Robotics: Science and Systems 2022
New York City, NY, USA, June 27-July 1, 2022

iSDF: Real-Time Neural Signed Distance Fields for Robot Perception

Joseph Ortiz[1,2] Alexander Clegg[2] Jing Dong[3] Edgar Sucar[1]
David Novotny[2] Michael Zollhoefer[3] Mustafa Mukadam[2]

[1]Imperial College London [2]Meta AI [3]Reality Labs Research

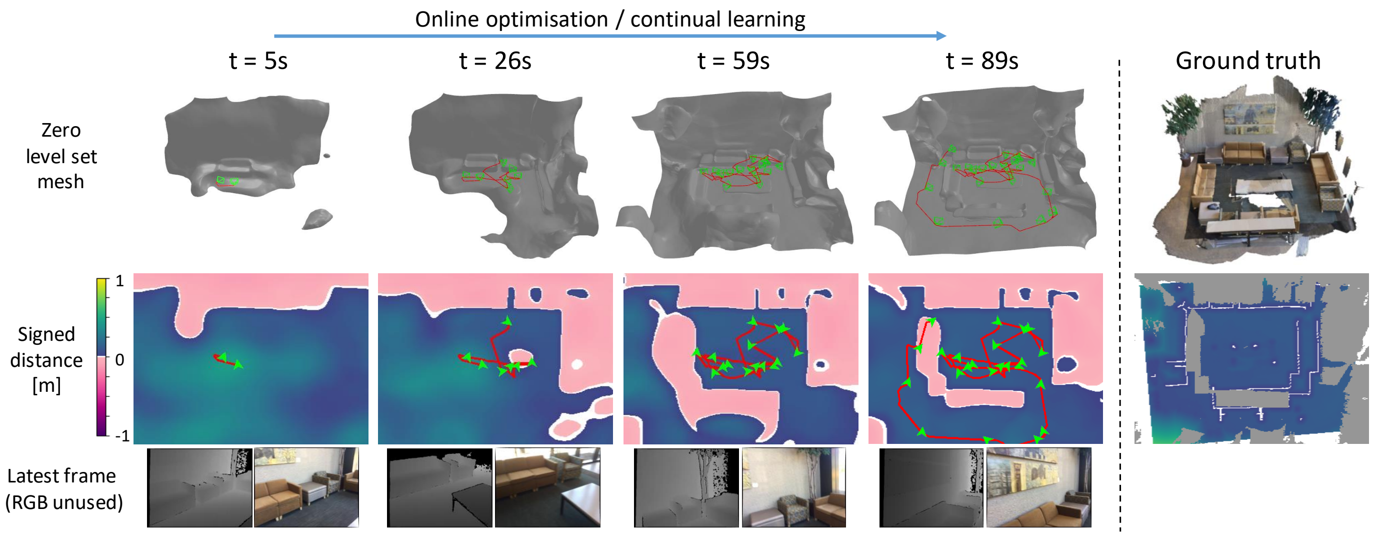

Fig. 1: We present iSDF, a system for real-time signed distance field reconstruction that optimises a randomly initialised network to regress to the signed distance for input 3D coordinates based on posed depth images from a live camera stream. To train the network in real-time we develop a batch-based self-supervision method. This network can be queried online to obtain collision costs and gradients for use by downstream planners in domains from navigation to manipulation. The trajectory (red) of the camera (green) is visualised on top of the zero level set (top row) and SDF slice (middle row) generated by iSDF over time (left to right).

***Abstract*—We present iSDF, a continual learning system for real-time signed distance field (SDF) reconstruction. Given a stream of posed depth images from a moving camera, it trains a randomly initialised neural network to map input 3D coordinate to approximate signed distance. The model is self-supervised by minimising a loss that bounds the predicted signed distance using the distance to the closest sampled point in a batch of query points that are actively sampled. In contrast to prior work based on voxel grids, our neural method is able to provide adaptive levels of detail with plausible filling in of partially observed regions and denoising of observations, all while having a more compact representation. In evaluations against alternative methods on real and synthetic datasets of indoor environments, we find that iSDF produces more accurate reconstructions, and better approximations of collision costs and gradients useful for downstream planners in domains from navigation to manipulation. Code and video results can be found at our project page: https://joeaortiz.github.io/iSDF/.**

I. Introduction

Robot perception is often the forgotten middle child. One instantiation of this syndrome occurs with signed distance fields (SDFs), a map representation common in both robotics and vision. SDFs are scalar fields that associate each point in space with the signed distance to the closest surface point, therefore encoding the surface as the zero level set. In robotics, SDFs are a common environment representation used for collision avoidance in motion planning [40], however they are usually assumed as given a priori or too expensive to compute in real-time. On the other hand, in vision, truncated signed distance fields (TSDF) are common in SLAM systems [23, 24], but there is little work on reconstructing non-truncated SDFs in room scale environments. In order to close this gap, in this work, we focus on the problem of real-time SDF reconstruction.

Within robot motion planning, signed distance fields are the prevailing map representation used in trajectory optimisation [40, 32, 28, 10, 22] where a trajectory is found by minimising an objective balancing smoothness and collision avoidance costs. Collision costs are dependent on the environment and robot state and in practise SDFs are a common intermediate representation used to mitigate challenges in building generic cost fields for high dimensional robots. Surface representations, such as occupancy maps or TSDFs, are common in SLAM systems but impractical for planning since trajectory optimisation requires many cost queries. It is more efficient to precompute a cheap-to-evaluate cost field, like a SDF, than to compute costs on the fly from a surface reconstruction. Although truncated SDFs can be rapidly constructed by fusing depth measurements, producing the full SDF is far more challenging as fusion alone leads to large errors away from surfaces.

Prior work on real-time reconstruction of non-truncated SDFs operates in two stages, first reconstructing surface geometry and then transforming it to the SDF using wavefront propagation algorithms [25, 15]. These methods are based on voxel grids and are limited to a resolution of 5cm due to the cost of wavefront propagation. This is the case for both efficient CPU-based propagation algorithms [25, 15] as well as for GPU-based brute force propagation algorithms [12].

Neural fields provide an alternative paradigm to voxel grids for modelling scenes [26, 20, 21]. Based on a multi-layer perceptron (MLP) that maps a 3D coordinate to occupancy, these models can be optimised from scratch to accurately fit a specific scene without prior training. Recent work has shown that neural fields can reconstruct highly accurate 3D geometry and that they can be trained in real-time as part of a SLAM system [30, 39].

Building on these advances, we present iSDF (*incremental Signed Distance Fields*), a system for real-time SDF reconstruction that is based on a neural network that regresses input 3D coordinates to signed distance. Unlike iMAP [30], which reconstructs a neural radiance field as part of a SLAM system, we focus on mapping with a neural signed distance field and develop novel self-supervision and sampling strategies.

More specifically, given a stream of posed depth images, a randomly initialised network is trained in a continual learning fashion via actively replaying past observations. Our approach is simple, yet effective, and runs in real-time (33ms per step); we employ a single small network, sparse sampling, and do not require pretraining. During online operation, the model is queried at points sampled along back-projected rays in the frames and the network weights are optimised to minimise a loss on both the predicted signed distance and its spatial gradient. Our loss bounds the SDF prediction with the distance to the closest surface point in the batch and enables the learning of signed distances far from surfaces without the need for wavefront propagation.

iSDF inherits the positive characteristics of neural fields. In contrast to prior work based on voxel grids, iSDF is: (i) efficient - it can seamlessly allocate memory capacity to model different parts of an environment with different levels of detail, and (ii) predictive - it can denoise and consolidate noisy measurements and sensibly fill in gaps in partially observed regions; all while having a more compact representation.

We show that iSDF outperforms prior work in terms of SDF accuracy on both synthetic and real datasets in indoor environments. iSDF achieves an SDF error of less than 6cm for all our sequences and is the only method that can reconstruct watertight meshes of the zero level set. We also evaluate on computing collision costs and gradients from signed distance and find that iSDF outperforms alternatives on these metrics, demonstrating its utility for downstream planners in domains from navigation to manipulation.

II. Related Work

Since the seminal works [26, 20, 21], neural fields have become a prevailing representation for modelling both 3D objects and full scenes. In particular DeepSDF [26] inspired a large number of works on reconstructing object surfaces as the zero level set of neural signed distance fields. Along this direction, Gropp et al. [14] introduce Eikonal regularisation to supervise the SDF learning for points in free space while Zhang et al. [38] use the surface normals to supervise free space points. A number of works address differentiable rendering to learn SDFs from images [17, 37, 34, 19] while Atzmon et al. [2, 3] focus on learning SDFs from raw pointcloud data.

Truncated SDFs have commonly been used in traditional SLAM systems [23, 24], however more recently a number of works have used neural fields to represent the TSDFs of room-scale environments [4, 36, 29]. To the best of our knowledge there is no prior work tackling non-truncated SDF reconstruction in room scale environments.

Neural fields showed great early promise with offline reconstruction tasks. Later, iMAP [30] was the first work to show that these models can be trained in a real-time continual learning setting as part of a SLAM system. Our work is strongly inspired by iMAP and indeed the active sampling and keyframe selection in iSDF are based on iMAP. NICE-SLAM [39] builds on iMAP by proposing to use a voxel grid of neural fields instead of a single global model. Although not a real-time system, Yan et al. [36] investigates offline continual learning for neural fields.

The most closely related prior works that tackle real-time SDF reconstruction from posed depth images are Voxblox [25] and FIESTA [15]. These methods operate in two phases and output a voxel grid of SDF values. First, depth images are fused into a surface representation (TSDF or occupancy grid) before the full SDF is computed via efficient wavefront propagation algorithms. The key to the efficiency is to only emanate wavefronts from updated voxels with the propagation being done by a breadth first search (BFS). We find that Voxblox and FIESTA perform comparably and in evaluations show that iSDF produces more accurate and complete SDFs than Voxblox.

Other notable works tackle SDF reconstruction in dynamic environments with composite fields [13] and in very large environments using submaps [27]. Lastly, other related directions tackle learning cost fields for motion planning [6, 9, 33] from SDFs [11, 16], navigating in a neural radiance fields [1] and learning neural fields for articulated [18] or deformable objects [35] for manipulation.

III. Signed distance fields

A signed distance field is a scalar field which associates every point in space with the signed distance to the closest surface. The distance is assigned a negative sign for points inside the surface and takes a positive sign outside of the surface. When representing three dimensional geometry, the signed distance function maps a 3D coordinate $\mathbf{x}$ to the scalar signed distance value $f : \mathbb{R}^3 \to \mathbb{R}$. The surface $\mathcal{S}$ is the zero level-set of the field:

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 \mid f(\mathbf{x}) = 0\} \ . \tag{1}$$

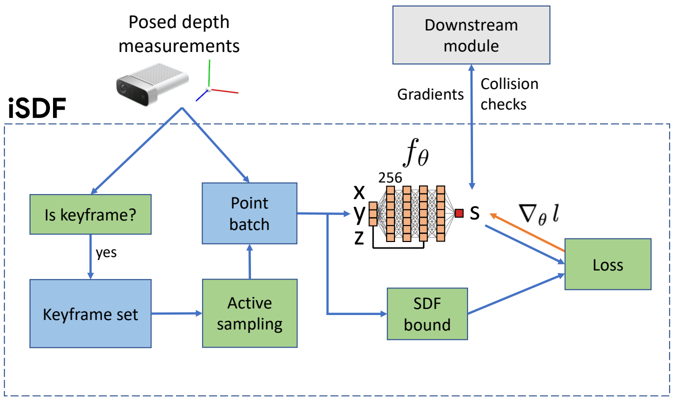

Fig. 2: **System diagram.** iSDF's MLP regresses input 3D coordinates $\mathbf{x} = (x, y, z)$ to the signed distance $s = f_\theta(\mathbf{x})$.

A signed distance field has several notable properties that we will use to construct our loss function (Sec. IV-D): (i) f is differentiable almost everywhere. It is not differentiable at points where there are multiple equidistant closest surface points, (ii) where the gradient exists, $-\nabla_\mathbf{x} f$ points towards the closest surface and on the surface the gradient is equal to the surface normal: $\nabla_\mathbf{x} f = \hat{\mathbf{n}}$, and (iii) the gradient vector satisfies the Eikonal equation: $|\nabla_\mathbf{x} f| = 1$, as moving a distance δx along the gradient vector increases the distance to the closest surface by δx.

IV. iSDF - Real-time SDF reconstruction

We present iSDF a system for real-time SDF reconstruction, that takes as input a stream of posed depth images captured by a moving camera and, during online operation, learns a function that approximates the true signed distance field of the environment. An overview of our system is given in Fig. 2. We focus on mapping rather than the full SLAM problem and therefore assume there is an external tracking module. The signed distance function is modelled by a multilayer perceptron (MLP), that maps a 3D coordinate $\mathbf{x}$ to the signed distance value s at that point: $f(\mathbf{x};\theta) = s$ (Sec. IV-A). The model is initialised with random weights and is optimised in real-time with respect to incoming measurements.

At each iteration, we select a subset of frames, i.e. a few of the posed depth images. Given the continual learning setting, the frames are selected via active sampling from a sparse set of representative keyframes to mitigate catastrophic forgetting. For each frame, we sample both random pixels and depths along the corresponding back-projected rays to obtain a batch of points (Sec. IV-B). The SDF model is queried at the sample points and we compute a loss that is minimised by refining the network weights with back-propagation. The loss supervises both the predicted signed distance and its spatial gradient (Sec. IV-D) and is formed by bounding the predicted signed distance with the distance to the closest surface point in the batch (Sec. IV-C).

A. Network architecture

We model the signed distance function using an MLP with 4 hidden layers, 256 activations each, and softplus activations in all intermediate layers. As in NeRF [21], we apply positional embedding to the 3D coordinates before feeding them to the network. This embedding transforms the input coordinates to a high dimensional vector using periodic activation functions and is crucial for reconstructing high frequency signals. We employ the "off-axis" positional embedding $\gamma(\mathbf{x}) = [\sin(2^0\mathtt{A}\mathbf{x}), \cos(2^0\mathtt{A}\mathbf{x}), \dots, \sin(2^L\mathtt{A}\mathbf{x}), \cos(2^L\mathtt{A}\mathbf{x})]$ with $L = 5$ from Barron et al. [5], where the rows of $\mathtt{A} \in \mathbb{R}^{21\times 3}$ are the outward facing unit-norm vertices of a twice-tessellated Icosahedron. The positional embedding is also concatenated to the third layer of the network.

B. Active sampling

The sampling procedure, broadly follows iMAP [30] on the level of frames and NeRF [21] on the level of points. Optimising the network with respect to all incoming frames would be both computationally infeasible and redundant. Therefore, as in iMAP [30], we maintain a sparse set of keyframes which are replayed via active sampling to avoid catastrophic forgetting. The first frame is always added to the keyframe set and subsequent frames are added based on the information gain metric from iMAP [30] that is computed using a frozen copy of the network when the last keyframe was added. At each iteration, we select 5 frames - the 2 latest frames and 3 keyframes from the keyframe set sampled from a distribution that is formed by normalising the running total losses of each keyframe. New frames are sampled for 10 iterations before checking for a more recent frame.

Given each selected frame, which comprises of a camera pose T_{WC} and a depth image D, we sample query points within the viewing frustum. First, we randomly sample 200 pixel coordinates $[u, v]$ in the depth image. Then, the viewing direction is computed using the camera intrinsic matrix K and transformed into world coordinates: $\mathbf{r} = T_{WC}K^{-1}[u, v]$.

For each ray, we sample $N + M + 1$ depths values d_i along the ray. The depth values are made up of N stratified samples in the range $[d_{min}, D[u, v] + \delta]$ and M samples from the Gaussian distribution $\mathcal{N}(D[u, v], \sigma_s^2)$ to provide more supervision around the surface where the SDF is harder to model. Additionally, we always include the sample at the surface given by the depth map by including $d_i = D[u, v]$. These depth samples, along with the viewing directions, specify the sample points along the rays: $\mathbf{x}_i = d_i\mathbf{r}$. The batch of points is the set of sample points from all selected frames.

In all experiments, we set $N = 20$, $M = 8$, $d_{min} = 7cm$, $\delta = 10cm$ and $\sigma_s = 10cm$.

C. Approximating the closest surface point

At each iteration, given the batch of points, we query the network to give the predicted signed distances: $\hat{s} = f(\mathbf{x};\theta)$. Ideally, we would like to supervise these predictions using the true signed distance values:

$$s(\mathbf{x}) = \text{sgn}(D[u, v] - d) \inf_{\mathbf{y}\in\mathcal{S}} |\mathbf{x} - \mathbf{y}| \ , \tag{2}$$

where $\inf$ denotes the infimum, sgn the sign function and $\mathcal{S}$ is the set of all true surface points. It would however be computationally infeasible to compute the distance to all surface points and would only yield an approximate signed distance due to the geometry being partially observed.

Instead of computing the distance to all surface points, we propose to instead use the distance to a single carefully chosen nearby surface point $\mathbf{p}$. The nearby surface point is unlikely to be the true closest surface point so will yield a signed distance value with larger absolute value than the true signed distance. In this way, computing the distance to a nearby surface point provides a bound $b(\mathbf{x}, \mathbf{p})$ on the signed distance:

$$b(\mathbf{x}, \mathbf{p}) = \text{sgn}(D[u, v] - d)\, |\mathbf{x} - \mathbf{p}| \ , \tag{3}$$

where $|s(\mathbf{x})| \leq |b(\mathbf{x}, \mathbf{p})|$, and with equality when $\mathbf{p}$ is the closest surface point to $\mathbf{x}$.

The closer the nearby surface point is to the query point, the tighter the bound $b(\mathbf{x}, \mathbf{p})$ is on the true signed distance. We explore a number of methods for choosing the nearby surface point $\mathbf{p}$ that trade of increasingly tighter bounds for additional computation.

The cheapest option we explore is to use the ray intersection with the surface: $\mathbf{p} = D[u, v]\, \mathbf{r}$. In this case, the bound is simply the signed distance along the ray between the query point depth and the measured depth: $b(\mathbf{x}, D[u, v]\, \mathbf{r}) = \text{sgn}(D[u, v] - d)|D[u, v] - d|$. If the surface is perpendicular to the ray / viewing direction $\mathbf{r}$, then this bound is more likely to be tight while if these vectors are not perpendicular we can always find a new surface point that is closer to the query point by moving a small distance along the surface.

To approximately account for this, we can apply a correction to the bound using the angle between the surface normal and the viewing direction. As illustrated in Fig. 3, if we assume the surface is planar around the ray surface intersection, then the normal adjusted bound is: $b(\mathbf{x}, \hat{\mathbf{r}}, \hat{\mathbf{n}}) = -\hat{\mathbf{r}} \cdot \hat{\mathbf{n}}\ (d - D[u, v])$, where $\hat{\mathbf{r}} = \frac{\mathbf{r}}{|\mathbf{r}|}$. As we assume that the surface is only locally planar, we apply the normal correction to points less than 30cm from the ray surface intersection, and use the ray bound elsewhere. To apply this correction, we must first compute the surface normal as the spatial gradient of the depth image.

The last method we consider is brute-force computation of the distance to the closest surface point in the batch. This gives the following bound:

$$b(\mathbf{x}, \mathcal{P}) = \text{sgn}(D[u, v] - d) \min_{\mathbf{p} \in \mathcal{P}} |\mathbf{x} - \mathbf{p}| \ , \tag{4}$$

where $\mathcal{P}$ is the set of surface points in the batch that are sampled at the measured depth along the ray (i.e. $\mathbf{p} = D[u, v]\mathbf{r}$).

For all methods, we can also compute an approximate SDF gradient at $\mathbf{x}$ using the fact that $-\nabla_{\mathbf{x}} f(\mathbf{x})$ points towards the closest surface. For example, using the closest batch surface point, the approximate unnormalised gradient is:

$$\mathbf{g}(\mathbf{x}, \mathcal{P}) = \text{sgn}(D[u, v] - d) \,.\, \left(\mathbf{x} - \arg\min_{\mathbf{p} \in \mathcal{P}} |\mathbf{x} - \mathbf{p}|\right) \ . \tag{5}$$

The three methods we consider above for computing the bound are illustrated on the left of Fig. 3. On the right, we plot the bound for each method at query points along different rays. The ray distance clearly provides very weak bounds and normal correction leads to minimal improvements. On the other hand, computing the batch distance can provide tight bounds depending on whether the batch has good surface coverage. In practice, we find that the batch distance method yields the best performance at little extra computation cost (see Sec. VI-D) and thus employ this method in all experiments.

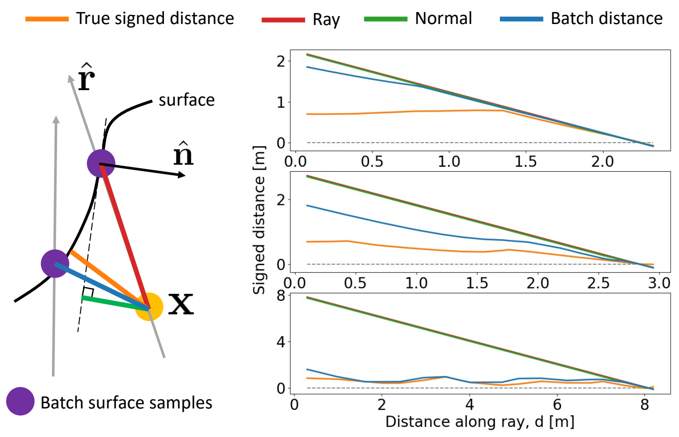

Fig. 3: *Left:* Different methods for computing the bound on the SDF prediction for sample point $\mathbf{x}$ (orange). *Right:* The bound for each method for sample points along a ray. Each plot shows a different ray and the distance along the ray is measured from the camera centre. The batch distance provides the tightest bounds on the true signed distance.

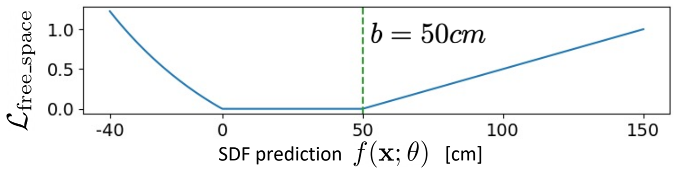

Fig. 4: **Free space loss.** $\mathcal{L}_{\text{free\_space}}(f(\mathbf{x};\theta), b)$ for target SDF.

D. *Loss for self-supervised continual learning*

SDF loss. Using the bound on the predicted SDF value, we construct a loss $\mathcal{L}_{\text{free\_space}}$ in Fig. 4 for points in free space. The loss is zero if the prediction is positive and less than b, linear in the prediction when it exceeds the upper bound b, and exponential for negative predictions (we set $\beta = 5$):

$$\mathcal{L}_{\text{free\_space}}(f(\mathbf{x};\theta), b) = \max\left(0,\ e^{-\beta f(\mathbf{x}_i;\theta)} - 1,\ f(\mathbf{x}_i;\theta) - b\right) . \tag{6}$$

Close to the surface, we expect the bound to be very tight and so instead of applying the free space loss, we directly supervise the SDF prediction to take the bound value via an $L1$ loss:

$$\mathcal{L}_{\text{near\_surf}}(f(\mathbf{x};\theta), b) = |f(\mathbf{x}_i;\theta) - b| \ . \tag{7}$$

Applying the near surface loss within a truncation region t and the free space loss elsewhere, our full SDF loss is:

$$\mathcal{L}_{\text{sdf}}(f(\mathbf{x};\theta), b) = \begin{cases} \lambda_{\text{surf}} \mathcal{L}_{\text{near\_surf}} & \text{if } |D[u, v] - d| \leq t \\ \mathcal{L}_{\text{free\_space}} & \text{otherwise.} \end{cases} \tag{8}$$

Gradient loss. The gradient of the SDF prediction can be efficiently computed via automatic differentiation. We therefore also supervise the SDF gradient prediction using our approximated gradient $\mathbf{g}$, computed using the closest surface point in the batch. We introduce a loss penalising the cosine distance between the two vectors:

$$\mathcal{L}_{\text{grad}}(\nabla_{\mathbf{x}} f(\mathbf{x};\theta), \mathbf{g}) = 1 - \frac{\nabla_{\mathbf{x}} f(\mathbf{x};\theta) \cdot \mathbf{g}}{\|\nabla_{\mathbf{x}} f(\mathbf{x};\theta)\| \|\mathbf{g}\|} \ . \tag{9}$$

For surface point samples, we replace $\mathbf{g}$ with the surface normal computed from the depth image.

Eikonal regularisation. Without very dense supervision, the above losses alone do not result in the model learning a valid signed distance field which satisfies the Eikonal equation almost everywhere. Consequently, following Gropp et al. [14] we apply Eikonal regularisation via the loss:

$$\mathcal{L}_{\text{eik}}(f(\mathbf{x};\theta)) = \begin{cases} | \, \|\nabla_{\mathbf{x}} f(\mathbf{x};\theta)\| - 1 | & \text{if } |D[u,v] - d| \geq a \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Different to prior work, we only regularise points a distance greater than $a = 10cm$ from the ray surface intersection. We find this improves performance, as gradient discontinuities that do not satisfy the Eikonal equation are far more common close to surfaces where nearby points often have different closest surfaces. The regularisation has the effect of propagating the field out from near the surfaces (where there is the strongest signal from the SDF loss) into free space. In this sense, it can be thought of as playing a similar role to the wavefront propagation algorithms.

The network parameters are optimised to minimise the loss:

$$l(\theta) = \mathcal{L}_{\text{sdf}} + \lambda_{\text{grad}} \mathcal{L}_{\text{grad}} + \lambda_{\text{eik}} \mathcal{L}_{\text{eik}} \ . \tag{11}$$

We set $\lambda_{\text{surf}} = 5$, $\lambda_{\text{grad}} = 0.02$ and $\lambda_{\text{eik}} = 0.25$ and the weights are optimised using the Adam optimiser with learning rate 0.0013 and weight decay 0.012.

V. Experimental Setup

A. Datasets

ReplicaCAD [31]. We experiment with 6 synthetic sequences from the ReplicaCAD dataset [31] generated by simulating the measurements recorded by a camera mounted on a mobile manipulator. For two different room configurations, we generate 3 sequences which each aim to evaluate different properties of the reconstruction. For each room we generate 1) a navigation sequence (*nav*) in which the robot explores the majority of the room by navigating between waypoints, 2) an object reconstruction sequence (*obj*) in which the robot approaches and looks directly at two large objects (e.g. a lamp or bag), and 3) a manipulation sequence (*mnp*) in which the robot navigates close to a small object (e.g. a bowl) with the intent to reach out and grasp it. We apply the noise model from Choi et al. [7] to all depth images to account for disparity-based quantisation effects, realistic high-frequency noise, and low-frequency distortions.

ScanNet [8]. We use 3 longer and 3 shorter randomly chosen sequences from the ScanNet dataset. These sequences were captured using a handheld RGB-D camera.

Ground truth SDF computation. For each sequence, we pre-compute a voxel grid of ground truth SDF values with a voxel resolution of 1cm. To evaluate the real-time SDF reconstruction, this grid is interpolated to produce values at query points. The SDF is computed by first voxelizing the mesh into an occupancy grid, then computing the euclidean distance transform of the occupancy and inverse occupancy grid, before subtracting the latter from the former to give the SDF. For the ReplicaCAD sequences, we compute separately the SDF for an empty room and for each object in the room. These are then composed using the min-operation to give the full room SDF. SDF computation is more challenging for the ScanNet sequences as the mesh is not watertight, making it difficult to determine whether a point is in free space or contained within a surface. Consequently, for the ScanNet sequences, we only evaluate in visible regions, where the SDF attains strictly positive values.

B. Comparisons

We compare iSDF against two voxel-based methods for real-time SDF reconstruction that employ a two stage pipeline of first reconstructing the surface via fusing depth images and then transforming to SDF. For both methods the second stage is the dominant computational cost.

Voxblox [25]. Voxblox is a CPU-only algorithm that first runs a fast TSDF fusion before efficiently computing the SDF by propagating wavefronts only from updated voxels. The voxel size is set to 5.5cm such that SDF updates take around half a second for room scale environments; any higher resolution would make Voxblox too slow for real-time planning. Voxblox employs complex data structures for efficient wavefront propagation and would therefore be challenging to parallelise on a GPU.

KinectFusion+. To the best of our knowledge there are no methods that leverage GPUs for real-time SDF reconstruction. Consequently, we implement our own GPU-based SDF reconstruction system in C++ with custom CUDA kernels. We call this method *KinectFusion+* as the first stage fuses depth images into an occupancy grid (much like KinectFusion [23]), before computing the SDF via the Euclidean distance transform (EDT) of the occupancy and inverse occupancy. We use the efficient EDT from Felzenszwalb and Huttenlocher [12] which allows for parallelisation by decomposing the computation along each axis. This method is again limited by real-time constraints to a voxel size of 7cm. More details on KinectFusion+ are provided in the supplementary document.

All experiments are run on an Intel i7-8700K CPU with a GeForce RTX 2080 GPU.

C. Evaluation metrics

We evaluate the reconstruction quality of the SDF using three different metrics. All metrics are evaluated at $200k$ points computed by randomly sampling both pixels from the frames and then depths along the back-projected rays. All methods are evaluated at the same points. Results for iSDF are averaged over 10 runs with different network initialisations; Voxblox and KinectFusion+ are deterministic.

From a mapping perspective, the first metric is the absolute error between the predicted and ground truth SDF:

$$\textbf{SDF error: } e_{\text{sdf}}(\mathbf{x}) = |\hat{s}(\mathbf{x}) - s(\mathbf{x})| \ . \tag{12}$$

To mimic the collision cost queries in a real-time planning scenario, the second metric is the error in the predicted collision cost. We use the cost function from CHOMP [40] (with $\epsilon = 2m$) which gives higher importance to regions close to and inside surfaces. The cost function is similar to the hinge loss but with a quadratic section close to the surface:

$$c(s) = \begin{cases} -s + \frac{1}{2}\epsilon & \text{if } s < 0 \\ \frac{1}{2\epsilon}(s - \epsilon)^2 & \text{if } 0 < s \leq \epsilon \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

With this, our second metric is:

$$\textbf{Collision cost error: } e_{\text{collision}}(\mathbf{x}) = |c(\hat{s}(\mathbf{x})) - c(s(\mathbf{x}))| \ . \tag{14}$$

The accuracy and smoothness of the gradient of the SDF is crucial for optimisation-based planners. Consequently, our last metric is the cosine distance between the predicted and ground truth gradient vectors:

$$\textbf{Gradient cosine distance: } e_{\text{grad}}(\mathbf{x}) = 1 - \frac{\nabla_{\mathbf{x}}\hat{s}(\mathbf{x}) \cdot \nabla_{\mathbf{x}}s(\mathbf{x})}{\|\nabla_{\mathbf{x}}\hat{s}(\mathbf{x})\|\|\nabla_{\mathbf{x}}s(\mathbf{x})\|} \ . \tag{15}$$

VI. Results

We qualitatively and quantitatively evaluate iSDF against Voxblox and KinectFusion+ and find that our method produces more accurate and complete SDFs. Additionally, due to the choice of a neural field representation, we demonstrate that iSDF can plausibly fill in partially observed regions and map at different levels of detail. We encourage the reader to view our supplementary video for visualisations of real-time SDF reconstruction.

Throughout the main paper, results are presented for 6 selected sequences (3 synthetic, 3 real) that are representative of all 12 (6 synthetic, 6 real). Results for the remaining 6 sequences, which provide further evidence for our claims, can be found in the supplementary document. We evaluate iSDF with the batch distance to the closest point as the bound for the loss unless otherwise stated.

A. Qualitative results

To visualise the reconstructed SDFs, we compare 2D slices of the SDFs at constant heights in Fig. 5. The first obvious difference is that, unlike Voxblox and KinectFusion+ which map only the visible region, iSDF is predictive and reconstructs a plausible field in the full environment. For example, in the interiors of the beanbag and table (highlighted by the orange boxes in Fig. 5) or the interior of a wall, iSDF is the only method that returns predictions.

In the main visible regions, all methods produce visually realistic looking fields that are comparable to the ground truth. There are significant differences however towards the edges of the visible region near surfaces, where iSDF is the only method able to produce a complete reconstruction. As Voxblox and KinectFusion+ employ fusion in the first stage, there are holes in the reconstructions in regions that no rays have reached, shown in white in the slices. This is particularly significant for Voxblox, which to achieve real-time performance on a CPU, discards rays that pass through already updated voxels. The holes, highlighted by the red boxes in Fig. 5, are therefore most common in distant regions that are partially occluded by nearby objects. These holes are an issue for downstream planners as unmapped regions must be marked as unnavigable and given a high cost. Additionally, as the gradient field is computed using finite differences, the holes in the gradient field are even larger.

Further evidence that iSDF produces more complete and plausible reconstructions can be seen by visualising the zero level set of the field in Fig. 6. Voxblox and KinectFusion+ produce incomplete and noisy level sets, with bumpy walls and large gaps in the floor. On the other hand, iSDF is able to sensibly fill in gaps and denoise noisy measurements to produce a smooth and water-tight mesh. This ability to denoise and interpolate remarkably comes only from priors in the network structure and features learned in a self-supervised manner by fitting the observed part of the room, and without any pretraining.

We further explore the ability of iSDF to plausibly interpolate in partially observed regions in Fig. 7. In the left of the figure, we consider a scenario common for mobile robots in which a camera is mounted at a fixed height and has limited visibility of the floor. In this case, iSDF is the only method that can make accurate predictions at floor level and above the camera. In Fig. 7 *middle*, we demonstrate that due to the global neural representation, in which features are shared for all predictions, iSDF can accurately fill in the unobserved backside of objects.

Unlike uniform voxel grids which set a single resolution for the whole scene, neural fields can adaptively allocate memory capacity to model different parts of the environment with different levels of detail. This property is very appealing in mobile manipulation settings that require coarse room level reconstructions to navigate towards a target and then fine-detailed reconstructions to manipulate the target. When reconstructing large environments from a distance, iSDF may miss surface detail that uniform voxel grids can capture, however when the camera approaches an object of interest, increased supervision causes iSDF to allocate more memory capacity to this region and produce fine-detailed object reconstructions. Fig. 7 *right* shows results for a sequence in which the robot approaches a salt shaker with the intent to manipulate it. iSDF is the only method capable of reconstructing the small salt shaker while the alternatives based on voxel grids produce coarse and noisy reconstructions.

B. Quantitative results

In Fig. 8 we present quantitative evaluations along our three evaluation metrics. iSDF consistently achieves the lowest SDF

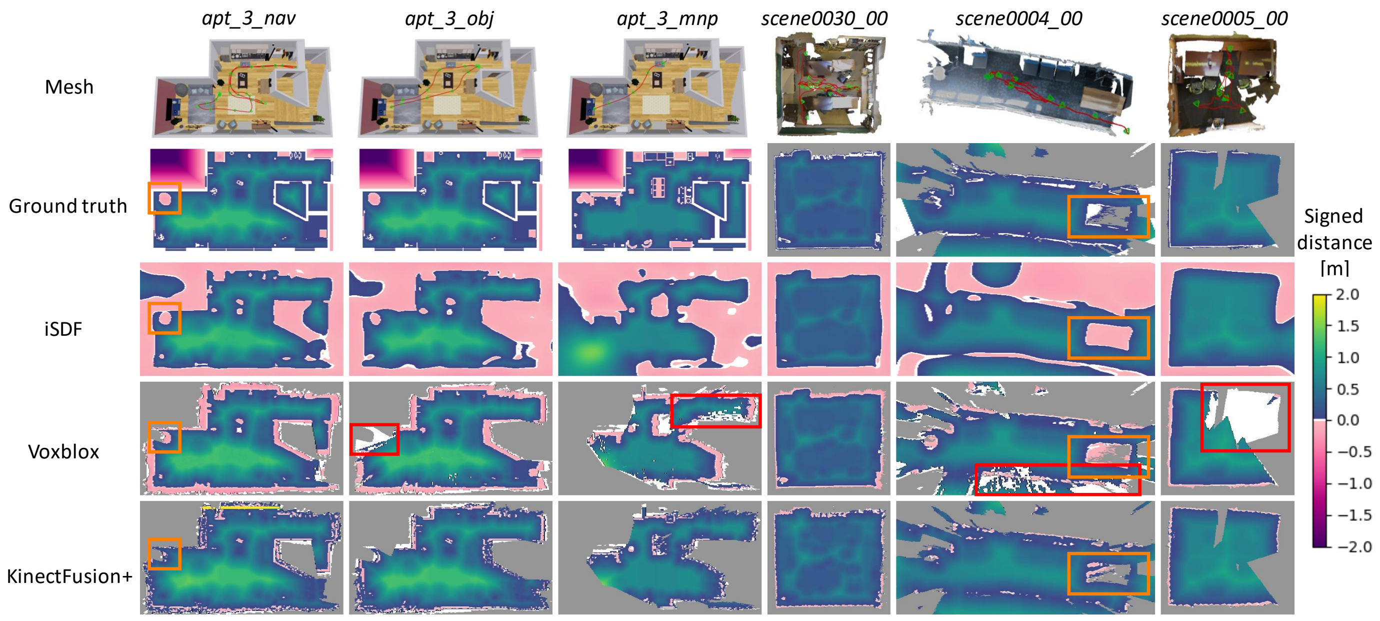

Fig. 5: **SDF slices**. Slices at constant height of the reconstructed SDF at the end of the sequence (we choose different heights for each sequence to capture the key scene elements). The meshes are shown for reference with the camera trajectory and keyframes selected by iSDF overlaid. For Voxblox and KinectFusion+, the slices are greyed out in the non-visible region as neither method makes predictions in this region. The ground truth ScanNet [8] slices are greyed out in the non-visible regions as we only have ground truth SDF values in visible regions. White regions in the Voxblox and KinectFusion+ slices are regions that are visible but unmapped (i.e. no rays reach this region).

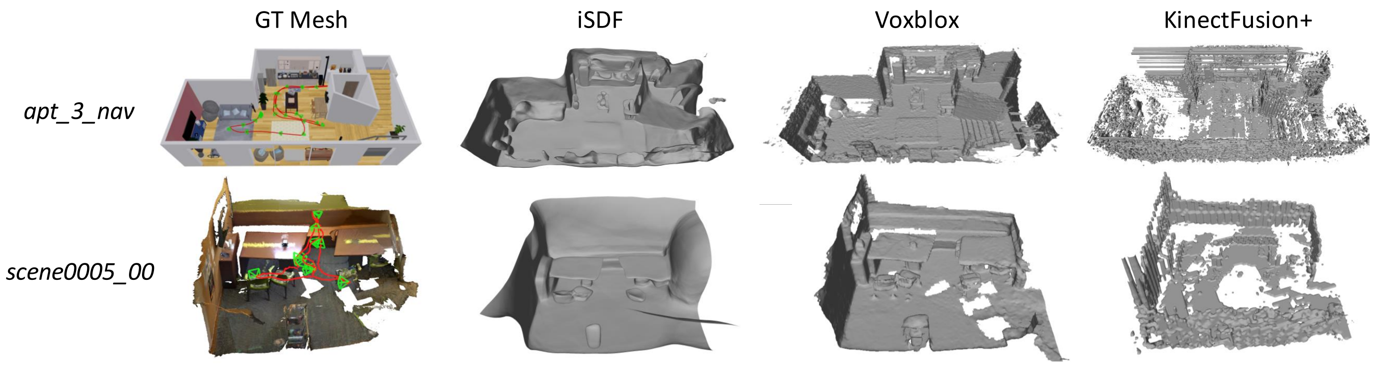

Fig. 6: The meshes are produced by querying the SDF predictions on a uniform grid and then running marching cubes to find the zero crossing of the level set. iSDF produces a complete and denoised mesh unlike Voxblox and KinectFusion+.

error, always reaching an error less than $6cm$ at the end of the sequence. The performance gap varies for different sequences, however iSDF often outperforms the other methods by more than $5cm$. The collision cost error follows a similar pattern, although notably, the performance of Voxblox is slightly worse along this metric as iSDF and KinectFusion+ are comparatively more accurate close to the surface. As iSDF produces the most accurate SDF, it is unsurprising that the gradient field is also in general the most accurate. The performance gap for gradients is however smaller with KinectFusion+ doing better on some sequences. To understand why iSDF is more accurate, in the supplementary, we split the SDF error into bins according to the true signed distance. We find that far away from surfaces iSDF is significantly more accurate than alternative methods, while very close to surfaces Voxblox can achieve similar accuracy to iSDF.

The difference in accuracy is primarily due to the large voxel size used by Voxblox and KinectFusion+ for real-time performance. Voxblox also introduces further errors by measuring distances only along the horizontal, vertical, and diagonal lines in the grid.

A notable result is that iSDF is often better along all metrics by the largest margin at the start of the sequence. This initial reconstruction is particularly crucial for navigation, as trajectories are more likely to be very poor early on when mapping is limited. As can be seen in the evolution of the reconstruction in Fig. 1, iSDF very rapidly captures the rough structure of the room after just a few seconds.

C. *Timings and memory*

In real robotics scenarios, memory and compute are often severely limited. Comparing memory footprints, iSDF requires only 1MB of memory for the network weights and around 20 keyframes to reconstruct a room, while Voxblox and KinectFusion+ typically require 5MB and 30MB respectively

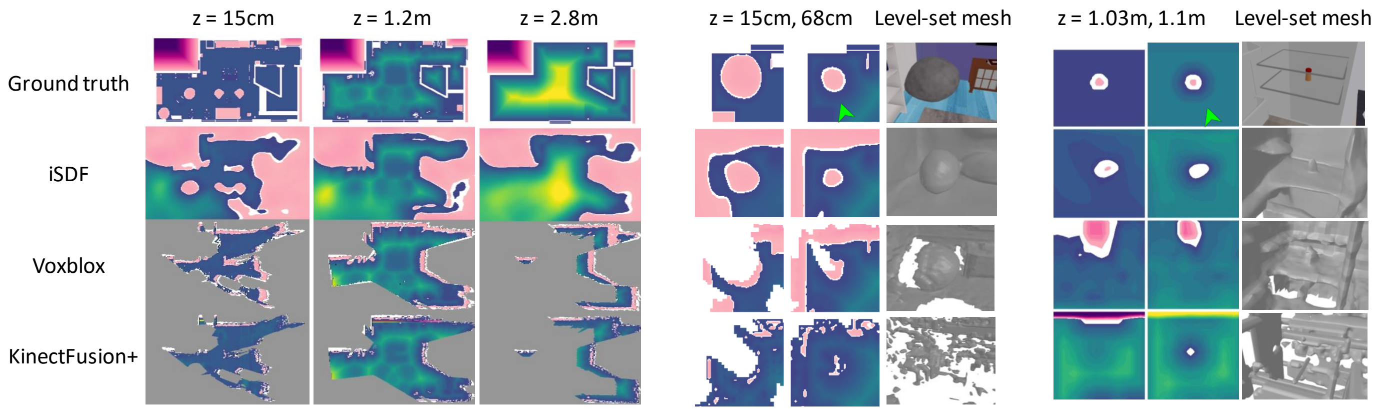

Fig. 7: All visualisations in this figure are produced using the reconstructed SDFs at the end of a sequence. *Left*: The reconstructed SDF at different heights for *apt_2_mnp*. The camera is mounted on the robot at a fixed pitch and height of around 1.2m (the height of the middle slices). The other 2 slices are at floor level (15cm) and above the camera (2.8m) and are mostly greyed out in the Voxblox and KinectFusion+ slices as there is limited visibility at these heights. *Middle*: SDF slices and the zero level-set mesh for the corner of the room in *apt_2_nav* around the beanbag. The mesh is viewed from the location of the green arrow in the ground truth slice. *Right*: Slices and zero level-set around the salt shaker for *apt_2_mnp*. During the sequence the robot navigates to and approaches the salt shaker, located in the fridge, as if it intends to manipulate it. iSDF is the only method that can reconstruct the backside of the beanbag and salt shaker in detail.

to store the voxel grids. The average time for a single iteration in iSDF is 33ms, with around 2ms for sampling, 2ms for computing bounds, 10ms for the forward pass, 19ms for the backwards pass.

On many robot platforms, compute must be shared between perception, planning, and various other modules. To study the effects of available compute we also evaluate all methods with half of the compute budget available in the previous real-time experiments. For iSDF, less compute means fewer optimisation steps per second, while for Voxblox and KinectFusion+ we must increase the voxel size to maintain real-time performance. We find that iSDF maintains the best performance with half the compute budget, and encourage the reader to see the supplementary document for more details.

D. *SDF supervision bounds*

We validate our choice of using the point batch distance to compute the SDF supervision bounds with an ablation study in the supplementary document. As expected, we find that tighter bounds lead to more accurate reconstructions. Using the batch distance gives the lowest SDF error, while the normal correction leads to better accuracy than the ray distance. There is a trade off between the tightness of the bound and computation time, however, computing the batch distance takes on average 2ms which is minimal in comparison to the 33ms for the full step.

VII. Limitations

There remain many challenges in training neural fields more effectively in real-time. Four important directions are introducing pretrained priors, handling dynamic environments, using local models to reduce replay, and designing more flexible positional embeddings to improve fine-detailed object reconstructions. This last point is particularly crucial for iSDF as the surfaces can be oversmoothed due to the choice of a low frequency positional embedding ($L = 5$) to fit the predominantly low frequency SDF in free space.

Additionally, the Eikonal equation is currently enforced by regularisation which does not handle singularities and leads to over-smoothing around singularities. An interesting direction would be to investigate constraining the function space to satisfy the Eikonal equation, while accounting for singularities. Lastly, for a downstream planner, it would be useful to have a measure of confidence associated with the iSDF predictions in unobserved regions.

VIII. Conclusion

We have presented iSDF, a continual learning system for real-time signed distance field reconstruction. Key to our approach is a novel self-supervised loss that bounds the predicted signed distance using the distance to the closest batch surface point to enable learning signed distances away from surfaces. Positive characteristics of iSDF include adaptive levels of detail, plausible filling in of partially observed regions, denoising of observations, and memory efficient representation. In evaluations against alternative methods on real and synthetic datasets, we show that iSDF produces more accurate reconstructions as well as better collision costs and gradients for downstream planners. In the near term, we hope that the generality and differentiability of iSDF means it can easily be integrated into downstream robotics applications from navigation to manipulation.

Acknowledgments

We would like to thank Andrew Davison and Zoe Landgraf for insightful discussions as well as Tom Whelan for initial help with datasets. This work was carried out while Joseph Ortiz interned at Meta AI.

References

[1] Michal Adamkiewicz, Timothy Chen, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeannette Bohg, and Mac Schwager. Vision-only robot navigation in a neural

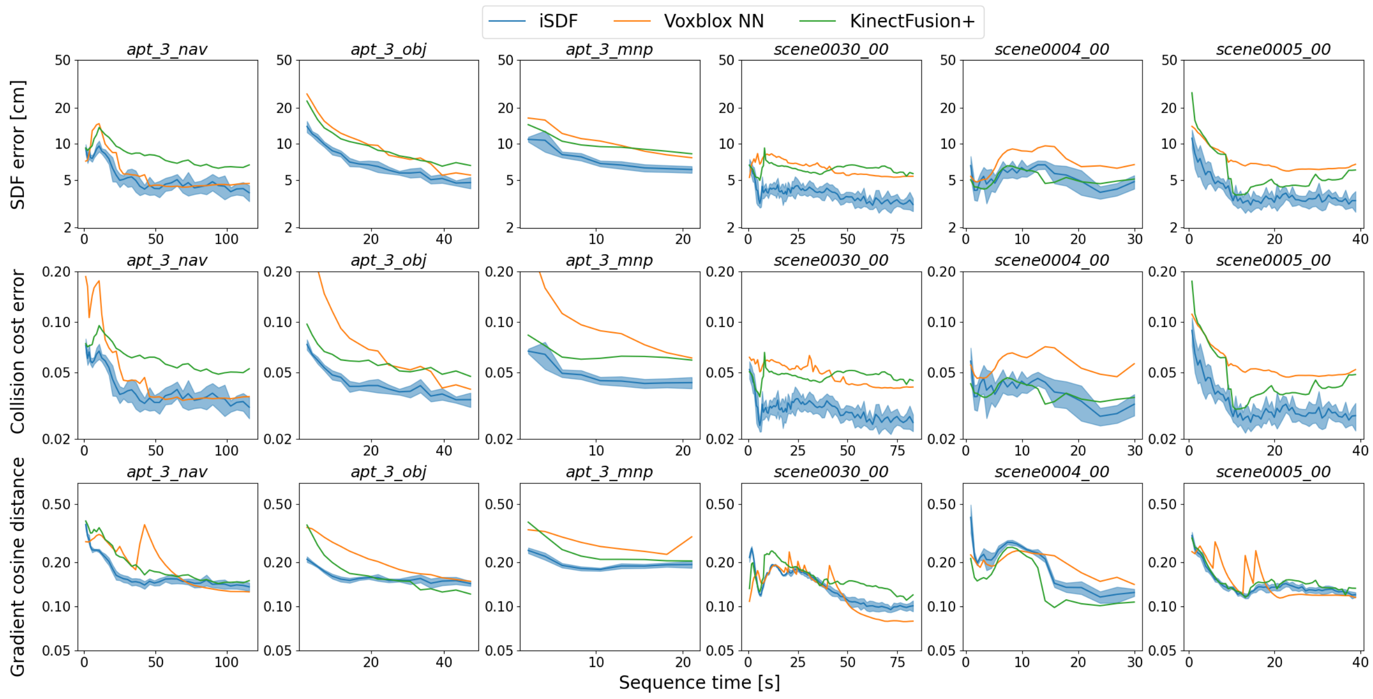

Fig. 8: We compare iSDF, Voxblox and KinectFusion+ along our three evaluation metrics: SDF error, collision cost error and gradient cosine distance. The metrics are evaluated at regular fixed intervals during the sequences with points sampled in the visible region at that time in the sequence. As Voxblox does not map the full visible region (see Fig. 5), we use nearest neighbour interpolation to evaluate the SDF error in unmapped regions. For the collision cost error, we allocate the surface cost $c(0)$ to unmapped regions as a robot would want to avoid unknown regions. In the supplementary document, we present the same plots but with all evaluation points in the Voxblox mapped region and show that there remains a similarly large performance gap between iSDF and other methods.

radiance world. *arXiv preprint arXiv:2110.00168*, 2021. URL Vision-onlyrobotnavigationinaneuralradianceworld.

[2] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2565–2574, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Atzmon_SAL_Sign_Agnostic_Learning_of_Shapes_From_Raw_Data_CVPR_2020_paper.html.

[3] Matan Atzmon and Yaron Lipman. SALD: Sign agnostic learning with derivatives. In *ICLR*, 2021. URL https://iclr.cc/virtual/2021/poster/3221.

[4] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural RGB-D Surface Reconstruction. *arXiv preprint arXiv:2104.04532*, 2021. URL https://arxiv.org/abs/2104.04532.

[5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. *arXiv*, 2021. URL https://arxiv.org/abs/2111.12077.

[6] Devendra Singh Chaplot, Deepak Pathak, and Jitendra Malik. Differentiable Spatial Planning using Transformers. In *ICML*, 2021. URL https://icml.cc/virtual/2021/poster/9101.

[7] S. Choi, Q. Zhou, and V. Koltun. Robust Reconstruction of Indoor Scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. URL https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Choi_Robust_Reconstruction_of_2015_CVPR_paper.html.

[8] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scene. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Dai_ScanNet_Richly-Annotated_3D_CVPR_2017_paper.html.

[9] Nikhil Das and Michael Yip. Learning-based proxy collision detection for robot motion planning applications. *IEEE Transactions on Robotics*, 36(4):1096–1114, 2020. URL https://arxiv.org/abs/1902.08164.

[10] Jing Dong, Mustafa Mukadam, Frank Dellaert, and Byron Boots. Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs. In *Robotics: Science and Systems*, volume 12, page 4, 2016. URL http://www.roboticsproceedings.org/rss12/p01.pdf.

[11] Danny Driess, Jung-Su Ha, Marc Toussaint, and Russ Tedrake. Learning models as functionals of signed-distance fields for manipulation planning. In *Conference on Robot Learning*, pages 245–255. PMLR, 2022. URL https://proceedings.mlr.press/v164/driess22a.html.

[12] P. F. Felzenszwalb and D. P. Huttenlocher. Distance

transforms of sampled functions. Technical report, Cornell Computing and Information Science, 2004. URL https://www.cs.cornell.edu/~dph/papers/dt.pdf.

[13] Mark Nicholas Finean, Wolfgang Merkt, and Ioannis Havoutis. Predicted Composite Signed-Distance Fields for Real-Time Motion Planning in Dynamic Environments. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, pages 616–624, 2021. URL https://ojs.aaai.org/index.php/ICAPS/article/view/16010.

[14] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit Geometric Regularization for Learning Shapes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3569–3579, 2020. URL https://arxiv.org/abs/2002.10099.

[15] Luxin Han, Fei Gao, Boyu Zhou, and Shaojie Shen. Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 4423–4430. IEEE, 2019. URL https://arxiv.org/abs/1903.02144.

[16] Rafi Hayne, Ruikun Luo, and Dmitry Berenson. Considering avoidance and consistency in motion planning for human-robot manipulation in a shared workspace. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3948–3954. IEEE, 2016. URL https://ieeexplore.ieee.org/abstract/document/7487584.

[17] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1251–1261, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Jiang_SDFDiff_Differentiable_Rendering_of_Signed_Distance_Fields_for_3D_Shape_CVPR_2020_paper.html.

[18] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *arXiv preprint arXiv:2202.08227*, 2022.

[19] 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. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Liu_DIST_Rendering_Deep_Implicit_Signed_Distance_Function_With_Differentiable_Sphere_CVPR_2020_paper.html.

[20] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Mescheder_Occupancy_Networks_Learning_3D_Reconstruction_in_Function_Space_CVPR_2019_paper.html.

[21] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. URL https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123460392.pdf.

[22] Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time Gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11):1319–1340, 2018. URL https://journals.sagepub.com/doi/10.1177/0278364918790369.

[23] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011. URL https://ieeexplore.ieee.org/document/6162880.

[24] Richard A Newcombe, Dieter Fox, and Steven M Seitz. DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. URL https://openaccess.thecvf.com/content_cvpr_2015/html/Newcombe_DynamicFusion_Reconstruction_and_2015_CVPR_paper.html.

[25] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2017. URL https://arxiv.org/abs/1611.03631.

[26] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Park_DeepSDF_Learning_Continuous_Signed_Distance_Functions_for_Shape_Representation_CVPR_2019_paper.html.

[27] Victor Reijgwart, Alexander Millane, Helen Oleynikova, Roland Siegwart, Cesar Cadena, and Juan Nieto. Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps. *IEEE Robotics and Automation Letters*, 5(1):227–234, 2019. URL https://arxiv.org/abs/2004.13154.

[28] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014. URL https://journals.

sagepub.com/doi/abs/10.1177/0278364914528132.

[29] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *Proc. NeurIPS*, 2020. URL https://nips.cc/virtual/2020/public/poster_53c04118df112c13a8c34b38343b9c10.html.

[30] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. iMAP: Implicit Mapping and Positioning in Real-Time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Sucar_iMAP_Implicit_Mapping_and_Positioning_in_Real-Time_ICCV_2021_paper.html.

[31] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training Home Assistants to Rearrange their Habitat. *arXiv preprint arXiv:2106.14405*, 2021. URL https://arxiv.org/abs/2106.14405.

[32] Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1049–1056, 2009. URL https://icml.cc/Conferences/2009/papers/271.pdf.

[33] Mrinal Verghese, Nikhil Das, Yuheng Zhi, and Michael Yip. Configuration Space Decomposition for Scalable Proxy Collision Checking in Robot Planning and Control. *IEEE Robotics and Automation Letters*, 2022. URL https://arxiv.org/abs/2201.04314.

[34] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *arXiv preprint arXiv:2106.10689*, 2021. URL https://arxiv.org/abs/2106.10689.

[35] Youngsun Wi, Pete Florence, Andy Zeng, and Nima Fazeli. Virdo: Visio-tactile implicit representations of deformable objects. *arXiv preprint arXiv:2202.00868*, 2022.

[36] Zike Yan, Yuxin Tian, Xuesong Shi, Ping Guo, Peng Wang, and Hongbin Zha. Continual Neural Mapping: Learning An Implicit Scene Representation from Sequential Observations. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. URL https://openaccess.thecvf.com/content/ICCV2021/papers/Yan_Continual_Neural_Mapping_Learning_an_Implicit_Scene_Representation_From_Sequential_ICCV_2021_paper.pdf.

[37] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Neural Information Processing Systems (NIPS)*, 2021. URL https://proceedings.neurips.cc/paper/2020/hash/1a77befc3b608d6ed363567685f70e1e-Abstract.html.

[38] Jingyang Zhang, Yao Yao, and Long Quan. Learning signed distance field for multi-view surface reconstruction. *arXiv preprint arXiv:2108.09964*, 2021.

[39] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. NICE-SLAM: Neural Implicit Scalable Encoding for SLAM. *arXiv preprint arXiv:2112.12130*, 2021. URL https://arxiv.org/abs/2112.12130.

[40] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013. URL https://journals.sagepub.com/doi/10.1177/0278364913488805.

Appendix

KinectFusion+

Here we describe in detail our baseline KinectFusion+, a GPU based method for real-time SDF reconstruction. KinectFusion+ operates in two stages. In the first stage, we use our own implementation of KinectFusion [23] to fuse depth measurements into an occupancy grid.

In the second stage, the occupancy grid is transformed to a signed distance field using the Euclidean Distance transform (EDT). The signed distance field is computed by subtracting the EDT of the inverse occupancy field from the EDT of the occupancy field. We employ the efficient EDT algorithm from Felzenszwalb and Huttenlocher [12] which has time complexity $O(n)$, where n is the grid size. This algorithm first computes the squared EDT before taking the square root using the insight that the computation of the squared EDT can be separated out along each dimension.

One important detail in KinectFusion+ is the choice to initialise all voxels as unoccupied. With this choice, regions inside objects that cannot be directly viewed will remain as unoccupied and SDF values in these regions should not be used in downstream tasks. In this way, much like Voxblox, KinectFusion+ only maps the shells of objects and cannot be queried inside objects. The alternative of initialising voxels as occupied is overly conservative and leads to highly inaccurate SDF values, although the interiors of objects can now be queried. The reason for the inaccurate predictions is that in free space, the closest occupied voxel may now be an unobserved voxel, meaning the signed distances are greatly underestimated.

Additional Results

A. Reduced Compute Budget Experiment

On many robot platforms, the compute budget must be shared between perception, planning and other modules. To simulate this constraint, in Fig. 9 we compare the SDF error when the compute budget available in the main real-time experiments is halved. With half the compute budget, the training speed of iSDF is reduced from around 30 iterations per second to 15. With a reduced compute budget, Voxblox and KinectFusion+ must operate with larger voxel sizes to maintain real-time performance. For Voxblox the voxel size increases from 5.5cm to 7.8cm, and for KinectFusion+ it increases from 7cm to 8cm.

In this reduced compute domain, we find that iSDF retains the best performance on 5 of the 6 sequences. For *scene0004_00*, which observes a particularly large environment in a short time, the iSDF reconstruction is less accurate.

B. SDF Supervision Bounds ablation

To validate our choice of using the point batch distance to compute the SDF supervision bounds we conduct an ablation study in Fig. 10. We find that the point batch distance produces the most accurate SDF reconstruction for all sequences. Using the normal correction improves on the ray bounds but to

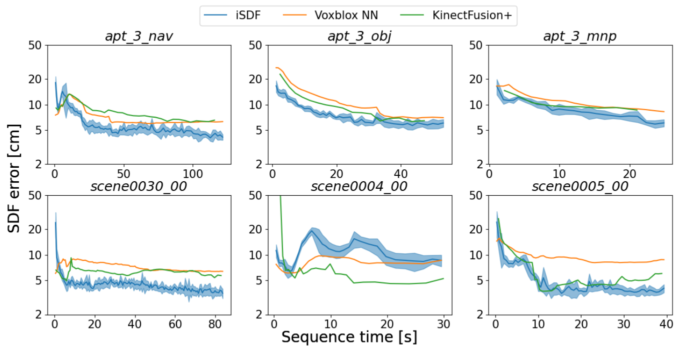

Fig. 9: We compare the SDF error for iSDF, Voxblox and KinectFusion+ when the compute budget is halved.

a lesser extent than using the batch distance. These results are expected as the normal correction generally gives tighter bounds than the ray distance while the batch distance bounds are guaranteed to be at least as tight as the ray bounds.

There is a trade off between the tightness of the bound and computation time. The average time taken to compute the supervision bounds for the ray, normal and batch distance methods is: 0.2ms, 0.6ms, 2.3ms. This time is negligible in comparison to the average total iteration time of 33ms, meaning that using the batch distance bounds has little effect on the number of steps per second.

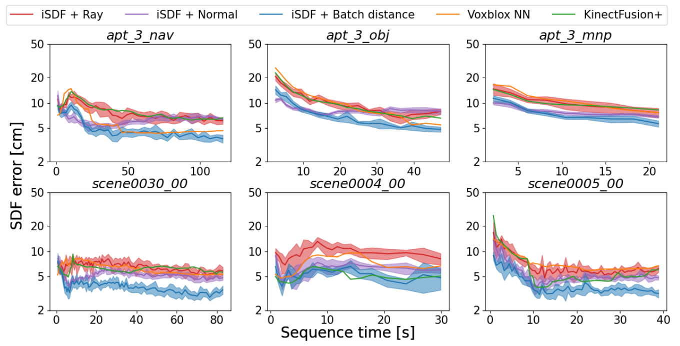

Fig. 10: SDF error for iSDF with the different supervision bounds and the baseline methods. As expected, performance improves with tighter bounds from ray, to normal to batch distance. iSDF with the point batch distance for the bound gives the lowest error.

C. Evaluation in Voxblox mapped region

As discussed in the main paper, the Voxblox reconstruction has significant holes in the visible region. These holes are caused by discarding rays during TSDF fusion that pass through voxels that have already been updated by rays in the same image. This simplification, which is necessary for real-time performance on a CPU, causes large holes in regions distant from the camera that are partially occluded by foreground objects.

One way to avoid these holes would be to use a GPU based TSDF fusion method in the first stage, that does not discard rays, and then use Voxblox's wavefront propagation algorithm to transform the resulting TSDF to SDF. To approximately evaluate against such a hypothetical system, we compare the existing methods along our three metrics evaluated at points only in the Voxblox mapped region in Fig. 11. The evaluation is conducted in the same manner as in the main paper by sampling along rays in the visible region, however points outside of the Voxblox mapped region are discarded.

In the Voxblox mapped region, iSDF still produces the most accurate SDFs and best approximations of collision costs. As expected, the performance of Voxblox improves relative to the other methods, however it still records higher SDF error than iSDF on all sequences. Voxblox also produces the least accurate gradients, with iSDF and KinectFusion+ producing similarly accurate gradient fields in this region.

D. SDF error binned by distance to the surface

To understand why iSDF achieves a lower SDF error than Voxblox and KinectFusion+, we break down the SDF error into bins according to the true signed distance for *scene0030_00* in Fig 12. We observe that far from surfaces iSDF and KinectFusion+ are significantly more accurate than Voxblox, while close to surfaces iSDF and Voxblox are the most accurate methods. iSDF therefore has the lowest average SDF error as is the only method that has low SDF error both close to and far from surfaces. Although we only show results here for one sequence, other sequences show a similar trend.

E. Results for 6 additional sequences

In this section, we present results for the 6 sequences (3 ReplicaCAD [31] and 3 ScanNet [8] sequences) that were not used in the main paper. These figures provide further evidence for the claims made in the paper and so we simply show them here and refer the reader back to the main paper for the discussion.

In Fig. 13 we compare slices at constant height of the reconstructed SDF at the end of the sequence. As in the main paper, we see that iSDF produces more complete and accurate slices. In Fig. 14 we compare the performance along our three evaluation metrics. iSDF consistently has the lowest SDF and collision cost error, while KinectFusion+ produces similarly accurate gradients.

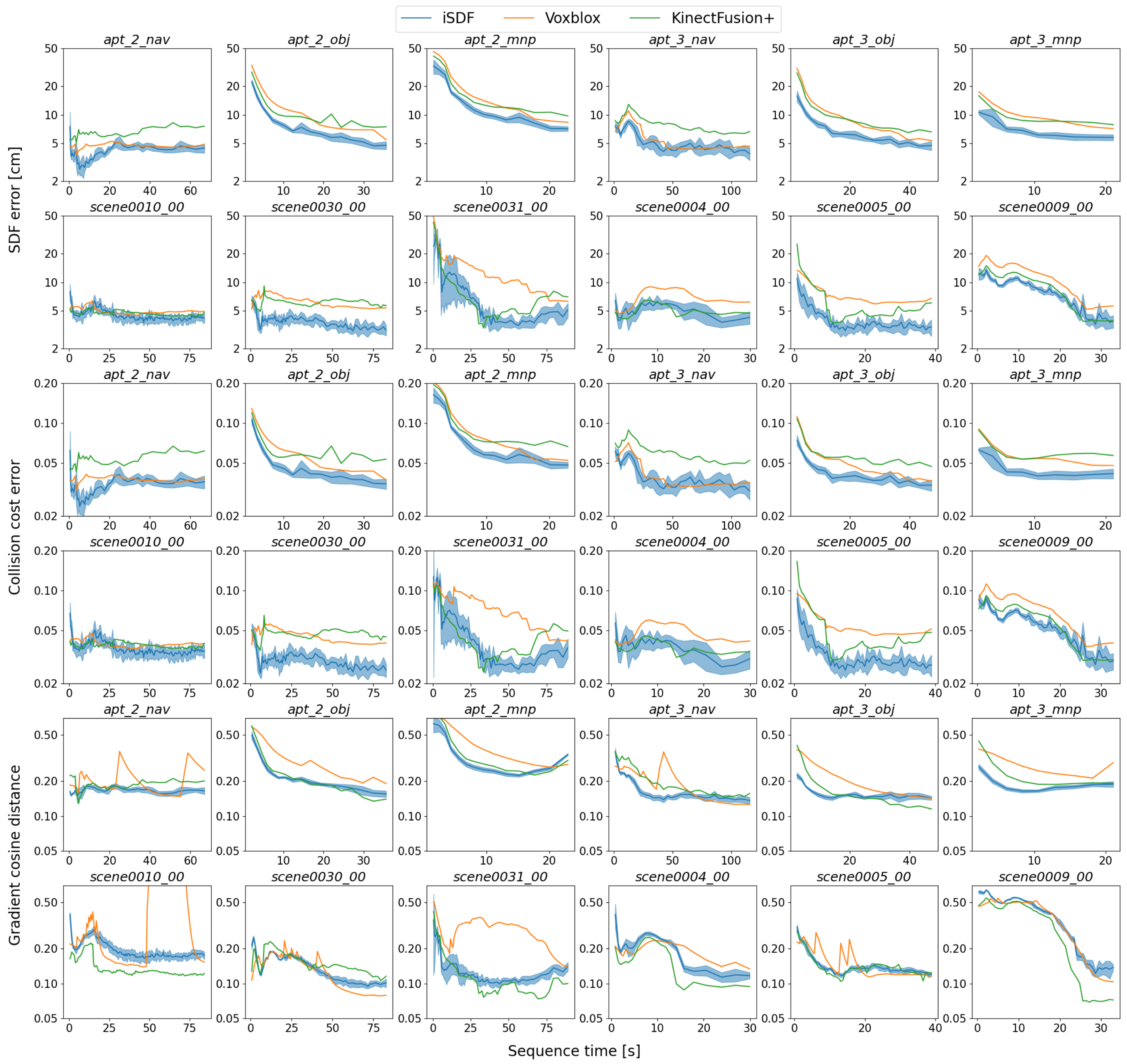

Fig. 11: We compare the SDF error evaluted only at points in the Voxblox mapped region. As a result, we discard points from the evaluation in the most challenging regions to reconstruct that are heavily occluded and distant from the camera.

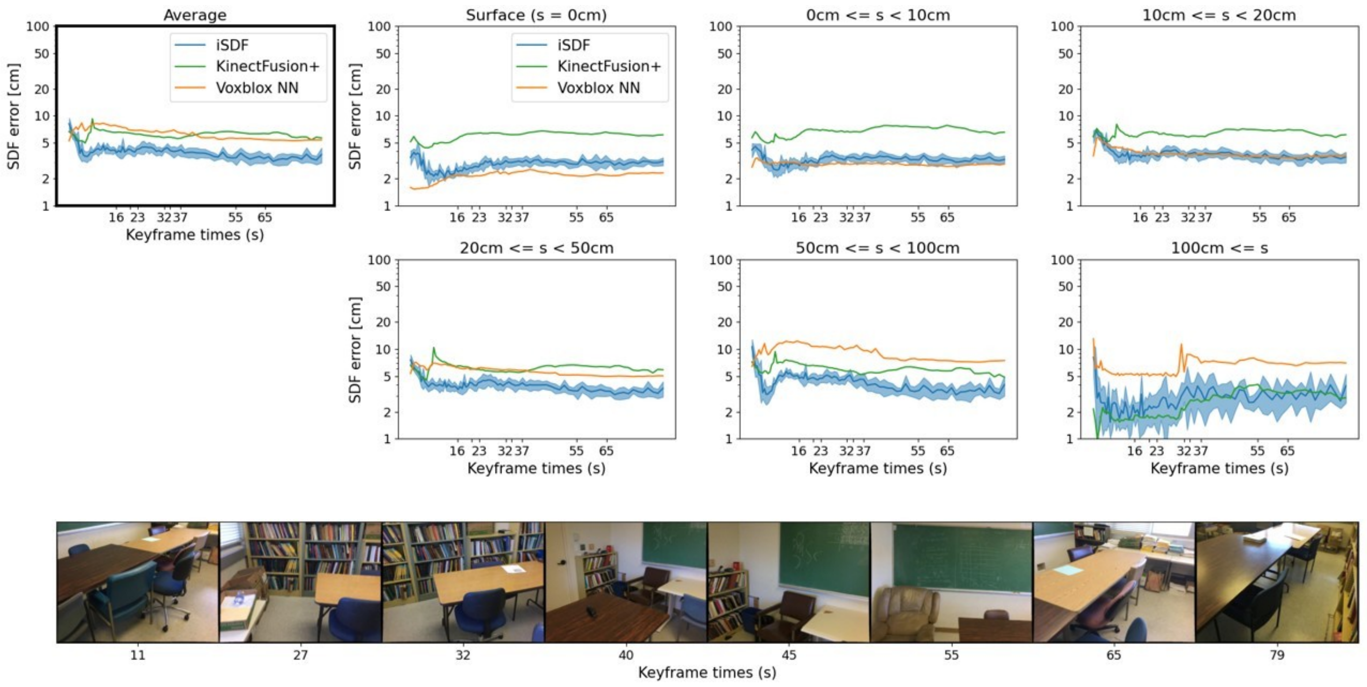

Fig. 12: We break down the average SDF error into bins according to the true signed distance (s). There are no bins for negative SDF values as for the ScanNet dataset we can only compute ground truth signed distances in regions outside of objects. In the bottom row, we show 8 of the keyframes selected by iSDF during the sequence *scene0030_00*.

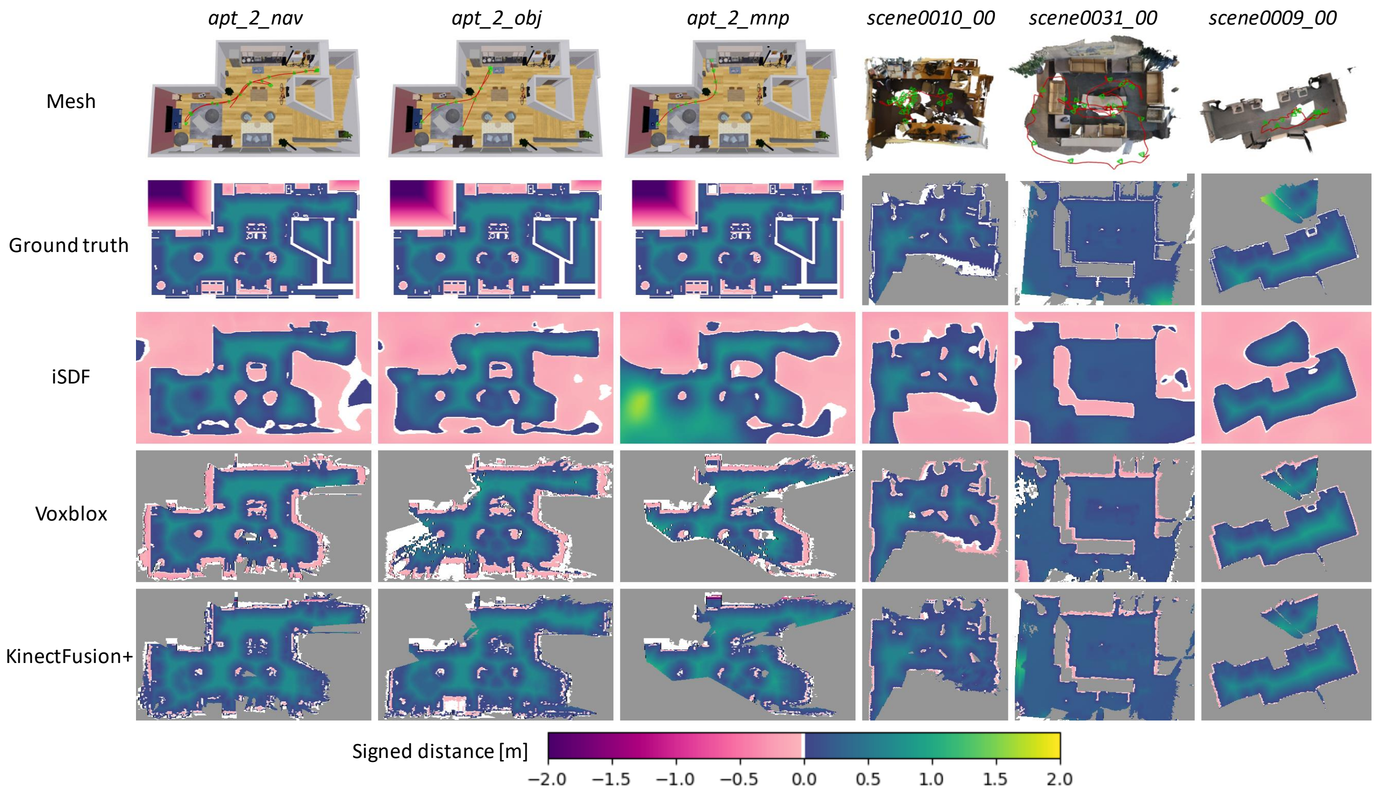

Fig. 13: **SDF slices**. Slices at constant height of the reconstructed SDF at the end of the sequence. The meshes are shown for reference with the camera trajectory and keyframes selected by iSDF overlaid. For Voxblox and KinectFusion+, the slices are greyed out in the non-visible region as neither method makes predictions in this region. The ground truth ScanNet [8] slices are also greyed out in the non-visible region as we only have ground truth SDF values in the visible region. White regions in the Voxblox and KinectFusion+ slices are regions that are visible but that are unmapped (i.e. no rays have reached this region). Note that in *scene0009_00* there is a mirror, causing all methods to reconstruct the reflection of the room.

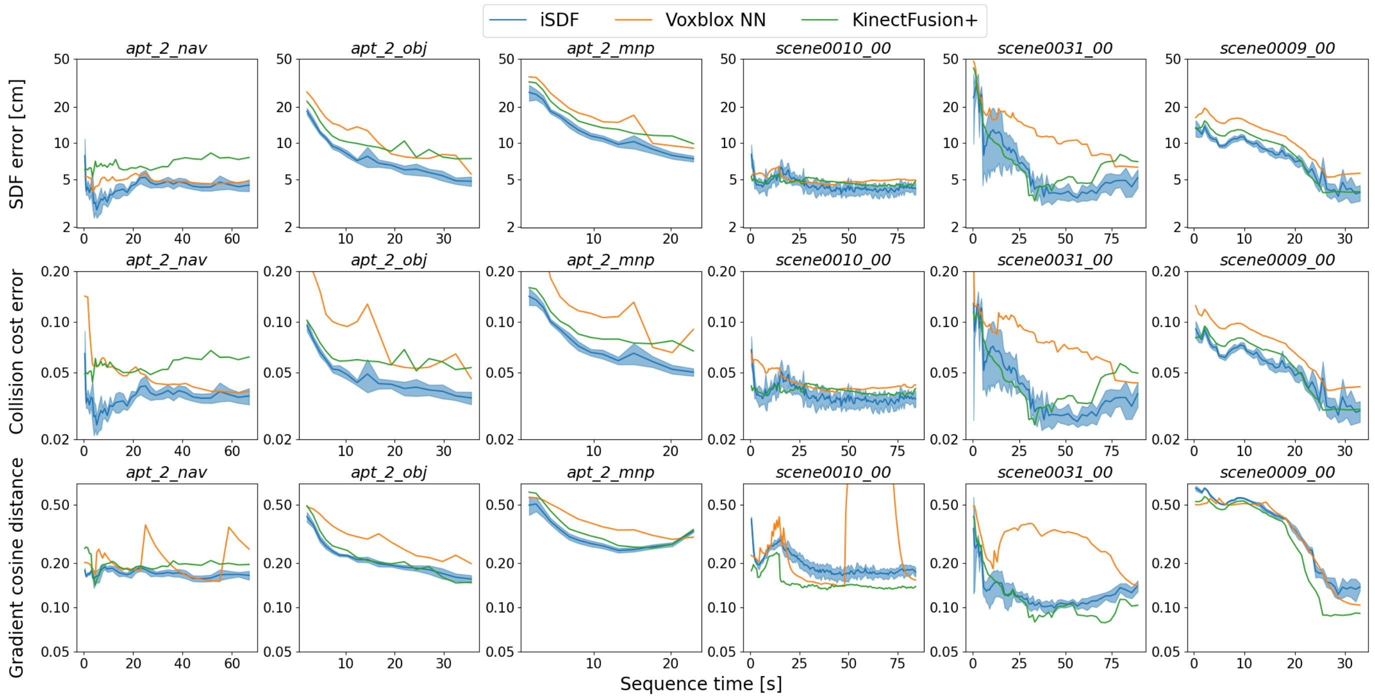

Fig. 14: We compare iSDF, Voxblox and KinectFusion+ along our three evaluation metrics - SDF error, collision cost error and gradient cosine distance. The metrics are evaluated at regular fixed intervals during the sequences with points sampled in the visible region at that point in the sequence. As Voxblox does not map the full visible region, we use nearest neighbour interpolation to evaluate the SDF error in unmapped regions. For the collision cost error, we allocate the surface cost $c(0)$ to unmapped regions as a robot would want to avoid unknown regions. In *scene0010_00*, Voxblox records very high gradient error in a duration of the sequence when there are many gaps in the Voxblox reconstruction.