistic signed distance function as objective function for camera pose estimation, with an approximate gradient for the objective function over camera pose, which makes it converge fast enough during the optimization. What’s more, our network is efficient in decoding the probabilistic signed distance function, leading to an efficient online tracking performance.

Tracking. We denote the RGB-D observation at frame t as $\mathcal{O}^t = \{\mathcal{I}^t, \mathcal{D}^t\}$ with \mathcal{I}^t and \mathcal{D}^t being the intensity and depth data. Given camera intrinsic parameters, the depth measurement \mathcal{D}^t can be re-projected to 3D as point measurements $\mathcal{P}^t = \pi'(\mathcal{D}^t)$, where π is the projection function and π' is its inverse. Our goal is to estimate \mathcal{O}^t ’s camera pose $\mathbf{T}^t \in SE(3)$ by optimizing the relative pose $T(\boldsymbol{\xi}^t) = \exp((\boldsymbol{\xi}^t)^\wedge)$ ($\boldsymbol{\xi}^t \in se(3)$ [1]) between \mathcal{O}^t and \mathcal{O}^{t-1} , i.e. $\mathbf{T}^t = \mathbf{T}^{t-1}T(\boldsymbol{\xi}^t)$. The following objective function is minimized in our system:

$$E(\boldsymbol{\xi}^t) = E_{\text{sdf}}(\boldsymbol{\xi}^t) + wE_{\text{int}}(\boldsymbol{\xi}^t), \quad (4)$$

where $E_{\text{sdf}}(\boldsymbol{\xi}^t)$ and $E_{\text{int}}(\boldsymbol{\xi}^t)$ are the SDF term and intensity term respectively, and w is a weight parameter. The objective function $E(\boldsymbol{\xi}^t)$ can be efficiently optimized using Gauss-Newton solver.

SDF Term $E_{\text{sdf}}(\boldsymbol{\xi}^t)$. The goal of our SDF term is to perform frame-to-model alignment of the point measurements \mathcal{P}^t of \mathcal{O}^t to the on-surface geometry decoded by \mathcal{V} . Different from traditional point-to-plane Iterative Closest Point (ICP) methods with projective association, as our map representation is fully backbone by implicit functions, we choose to minimize the signed distance value of each point in \mathcal{P}^t when transformed by the optimized camera pose. Thus we design the objective function as:

$$E_{\text{sdf}}(\boldsymbol{\xi}^t) = \sum_{\mathbf{p}^t \in \mathcal{P}^t} \rho(r(G(\boldsymbol{\xi}^t, \mathbf{p}^t))), \quad (5)$$

$$G(\boldsymbol{\xi}^t, \mathbf{p}^t) = \mathbf{T}^{t-1}T(\boldsymbol{\xi}^t)\mathbf{p}^t, \quad r(\mathbf{x}) = \frac{\mu_{\text{D}}(\mathbf{x}, \mathbf{l}_m(\mathbf{x}))}{\sigma_{\text{D}}(\mathbf{x}, \mathbf{l}_m(\mathbf{x}))},$$

where $\rho(\cdot)$ is the Huber robust function, $\{\mu_{\text{D}}, \sigma_{\text{D}}\}$ represents the probabilistic signed distance function decoded by the latent vector \mathbf{l}_m from the point measurements.

One important step to optimize the SDF term is the computation of $r(\cdot)$ ’s gradient with respect to $\boldsymbol{\xi}^t$, i.e. $\frac{\partial r}{\partial \boldsymbol{\xi}^t}$. Note that σ_{D} and μ_{D} are the output for decoder ϕ_{D} , so $r(\cdot)$ is an highly non-linear composite function of decoder ϕ_{D} , which will lead to poor local approximation. We instead propose to treat σ_{D} to be constant during the local linearization, which will efficiently improve convergence during the optimization. Another benefit is that we can control the influence of σ_{D} up to the zeroth order approximation, achieving robust tracking performance even when σ_{D} prediction is wrong. Specifically, the approximate gradient is computed by:

$$\frac{\partial r}{\partial \boldsymbol{\xi}^t} = \frac{1}{\sigma_{\text{D}}} \frac{\partial \mu_{\text{D}}(\cdot, \mathbf{l}_m(\mathbf{x}))}{\partial \mathbf{x}} (\mathbf{R}^{t-1})^\top (T(\boldsymbol{\xi}^t)\mathbf{p}^t)^\odot, \quad (6)$$

where \mathbf{R}^{t-1} is the rotation part of \mathbf{T}^{t-1} and $\frac{\partial \mu_{\text{D}}}{\partial \mathbf{x}}$ can be efficiently computed using back-propagation through the decoder network. Also note that $m(\mathbf{x})$ is an indicator function which keeps fixed during the optimization, we doesn’t need to compute its gradient. As an empirical proof of the effectiveness of the approximation, we show in Fig. 4 the comparison of convergence property between our method and a first-order gradient decent (GD) method which performs the update step as $\boldsymbol{\xi}^t \leftarrow \boldsymbol{\xi}^t - \alpha \frac{\partial \sum \mathcal{L}_m}{\partial \boldsymbol{\xi}^t}$. The Y-axis represents the relative translation error of $\boldsymbol{\xi}^t$ between ground-truth and estimated pose. Our approximate gradient enables convergence within a few iterations, while other gradients need a few hundred iterations to achieve comparable error, demonstrating the efficiency and effectiveness of such an approximation scheme.

Intensity Term $E_{\text{int}}(\boldsymbol{\xi}^t)$. We also add a photometric error term between the corresponding intensity data (called

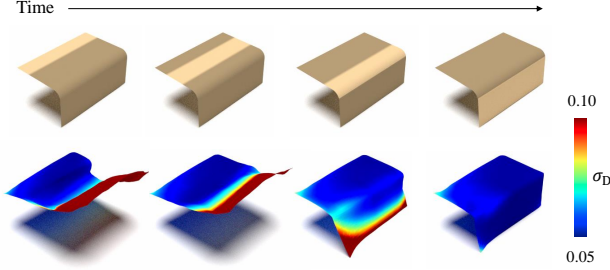


Figure 5. A tiny example demonstrating the effect of incremental integration. The top row shows ground-truth underlying geometry and at each time step, we sample point from the region with lighter color and fuse it into our PLIVox using Eq (8). The bottom row shows corresponding geometry after each frame is integrated. The color of the mesh is coded by σ_D , reflecting the uncertainty on the surface.

intensity term) defined as:

$$E_{\text{int}}(\xi^t) = \sum_{\mathbf{u} \in \Omega} (\mathcal{I}^t[\mathbf{u}] - \mathcal{I}^{t-1}[\pi(T(\xi^t)\pi'(\mathbf{u}, \mathcal{D}^t[\mathbf{u}])))])^2, \quad (7)$$

where Ω is the image domain. This intensity term takes effect when the SDF term fails in areas with fewer geometric details such as wall or floor.

3.3. Surface Mapping

After the camera pose of RGB-D observation \mathcal{O}^t is estimated, we need to update the mapping from observation \mathcal{O}^t based on the deep implicit representation, by fusing new scene geometry from new observations with noise, which is also referred to as geometry integration. Besides, we also describe an optional mesh extraction step which extracts the on-surface watertight mesh with textures for visualization.

Geometry Integration. Instead of updating SDF as previous works [9], we propose to update our deep implicit representation in the domain of *latent vectors*. We perform the geometry integration by updating the geometry latent vector \mathbf{l}_m with the observation latent vector \mathbf{l}_m^t encoded by the point measurements \mathcal{P}^t . More specifically, we first transform \mathcal{P}^t according to \mathbf{T}^t and then estimate the normal of each point measurement, thus obtaining $\mathcal{X}^t = \{(x_i, \mathbf{n}_i)\}$. In each PLIVox v_m , we calculate the point measurements $\mathcal{Y}_m^t \subset \mathcal{X}^t$ located in such PLIVox and compute the observation latent vector using $\mathbf{l}_m^t = \frac{1}{w_m^t} \sum_{(\mathbf{y}, \mathbf{n}) \in \mathcal{Y}_m^t} \phi_E(\mathbf{y}, \mathbf{n})$. The geometry latent vector \mathbf{l}_m within such PLIVox v_m is then updated as:

$$\mathbf{l}_m \leftarrow \frac{\mathbf{l}_m w_m + \mathbf{l}_m^t w_m^t}{w_m + w_m^t}, \quad w_m \leftarrow w_m + w_m^t, \quad (8)$$

where the weight w_m^t is set to the number of points within the PLIVox as $|\mathcal{Y}_m^t|$. In this way, our geometry integration

is much more efficient than the previous approaches, since we only need to update the latent vectors within a PLIVox but not the SDFs of massive individual voxels as in Voxel-Hashing [32]. The observation latent vector from forward pass of ϕ_E is also very efficient to compute, which make our geometry integration fast enough during the online surface mapping without latent vector initialization and optimization used in previous auto-decoder methods [4, 33]. Fig. 5 shows a tiny example demonstrating the effect of incremental geometry integration using our method.

In the meantime, benefit from the learnt deep priors, we do not need to perform integration for every frame. Instead, we choose to integrate sparse frames sampled from every N incoming frames, which improve system efficiency largely while still maintaining tracking accuracy.

Mesh Extraction. Since we keep the *continuous* signed distance function encoded as latent vectors within each PLIVox, we can extract the geometry surface at arbitrary resolution if necessary, which is different from previous mesh extraction methods which are only at predefined resolutions. Given a desired resolution during the extraction, we divide each PLIVox into equally-spaced volumetric grids and query the SDFs for each grid with the decoder ϕ_D using the PLIVox’s latent vector. Then the final on-surface mesh can be extracted with Marching Cubes [25]. Here, to maintain the *continuity* across PLIVox boundaries, we double each PLIVox’s domain such that the volumetric grids between neighboring PLIVoxs overlap with each other. The final SDF of each volumetric grid is trilinearly interpolated with SDFs decoded from the overlapping PLIVoxs [19]. For textures, we simply assign the vertices on the extracted mesh with texture colors averaged from the nearest point measurements projected from multiple observations.

4. Experiments

4.1. System Implementation

To train an effective encoder-decoder neural network for descriptive deep priors, we build up two training datasets (\mathcal{S} and \mathcal{D} for encoder and decoder, respectively, *c.f.* Sec. 3.1), on ShapeNet dataset [5] which contains a large variety of 3D shapes with rich local geometry details. We employ the 6 categories of the 3D shapes from the ShapeNet dataset, *i.e.* bookshelf, display, sofa, chair, lamp and table, and for each category we sample 100 3D shapes. Each shape is then divided into PLIVoxs with voxel size $a = 0.1\text{m}$. For each PLIVox v_m , we randomly sample $n_d = 4096$ (\mathbf{y}_i, s_{gt}^i) tuples for \mathcal{D}_m , and the count of $(\mathbf{y}_i, \mathbf{n}_i)$ tuple n_s for each \mathcal{S}_m is randomly chosen from 16 to 128. We further augment \mathcal{D}_m and \mathcal{S}_m by adding random jitter for positions \mathbf{y}_i and perturbation for normal direction \mathbf{n}_i so as to train the encoder-decoder neural network with enough robustness against depth observation noise.

Table 1. Comparison of ATE on ICL-NUIM [18] benchmark (measured in centimeters).

	lr kt0	lr kt1	lr kt2	lr kt3
DVO-SLAM [22]	10.4	2.9	19.1	15.2
Surfel Tracking [21]	1.7	1.0	2.2	43.2
TSDF Tracking [35]	4.5	2.1	1.3	12.5
Ours (w/o Prob)	1.3	1.8	2.6	15.2
Ours (max)	1.4	2.0	2.4	7.1
Ours	1.1	1.4	2.6	6.2

Table 2. Comparison of surface error on ICL-NUIM [18] benchmark (measured in centimeters).

	lr kt0	lr kt1	lr kt2	lr kt3
DVO-SLAM [22]	3.2	6.1	11.9	5.3
Surfel Tracking [21]	1.1	0.7	1.0	22.5
TSDF Tracking [35]	0.6	1.0	0.8	11.7
Ours (w/o Prob)	0.7	2.0	1.2	4.8
Ours (max)	0.8	1.8	1.2	4.8
Ours	0.6	1.5	1.1	4.5

We set the length of the latent vector as $L = 29$ in both encoder ϕ_E and decoder ϕ_D sub-networks. The encoder ϕ_E contains 5 fully connected (FC) layers, with layer size set as 6-32-64-256-29 respectively. The decoder contains 6 FC layers with layer size set as 32-128-128-128-128-2. We choose to use the Adam optimizer for training the encoder-decoder network with an initial learning rate set as 10^{-3} . For the regularization on the loss function \mathcal{L} , we set $\delta = 10^{-2}$.

We manage the PLIVoxs using a similar mechanism as [32] and allocate PLIVoxs only when there are enough point measurements gathered (16 in our experiments). Our DI-Fusion system is implemented using PyTorch framework [34]. With its highly-optimized auto-differentiation, the jacobian matrix of our tracking term (c.f. Eq (6)) can be efficiently computed.

4.2. Quantitative Evaluations

In this subsection we demonstrate the effectiveness of DI-Fusion and its core component, *i.e.* the PLIVox representation, on ICL-NUIM dataset [18], which contains both ground-truth camera trajectory and 3D scene reconstruction to evaluate the accuracy of the camera pose estimation for depth fusion approaches. We adopt the Absolute Trajectory Error (ATE) as metric to evaluate the accuracy of camera pose estimation and the surface error [42] to evaluate the reconstructed 3D surface quality.

Since our DI-Fusion does not contain loop closure component, for a fair comparison we choose to compare with previous depth fusion approaches using TSDF or Surfel representation which do not contain loop closure either. For TSDF-based camera tracking, we adopt the frame-to-model ICP tracking method as implemented in [35], denoted as

‘TSDF tracking’. For surfel-based camera tracking, we adopt the method in [21, 42], denoted as ‘surfel tracking’. Additionally, we also compare to another baseline approach, *i.e.* DVO-SLAM [22], which performs frame-to-frame camera tracking directly *without* the aid of any 3D scene representations.

As shown in Tab. 1, our approach achieves the lowest ATE in sequence `lr_kt0` and `lr_kt3`. This is mainly due to two reasons: (1) The deep priors learnt in our PLIVox provide more continuous underlying surface prediction than TSDF or surfel, which provides smoother gradient estimation for camera tracking (Sec. 3.2). (2) The uncertainty estimated by our network also effectively filter out noisy RGB-D observations, thus leading to more accurate camera tracking than TSDF or surfel-based approaches. Though there are not too many scene priors (such as detailed geometry and objects priors) in the 3D scenes of sequence `lr_kt1` and `lr_kt2`, our approach can still achieve comparable ATE for the camera pose estimation accuracy. Note that, although TSDF tracking achieves better ATE than our approach in sequence `lr_kt2`, to support such tracking it consumes nearly an order of magnitude more memory storage than ours (*c.f.* Fig. 6 bottom-left).

The surface error comparison shown in Tab. 2 also verifies the conclusion above. Benefit from the learnt deep priors provided by our PLIVox representation, our approach can perform better (at least comparable) surface error than TSDF or surfel based camera tracking, with much less memory storage usage.

4.3. Qualitative Results

We compare the visual quality of the reconstructed 3D surfaces between our approach and the other two camera tracking approaches (TSDF tracking and surfel tracking). For a fair comparison we implement TSDF tracking with two versions: (1) A low resolution version: ‘TSDF Low-res’ which allocates larger voxels, such that the parameters used to represent the geometry cost the same amount of memory storage as our approach; (2) A high resolution version: ‘TSDF High-res’ with much smaller voxel size (5mm in our experiments) for a higher quality 3D surface reconstruction. Fig. 6 shows the visual effect comparison in ICL-NUIM dataset using `lr_kt0` sequence among different tracking approaches. Our approach can achieve more complete 3D surface reconstruction results than both the ‘TSDF High-res’ and surfel tracking but only uses about $10\times$ less memory storage (Fig. 6 bottom left). If the same amount of memory is used as ours, both the camera tracking and the reconstruction quality would be severely hampered for TSDF approaches as shown in ‘TSDF Low-res’ results.

To make a comprehensive comparison, we evaluate the qualitative results between different approaches on ScanNet dataset [10]. The ScanNet dataset contains large scale real-

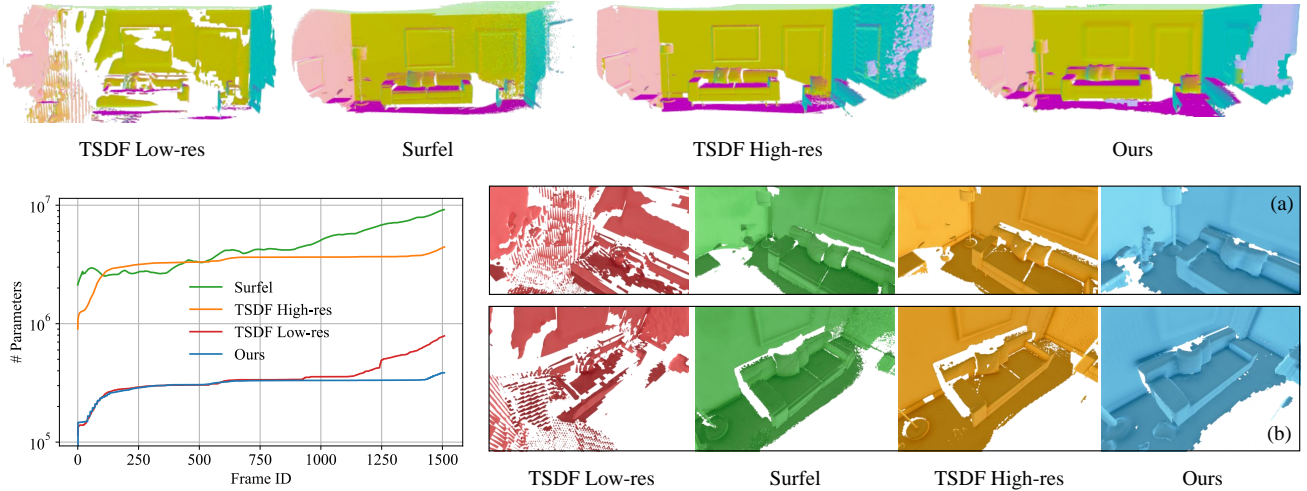


Figure 6. Qualitative comparisons and memory analysis on the `lrkt0` sequence of ICL-NUIM [18] dataset. On the top row we show a global view of the reconstructed 3D scene, where the colors represent the normal directions. Close-up looks at the details are visualized at the bottom (right), along with the number of parameters curves used in each approach in respect with the frames.

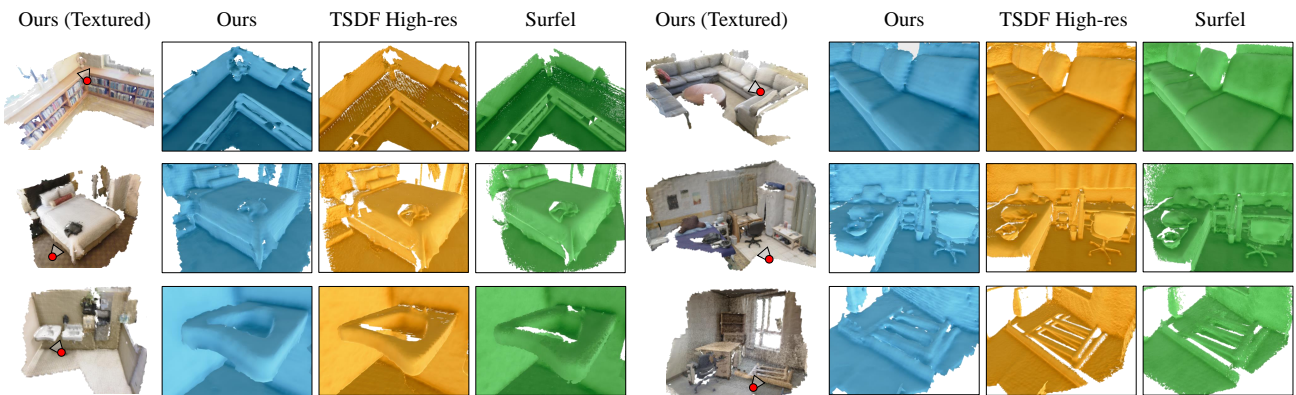


Figure 7. Qualitative comparisons on ScanNet [10] dataset. The 1st column of every scene shows a global view of our reconstruction with textures applied while detailed close-up comparisons with other baselines are shown in other columns. The close-up views’ camera positions are plotted on the textured reconstruction.

world 3D indoor scenes captured with a commodity RGB-D camera. As shown in Fig. 7, our approach can also achieve more complete 3D reconstruction results than the other two approaches. Note that we only use the pre-trained weight trained on ShapeNet dataset for the encoder-decoder network without any fine tuning on ScanNet dataset, which shows that our approach has good generalization performance across different scene categories.

The reason that our approach can produce more complete 3D surface reconstruction is mainly due to benefits from the learnt deep priors. Different from the other two representations (TSDF or surfel) without any knowledge about 3D scenes, our PLIVox encodes scene priors effectively such that the 3D surface can be recovered at the

position even without enough observations. Fig. 8 shows more visual results of our approach on ScanNet dataset. For video demonstrations, please visit <https://youtu.be/yxkIQFXQ6rw>.

4.4. System Ablations and Analysis

Probabilistic Modeling. To evaluate the effect of using probabilistic SDF for tracking, we ignore the decoder branch outputting σ_D and perform camera tracking by assuming the standard deviation as a constant 1.0 and evaluate the system on the ICL-NUIM dataset (denoted as ‘Ours w/o Prob’). As demonstrated by the consistent worse results than Ours for both camera trajectory estimation (Tab. 1) and surface error (Tab. 2), the way we encode the scene pri-



Figure 8. Other qualitative textured results on ScanNet [10] reconstructed online with our method.

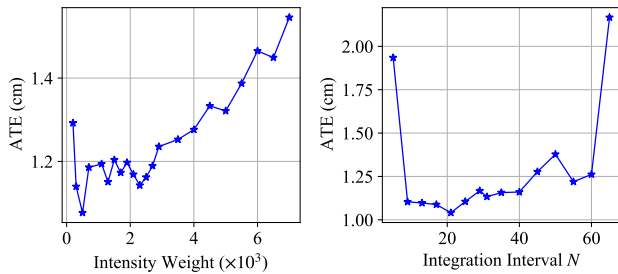


Figure 9. Influence of different parameters on ATE benchmark of ICL-NUIM [18].

ors with a *probabilistic* signed distance function takes effect compared to the previous representations [28, 33].

Intensity Term. The intensity term used in our camera tracking (Sec. 3.2) also influences the camera pose estimation accuracy. Here we modify our camera tracking with different weights of the intensity term in the objective function (Eq. 4), and evaluate it on the ICL-NUIM dataset. The ATE curve with respect to different intensity weights is demonstrated in Fig. 9 (left), showing the complementary effect between the two terms we used. Pure SDF tracking (*e.g.* $w = 0$) tends to fail in places with few geometric details and too large intensity weights (*e.g.* $w > 4000$) will ignore the reconstructed scene representation, causing increased drift. In practice, we set the weight of the intensity term as $w = 500$ in our experiments, which roughly balances the absolute scale of the SDF term E_{sdf} and intensity term E_{int} for accurate camera pose estimation.

Latent Vector Update. One alternative way for the geometry integration is to update the latent vector using the max operator as in [36] instead of what we propose in Eq (8), *i.e.* $l_m \leftarrow \max(l_m, l_m^t)$ and we call this baseline ‘Ours (max)’. Tab. 1 and Tab. 2 shows the average ATE and surface error of this approach, which are consistently worse than our proposed approach of taking the mean operator. One possible reason would be that the max operator is sensitive to sensor noise, thus leading to spurious reconstructions, while

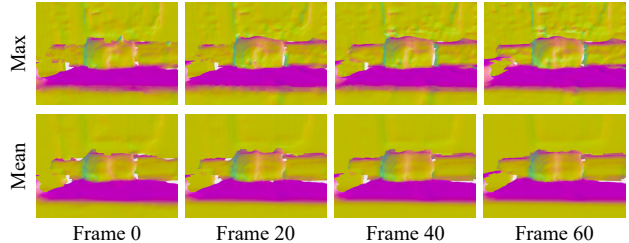


Figure 10. Comparison with alternative integration method using the max operator. Each column shows the actual frames we perform the integration.

Table 3. Timing of our system.

	Pre-processing	Tracking	Integration [†]	Meshing [†]
Time	20.3ms	62.8ms	34.4ms	30.0ms

[†]: Asynchronous.

our averaging update could provide more consistent reconstructions against sensor noise. Fig. 10 shows more visual results comparing the two integration methods.

Geometry Integration. The way we perform the geometry integration at sparse frame sampled from every N frames would also influence the surface mapping quality. Fig. 9 (right) shows the average ATE curve for the camera pose estimation along with different frame integration intervals. The average ATE becomes larger along with the increasing of integration interval.

Timing. Our system can run online at $\sim 12\text{Hz}$ on a platform with a GeForce GTX 1080 GPU and 32GB memory. The timing of each individual component of our system is reported in Tab. 3.

5. Conclusions

Limitations. Our approach has two main limitations: (1) we assume that each PLIVox is independent and do not consider the relationships between neighboring PLIVoxs. Therefore the learnt deep priors would not be spatially consistent, which would influence the accuracy of camera pose estimation. (2) No loop closure component has been incorporated to enforce global consistency. We hope to investigate deeply into both the limitations in the future.

In this paper we present DI-Fusion, which performs the online implicit 3D reconstruction with deep priors. With a novel PLIVox representation, our approach effectively incorporates scene priors (geometry and uncertainty priors) into both the camera tracking and surface mapping component, achieving more accurate camera pose estimation and higher quality 3D reconstruction. We hope that our approach can inspire more effective understanding of 3D scenes for advanced online 3D reconstruction.

References

- [1] Timothy D Barfoot. *State estimation for robotics*. Cambridge University Press, 2017. 4
- [2] Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew J. Davison. Codeslam - learning a compact, optimisable representation for dense visual SLAM. In *IEEE CVPR*, pages 2560–2568, 2018. 2
- [3] Yan-Pei Cao, Leif Kobbelt, and Shi-Min Hu. Real-time high-accuracy three-dimensional reconstruction with consumer RGB-D cameras. *ACM Trans. Graph.*, 37(5):171:1–171:16, 2018. 1, 4
- [4] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard A. Newcombe. Deep local shapes: Learning local SDF priors for detailed 3d reconstruction. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *ECCV*, volume 12374, pages 608–625, 2020. 1, 2, 5
- [5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5
- [6] Jiawen Chen, Dennis Bautembach, and Shahram Izadi. Scalable real-time volumetric surface reconstruction. *ACM Trans. Graph.*, 32(4):113:1–113:16, 2013. 1, 2
- [7] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. Bae-net: Branched autoencoder for shape co-segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8490–8499, 2019. 2
- [8] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *IEEE CVPR*, pages 5939–5948, 2019. 1
- [9] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *ACM SIGGRAPH*, pages 303–312, 1996. 1, 2, 5
- [10] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *IEEE CVPR*, 2017. 2, 6, 7, 8
- [11] 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 Trans. Graph.*, 36(3):24:1–24:18, 2017. 1, 2, 4
- [12] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *IEEE CVPR*, pages 4578–4587, 2018. 1
- [13] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE TPAMI*, 29(6):1052–1067, 2007. 2
- [14] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE TPAMI*, 40(3):611–625, 2018. 2
- [15] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: large-scale direct monocular SLAM. In *ECCV*, pages 834–849, 2014. 2
- [16] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Jimenez Rezende, and S. M. Ali Eslami. Conditional neural processes. In Jennifer G. Dy and Andreas Krause, editors, *ICML*, volume 80, pages 1690–1699, 2018. 3
- [17] M. Garon and J. Lalonde. Deep 6-dof tracking. *IEEE TVCG*, 23(11):2410–2418, 2017. 2
- [18] A. Handa, T. Whelan, J.B. McDonald, and A.J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE ICRA*, Hong Kong, China, May 2014. 2, 6, 7, 8
- [19] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *IEEE CVPR*, pages 6001–6010, 2020. 1, 2, 5
- [20] 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 TVCG.*, 21(11):1241–1250, 2015. 1, 2
- [21] 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 *2013 International Conference on 3D Vision-3DV 2013*, pages 1–8. IEEE, 2013. 2, 6
- [22] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Dense visual SLAM for RGB-D cameras. In *IEEE IROS*, pages 2100–2106, 2013. 6
- [23] Boram Lee, Clark Zhang, Zonghao Huang, and Daniel D Lee. Online continuous mapping using gaussian process implicit surfaces. In *IEEE ICRA*, pages 6884–6890, 2019. 2
- [24] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2019–2028, 2020. 2
- [25] 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
- [26] Wolfram Martens, Yannick Poffet, Pablo Ramon Soria, Robert Fitch, and Salah Sukkarieh. Geometric priors for gaussian process implicit surfaces. *IEEE Robotics Autom. Lett.*, 2(2):373–380, 2017. 2
- [27] John McCormac, Ronald Clark, Michael Bloesch, Andrew J. Davison, and Stefan Leutenegger. Fusion++: Volumetric object-level SLAM. In *3DV*, pages 32–41, 2018. 2
- [28] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE CVPR*, pages 4460–4470, 2019. 1, 2, 8
- [29] 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. *arXiv preprint arXiv:2003.08934*, 2020. 2
- [30] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics (TRO)*, 33(5):1255–1262, 2017. 2

- [31] 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 *IEEE ISMAR*, pages 127–136, 2011. [1](#), [2](#), [4](#)
- [32] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 32(6):169:1–169:11, 2013. [1](#), [2](#), [5](#), [6](#)
- [33] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE CVPR*, pages 165–174, 2019. [1](#), [2](#), [3](#), [5](#), [8](#)
- [34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. [6](#)
- [35] Victor Adrian Prisacariu, Olaf Kähler, Stuart Golodetz, Michael Sapienza, Tommaso Cavallari, Philip HS Torr, and David W Murray. Infinitam v3: A framework for large-scale 3d reconstruction with loop closure. *arXiv preprint arXiv:1708.00783*, 2017. [6](#)
- [36] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE CVPR*, pages 652–660, 2017. [8](#)
- [37] Vincent Sitzmann, Eric R. Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. In *NeurIPS*, 2020. [2](#)
- [38] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020. [2](#)
- [39] Edgar Sucar, Kentaro Wada, and Andrew Davison. NodeSLAM: Neural object descriptors for multi-view shape reconstruction. In *3DV*, 2020. [2](#)
- [40] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. CNN-SLAM: real-time dense monocular SLAM with learned depth prediction. In *IEEE CVPR*, pages 6565–6574, 2017. [2](#)
- [41] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice F. Fallon, John J. Leonard, and John McDonald. Real-time large-scale dense RGB-D SLAM with volumetric fusion. *I. J. Robotics Res.*, 34(4-5):598–626, 2015. [2](#)
- [42] Thomas Whelan, Stefan Leutenegger, Renato F. Salas-Moreno, Ben Glocker, and Andrew J. Davison. Elasticfusion: Dense SLAM without A pose graph. In *Robotics: Science and Systems*, 2015. [1](#), [2](#), [6](#)
- [43] Ming Zeng, Fukai Zhao, Jiaxiang Zheng, and Xinguo Liu. Octree-based fusion for realtime 3d reconstruction. *Graph. Model.*, 75(3):126–136, 2013. [2](#)