UC San Diego

UC San Diego Electronic Theses and Dissertations

**Title**

Neural Implicit Scene-Level Surface Representation

**Permalink**

https://escholarship.org/uc/item/417733mz

**Author**

PASKAL, ALEXANDER

**Publication Date**

2023

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library University of California

UNIVERSITY OF CALIFORNIA SAN DIEGO

# Neural Implicit Scene-Level Surface Representation

A thesis submitted in partial satisfaction of the
requirements for the degree Master of Science

in

Electrical Engineering (Medical Devices and Systems)

by

Alexander Paskal

Committee in charge:

Professor Nikolay Atanasov, Chair
Professor Xiaolong Wang
Professor Michael Yip

2023

Copyright

Alexander Paskal, 2023

All rights reserved.

The thesis of Alexander Paskal is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2023

DEDICATION

Dedicated to Professor Atanasov for his boundless patience, and to HoJoon for sticking it out in the trenches with me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First I would like to acknowledge Professor Nikolay Atanasov for giving me the opportunity to work on such interesting problems and for mentoring me through the process - my experiences in the lab under his guidance this past year have been nothing short of transformative for me and have been fundamental in shaping me as an academic and as a person.

Second I would like to acknowledge Hojoon Shin for working through this project with me every step of the way. He has been an exemplary researcher and an excellent friend, and I know that I would not have gotten nearly as far nor enjoyed the process nearly as much without his help.

I would also like to acknowledge Zhirui Dai for answering all of our questions and lending his experience and advice, and Peiran Liu for jumping in and helping us explore some of the problems we would not have otherwise been able to get to.

Finally I would like to acknowledge everyone at the Existential Robotics Laboratory for creating such a positive and welcoming environment

Chapter 4 section 4 is, in part, coauthored with Shin, Hojoon. The thesis author was the primary author of this section.

ABSTRACT OF THE THESIS

Neural Implicit Scene-Level Surface Representation

by

Alexander Paskal

Master of Science in Electrical Engineering (Medical Devices and Systems)

University of California San Diego, 2023

Professor Nikolay Atanasov, Chair

Neural implicit surface representations are a promising new development in surface modeling. However, the challenges inherent in training neural networks in a continual fashion are still holding them back from being widely used in real-time, incremental scene mapping. We propose a method for learning a neural representation of a signed distance function from trajectories of posed depth images that is both computationally efficient and avoids the problem of catastrophic forgetting. We demonstrate our approach by producing high-quality scene reconstructions in 2D and 3D and incrementally building 2D neural-implicit maps.

# Chapter 1

# Introduction

## 1.1 Background

Mapping is a fundamental task in robotics. Given a trajectory of data from various sensor modalities, robots need to be able to reconstruct a representation of the environment, which can then be used for motion planning, collision detection, view rendering and higher-level reasoning.

There are different type of mapping representations in use, each with their own benefits and drawbacks. The most common forms of representations are continuous explicit representations such as point clouds and meshes, or discretized representations such as occupancy maps. Also common are hierarchical representations such as octrees.

An alternative to explicit mapping representations are implicit ones, where the map is represented as a function mapping coordinate values to a scalar function value. A very common implicit representation of an environment is a euclidean distance field (EDT), which maps the distance of any point in the environment to the nearest surface boundaries. EDTs and their variants are popular map representations because of a few nice properties. First, they allow for surfaces to be easily reconstructed, since the surface is the 0-level set of the function. Second, they are useful in motion planning since they offer efficient collision detection. However, they can be difficult or impossible to compute explicitly for complex environments.

A recent revolution in the field of 3D computer vision is the advent of neural implicit modeling, which involves using a neural network as a function approximator to learn an implicit mapping of an environment. This approach was first popularized as a method of learning representations of objects or classes of objects, and for inverse rendering tasks. More recently it is being applied to the problem of map reconstruction from a trajectory. The benefit of these neural implicit maps is that they are very memory-efficient, being able to represent rooms in arbitrary degrees of detail. They also produce watertight meshes from incomplete observations, which is a desirable property for rendering.

This work focuses on one variant of EDFs known as signed distance fields (SDFs) and their uses in shape modeling and scene reconstruction. We explore different methods of training neural networks to represent SDFs and their results in 2D and 3D. We also explore and compare approaches for learning the SDF of a scene-level environment from an observed trajectory, and offer an approach for training an SDF in real-time in an incremental, self-supervised fashion.

## 1.2 Related Works

### 1.2.1 Neural Implicit Surface Representations

Implicit functions have been widely used in computer vision and graphics as surface representations. Methodologies for explicitly computing the distance fields of objects exist such as the Fast Marching Method, but can be computationally difficult to perform in real time for complex surfaces.

One of the foundational works in neural implicit surface representations was Occupancy Networks by Mescheder et. al., in which they explored using neural networks for representing 3D surfaces as a continuous decision boundary[6]. They demonstrate that complex 3D structures can be encoded in the parameters of a coordinate-based MLP mapping cartesian coordinates to an occupancy prediction, and were able to learn these shape representations from various forms of data. While this approach was fundamental in laying the groundwork for neural implicit modeling as a subfield, the constrained nature of the training methodology and limitation to learning a single shape per network prevented it from receiving widespread use.

This approach of using neural networks to learn implicit surface functions was extended by Part et. al. in their work DeepSDF, in which they trained coordinate MLPs on sampled data to regress signed distance fields directly[10]. They also extended the approach to learn MLP decoders for classes of shapes, and infer latent code representations for a particular instance of a shape class using a novel ”auto-decoder” framework. These results represented an important step in demonstrating the utility of neural implicit representations for shape inference and remain influential in the short history of the field. However, the necessity of human-labeled samples to train on as well exorbitant training times were seen as major drawbacks of the method.

An approach for reducing the long training times of DeepSDf was presented by Chabra et. al. in their work Deep Local Shapes, where they rectify this approach by storing a grid of localized latent codes and learning each independently[3]. This enabled much more efficient learning of each independent neighborhood and higher levels of detail, at the expense of increased memory storage.

An alternative approach for learning signed distance fields directly from point cloud representations called implicit geometric regularization (IGR) was presented by Gropp. et. al., in which they add regularization terms to the loss function that enforced properties of signed distance fields such as eikonality and gradient-normal correspondence at the surface[4]. They train their networks directly on point cloud data without any additional sampling away from the surface and allow the signed distance field to ”grow” from the surface in such a way that it satisfies these constraints. While IGR-nets are very beneficial in that they don’t require any labeled signed distance field data and thus are a form of unsupervised learning, they still suffered from long training times as well as poor estimations far from the surface.

An important breakthrough in the architectures of neural networks was made by Sitzmann. et. al., with the introduction of periodic activation functions[12]. They demonstrate that by using sinusoidal activation layers in their networks, implicit neural representations are able to represent higher-frequency surfaces as opposed to using ReLU activations. They demonstrate that their networks are able to reconstruct scenes with high levels of detail.

An interesting extension to the signed distance function is the signed directional distance function (SDDF), which uses an MLP to regress a distance value from a cartesian coordinate as well as a viewing direction. The inclusion of direction presents a significant complication as it introduces general discontinuities in the underlying implicit function that the network is trying to model, as well as increasing the dimensionality of the input domain. This is still an open problem in research. One formulation is presented by Zobeidi and Atanasov in their work DeepSDDF, who are able to learn SDDF models for shapes and classes of shapes by introducing a novel coordinate transformation that ensures the network prediction satisfies the eikonal constraint along a given viewing direction[23]. While these results are a promising first step, they depend on observing an object from the outside in and thus as presented are not trivially extended to scene-level representations.

### 1.2.2 Implicit Scene Reconstruction

Network representation for scenes is more difficult than for objects as scenes are inherently more complex, thus stressing the computational capacity of the MLP representation. Furthermore, ground truth shape information often is not directly available for scene reconstructions, making the ability to reconstruct scenes from observed sensor data (typically LIDAR or RGB-D images) an important quality of a neural implicit model.

A work presented by Sitzmann et. al. called Scene Representation Networks (SRNs) introduces the concept of differential rendering, which optimizes a scene representation by directly supervising the reconstruction of observed images, without access to underlying 3D geometry[13]. SRNs regress 3D coordinates to feature representations of the scene at that particular coordinate, and then are able to update this representation using a differentiable ray-marching algorithm to predict a camera view at a given coordinate and then supervising with observed views and doing backpropagation to learn the network weights. Differentiable rendering was a novel idea and an extremely important contribution to the field - however, the implementation of SRNs is extremely inefficient, with some models taking on the order of weeks to train.

The concept of a differentiable rendering pipeline was expanded by Mildenhaal et. al. in their work Neural Radiance Fields (NeRF), which modifies the implicit representation to represent a radiance field which maps a coordinate and a viewing direction to the predicted radiance density and color of the coordinate when observed from that direction[7]. They also introduced a positional embedding to help the network represent high frequency details, and show the effectiveness of this network for generating photo-realistic novel views. This work was highly influential and has spawned a number of spinoffs which optimize the fundamental approach for different use cases such as large-scale reconstruction, incremental scene mapping and efficient inference [2], [21]. However, while NeRFs are extremely effective for rendering novel views, extracting ground truth geometry from them is an unwieldy process, limiting their usefulness as an implicit representation in the field of robotics.

in VolSDF, Yariv et. al. present a methodology for learning signed distance fields directly through differentiable rendering pipelines by transforming the signed distance field into a density prediction by applying Laplace's cumulative distribution function, and were able to reproduce high quality scene surface reconstructions[19]. This is an important step in scene reconstruction with signed distance fields - however, the reconstruction process is still inefficient and they do not present a methodology for learning incrementally.

In order to make these differentiable rendering pipelines more efficient, Muller et. al. extend the idea presented by (DeepLocalShapes) introduce hash-tables encoding latent vector grids in their work Instant Neural Graphics Primitives (NGPs) [8]. They introduce trilinear interpolation as a method for inferring the latent code of a specific coordinate within a rectilinear grid of feature vectors, and using this interpolated code to infer implicit functions. The inclusion of this feature grid dramatically decreased training time as it parallelized updates to local regions of space, and represented a similar jump in efficiency as from DeepSDF to DeepLS.

### 1.2.3 Incremental Learning for Neural Implicit Representations

In order to the neural implicit representations to be maximally useful in the field of robotics and embodied AI, it is important to have methods for updating a neural implicit model as new observations are obtained. This is especially difficult with neural implicit models, as neural networks are known to suffer from catastrophic forgetting

Once of the first real attempts at real-time, online SLAM with neural implicit models was iMAP, in which Sucar et. al. present an approach to incrementally learn scene geometry by adopting the same differentiable rendering pipeline and continuously optimizing both network parameters and estimated trajectories[14]. They address the continual learning problem by maintaining an experience replay buffer, selecting novel images based on information gain and actively orienting training data towards areas generating large error. This represented an excellent first step in the field of incremental, online neural implicit SLAM. However, iMAP was not accurate and relied on a highly optimized, complex and cumbersome implementation in order to achieve real-time mapping speeds.

The experience replay approach for incremental learning is applied by Ortiz. et. al. to signed distance fields in their work iSDF. They estimated a signed distance field using a sampling approach from depth images, and then directly supervise the network on sampled data[9]. They maintain a lighter network size to allow to network to run in real-time on much less sophisticated hardware, with significantly fewer implementation details. However, these results came at the heavy cost of accuracy, with reconstructions produced by iSDF being oversmoothed and often incomplete, an inherent flaw in the experience replay approach since updates made to the MLP representation are always global. While active training approaches are offered to combat this, in practice they result in an over-averaging of surface representations.

An alternative to the experience replay approach for incremental learning is presented in the work NICE-SLAM, where Zhu et. al leverage the local feature grids and trilinear interpolation scheme used in DeepLS and leverage a differential rendering front end to learn maps incrementally. By freezing the decoding layer, NICE-SLAM is able to make local updates by only updating the features vectors queried in the linear interpolation of a given cartesian point. This approach is effective and implements a structural change that eliminates the problem of catastrophic forgetting (as opposed to merely attempting to combat it with experience replay) but is slow and computationally intensive, as they used many multi-level feature grids and are bound to the slow differential-rendering pipeline.

## 1.3 Contributions

In this work we propose a system for learning full metric SDFs of scenes using neural implicit networks. We propose a sampling method for estimating the signed distance fields accurately in a self-supervised manner directly from observed depth images, and for obtaining high fidelity surface reconstructions. We demonstrate the efficacy of our approach by reconstructing signed distace fields from trajectories rendered in 2D and 3D. We also demonstrate that by leveraging implicit feature grids and trilinear interpolation, we are able to learn accurate, high-quality signed distance field representations incrementally without suffering from catastrophic forgetting. We show that we can effectively learn a decoding prior online without pretraining and incrementally make updates to our feature grid representation, and demonstrate our results in 2D.

# Chapter 2

# Implicit Surface Representations

## 2.1 Common Implicit Functions

### 2.1.1 Occupancy Fields

Perhaps the most intuitive type of implicit function is a binary classifier which takes in continuous coordinates and outputs a binary label:

$$f(x,y,z) : \mathbb{R}^n \to \{0,1\} \tag{2.1}$$

This idea is a natural extension of occupancy grids to continuous space, and formed the basis for neural implicit shape research. However, occupancy grids don't provide any useful information about the surface beyond the decision boundary, and as such using them for important tasks such as inverse rendering and collision detection is computationally inefficient.

### 2.1.2 Radiance Fields

An alternative to the occupancy network representation that has gained tremendous popularity in recent years is the radiance field representation. Here the implicit function maps a continuous coordinate and viewing direction to a density and color value, and constructs a probabilistic model of the geometry and particle effects of a scene. Specifically, for coordinate $\mathbf{x} = (x,y,z)$ and viewing direction $(\theta, \phi)$

$$f(x, y, z, \theta, \phi) = (\alpha, C) \tag{2.2}$$

where $\alpha$ denotes the observed density and $C$ denotes the observed color at that point and viewing angle, respectively.

These networks are typically used for inverse rendering and leverage a technique known as alpha compositing, which involves sampling a viewing ray along different points and estimating the final viewing color along a given viewing direction as an average of the colors of each of the points, weighted by their respective alpha values. This process is differentiable, which allows for neural networks to learn radiance fields directly from images by using backpropagation. This process is known as differentiable rendering. While radiance fields are excllent for novel view synthesis, and have been proven to encode accurate geometry priors [7], the extraction of surface geometry is computationally intensive and thus not suitable for online use.

## 2.2 Signed Distance Fields

A very common implicit function used in surface modeling is a signed distance function, which maps a coordinate to the distance from that coordinate to the nearest surface coordinate. The sign of the scalar output denotes whether that coordinate is found. An example of an SDF can be seen in **Figure** 2.1

Let us partition cartesian space into a set of coordinates in free space and a set of coordinates in occupied space, and let us define $\Omega \subset \mathbb{R}^n$ as the former set of coordinates in free space. A function $f$ is a signed distance function if it satisfies the following definition.

$$f(\mathbf{x}) = \begin{cases} d(\mathbf{x}, S) & \text{if } \mathbf{x} \in \Omega \\ -d(\mathbf{x}, S) & \text{otherwise} \end{cases} \tag{2.3}$$

where $d(\mathbf{x}, S)$ is the euclidean distance transform:

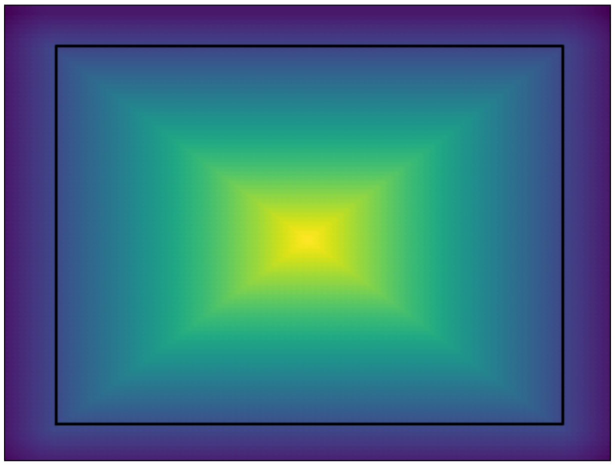


**Figure 2.1.** An example of a signed distance field for a square. The eikonal property of distance fields results in a very noticeable skeleton at points in the field where the nearest surface point is non-unique: such a point is called a locus point.

$$d(\mathbf{x}, S) := \inf_{\mathbf{y} \in S} ||\mathbf{x} - \mathbf{y}||_2 \tag{2.4}$$

we can define the surface represented by a signed distance function as the following:

An implicit surface $S$ is a surface in n-dimensional euclidean space represented by a signed distance function $f$ such that:

$$S = \{\mathbf{x} \in \mathbb{R}^n \mid f(x) = 0\} \tag{2.5}$$

if $S$ is piecewise smooth, then the signed distance function $f$ is differentiable almost everywhere (with exception at the cut locus or set of points whose closest surface point is non-unique) and its gradient has a unit norm:

$$|\nabla_{\mathbf{x}} f(\mathbf{x})| = 1 \tag{2.6}$$

This characteristic is known as the eikonal constraint. At any point on the surface boundary, the gradient of the normal of function is equivalent to the surface normal of $S$ at that point. This relationship is depicted in **Figure 2.2**.

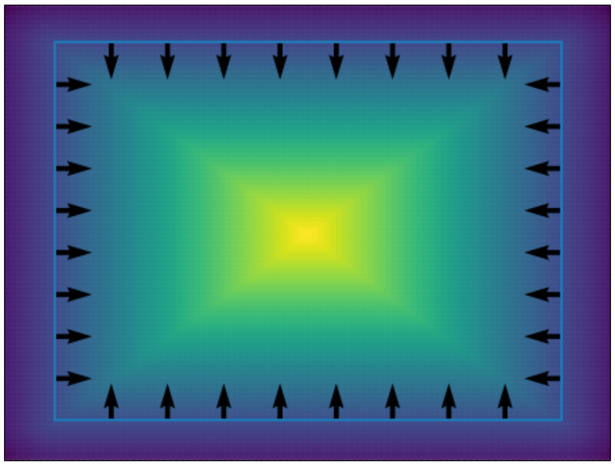


**Figure 2.2.** A depiction of the surface normals of a signed distance field in 2D. Note how the normals align directly with the gradient of the field. This is a useful property in learning signed distance fields from surface data.

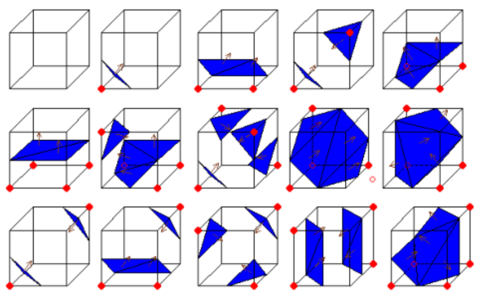
$$f(\mathbf{x}) = 0 \Longleftrightarrow \nabla_{\mathbf{x}} f(\mathbf{x}) = N(\mathbf{x}) \tag{2.7}$$

## 2.3 Applications of Signed Distance Fields

### 2.3.1 Mesh Extraction

Triangle meshes are one of the most common representations of 3D data in use today. Therefore, it is important that any implicit surface function be easily convertible into a surface mesh.

One of the most common algorithms for constructing meshes is the marching cubes algorithm, which takes in a voxel grid of occupancy values and extracts 8 point cubes around each voxel. It then maps each 8-point occupancy pattern to a known mesh primitive, and fuses the primitives together into a mesh. A visualization of the geometry primitives and a reconstructed mesh can be seen in **Figure 2.3**.


**(a)** Marching Cubes Primitives

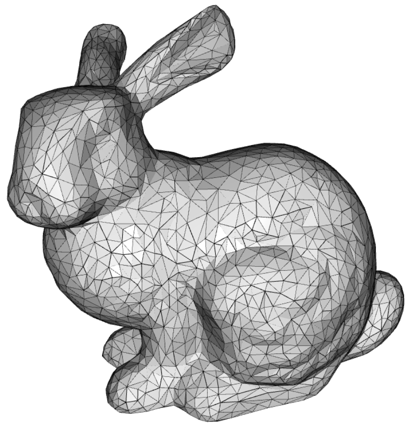

**(b)** Reconstructed Mesh

**Figure 2.3.** (a) The mesh primitives used in the marching cubes algorithm. (b) An example of a triangle mesh of the Stanford Bunny, constructed by the marching cubes algorithm.

### 2.3.2 Rendering Signed Distance Fields

One of the chief benefits of a signed distance field representation over an occupancy field is the ability to use ray marching to quickly render. Ray marching involves querying the signed distance value at a particular point on a ray of interest, jumping along the ray by that quantity, and repeating the process until the signed distance value falls within some threshold $\varepsilon$. The ray marching algorithm can be seen in **Algorithm** 1 and is visually depicted in **Figure 2.4**

**Algorithm 1.** Ray marching

```
1: Given a ray denoted by point $\mathbf{p} \in \mathbb{R}^3$ and unit-norm viewing direction $\mathbf{v} \in \mathbb{R}^3$, signed distance
   function $f : \mathbb{R}^3 \to \mathbb{R}$ and depth threshold $\varepsilon$
2: $\mathbf{p}_{cur} \leftarrow p$
3: while $f(\mathbf{p}_{cur}) < \varepsilon$ do
4:     $\mathbf{p}_{cur} \leftarrow f(\mathbf{p}_{cur}) * \mathbf{v}$
5: end while
6: $d_{\mathbf{p},\mathbf{v}} = ||\mathbf{p} - \mathbf{p_{cur}}||_2$
```

### 2.3.3 Motion Planning

Signed distance fields are useful representations for motion planning, as they provide constant-time collision checking as well as allow for the construction of a maximum clearance

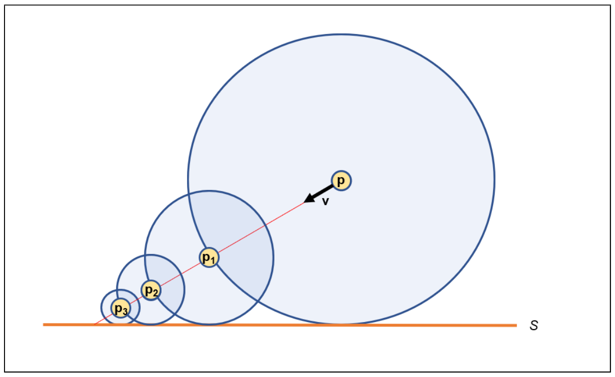


**Figure 2.4.** A visualization of the ray marching algorithm. Starting at point **p**, the ray along direction **v** is rendered by iteratively jumping along **v** by the signed distance of **p** to get to a new point $\mathbf{p}_1$ and continuing until the signed distance is within a prespecified tolerance.

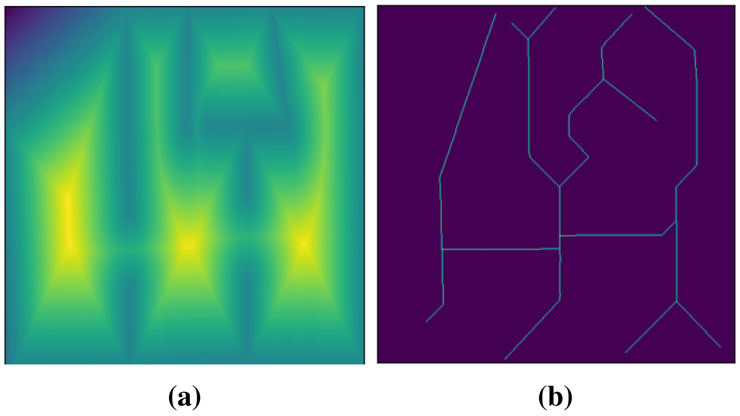


**Figure 2.5.** (a) The signed distance field of an example scene from the HouseExpo dataset with multiple rooms. (b) A maximum clearance roadmap extracted from the signed distance field, which corresponds to the cut locus of the field.

roadmap of obstacles. This roadmap will coincide with the cut locus of the distance field, or the set of points whose nearest surface points are non-unique.

# Chapter 3

# Problem Formulation

## 3.1 Surface Reconstruction from Posed Depth Images

Given a trajectory of depth images with associated poses $\{(T_i, D_i)\}_{1:T}$, we are interested in estimating estimating the signed distance function defining the surfaces observed in this trajectory. Specifically, let us define $X_i[u,v]$ as the world-frame coordinate observed by the depth image $D_i$ at pixel coordinates $(u,v)$ from pose $T_i$.

$$X_i[u,v] = \frac{1}{D_i[u,v]} T_i K^{-1} \begin{bmatrix} u & v & 1 \end{bmatrix}^T \tag{3.1}$$

where $K$ denotes the intrinsic matrix of the camera. We assume no noise and that all points are samples of the ground truth surface $S \subset \mathbb{R}^n$ i.e. $X \subset S$. Our problem can then be defined as learning the parametrization of an implicit function $f$ such that it satisfies the constraints of a signed distance function:

$$f_\theta(\mathbf{x}) = 0, \; \mathbf{x} \in X \tag{3.2}$$

$$|\nabla_{\mathbf{x}} f_\theta(\mathbf{x})| = 1, \mathbf{x} \in \mathbb{R}^n \tag{3.3}$$

$$\nabla_{\mathbf{x}} f_\theta(\mathbf{x}) = N(\mathbf{x}), \; \mathbf{x} \in X \tag{3.4}$$

## 3.2 Incremental Mapping

Additionally, we consider an extension to the above formulation where we no longer have a predefined domain, and are receiving new posed depth data and need to update our function parameters online, while avoiding forgetting prior information. Specifically, given prior parameters $\theta$ and new posed depth $(T_{T+1}, D_{T+1})$ we want to find $\theta'$ such that.

$$f_{\theta'}(\mathbf{x}) = 0,\ \mathbf{x} \in X_{1:T+1} \tag{3.5}$$

$$|\nabla_{\mathbf{x}} f_{\theta'}(\mathbf{x})| = 1, \mathbf{x} \in \mathbb{R}^n \tag{3.6}$$

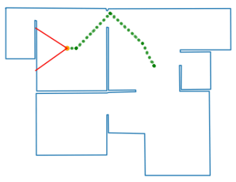
$$\nabla_{\mathbf{x}} f_{\theta'}(\mathbf{x}) = N(\mathbf{x}),\ \mathbf{x} \in X_{T+1} \tag{3.7}$$

An additional objective is to minimize computation in performing this parameter update so as to maximize performance.

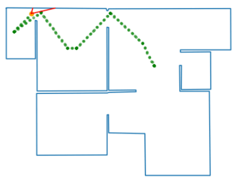
## 3.3 Datasets

We test our method on datasets both in 2D and 3D. In 2D we use the houseexpo dataset, which is a large scale dataset of indoor layouts [18]. Trajectories were handcrafted to sufficiently explore the entire environment. We perform qualitative and quantitative evaluations on 4 scenes with varying trajectory lengths and scene complexities. Frames from a rendered trajectory can be seen in **Figure 3.1**

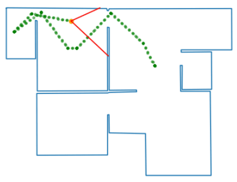
In 3D we use Replica-CAD Apt2, an artistic recreation of ones of the apartments from the Replica dataset [15].. We use the trajectory generated by Ortiz et al.[9].



**(a)** Room 1

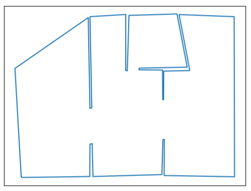


**(b)** Room 2

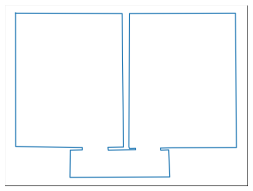


**(c)** Room 3

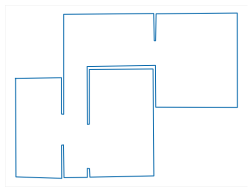
**Figure 3.1.** Visualizations of a lidar scans in a trajectory through room 4 of the HouseExpo dataset. Blue indicates surface walls, orange indicates the current position of the robot, green indicates previous robot positions, and red indicates the viewing cone of the robot.



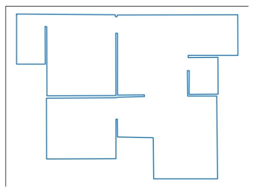
**(a)** Room 1



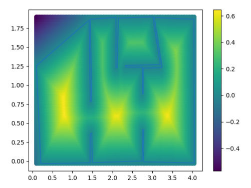
**(b)** Room 2



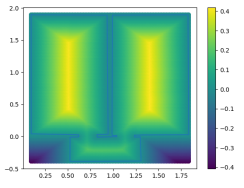
**(c)** Room 3



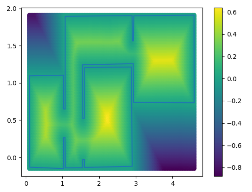
**(d)** Room 4



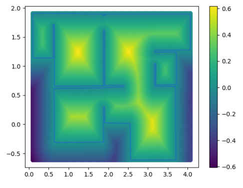
**(e)** Room 1



**(f)** Room 2

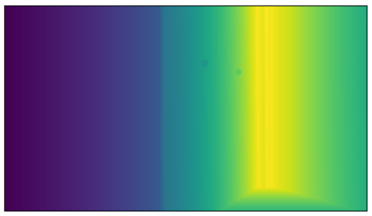


**(g)** Room 3



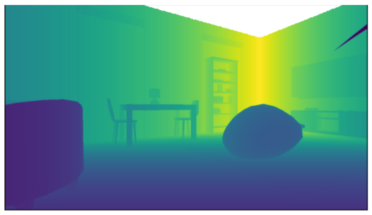
**(h)** Room 4

**Figure 3.2.** The 4 scenes from the HouseExpo dataset being used. (a)-(d) show the ground truth surfaces of the scenes. (e)-(h) show the ground truth SDF maps of each of the scenes

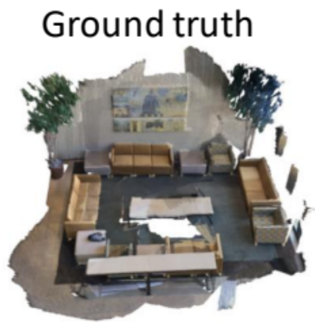


**(a)** t = 0

**(b)** t = 1000



**(c)** t = 2000



**(d)**

**Figure 3.3.** (a)(b)(c) Depth Images from the sampled trajectory of apartment 2 in ReplicaCAD. All images were simulated from the perspective of a ground level robot. d) Isometric View of the apartment.

## 3.4 Metrics

We evaluate our learned implicit function $f_\theta$ using two error metrics, both as presented in [9]. The first is the absolute error between predicted and ground truth signed distance values:

$$e_{sdf}(\mathbf{x}) = |f_{true}(\mathbf{x}) - \hat{f}(\mathbf{x})| \tag{3.8}$$

The second metric is the cosine distance between the gradients of the predicted and ground truth signed distance fields:

$$e_{grad}(\mathbf{x}) = 1 - \frac{\nabla_{\mathbf{x}}\hat{f}(\mathbf{x}) \cdot \nabla_{\mathbf{x}} f_{true}(\mathbf{x})}{||\nabla_{\mathbf{x}}\hat{f}(\mathbf{x})||||\nabla_{\mathbf{x}} f_{true}(\mathbf{x})||} \tag{3.9}$$

# Chapter 4

# Methodology

## 4.1 Model Architecture

### 4.1.1 Multilayer Perceptron

We used a feedforward neural network to process input vectors. For an input vector $\mathbf{v}$, we define the feedfoward process as follows:

$$\mathbf{v}_{i+1} = h(W_i \mathbf{v}_i + b_i) \tag{4.1}$$

where $W_i$ and $b_i$ are the weights and bias of the $i$'th layer, respectively, and $h$ is a non-linear activation function. We use a rectified linear unit (ReLU) activation function, defined as the following:

$$h(x) = \begin{cases} x & x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

The input to the network is an input coordinate $\mathbf{x} \in \mathbb{R}^n$, which outputs a scalar value $z \in \mathbb{R}$. For a network with $L$ layers, we define our parameters $\theta_{MLP}$ as the set of weights and biases $\{(W_i, b_i)\}_{i \in L}$. A graphical depiction of the MLP can be seen in **Figure 4.1**.

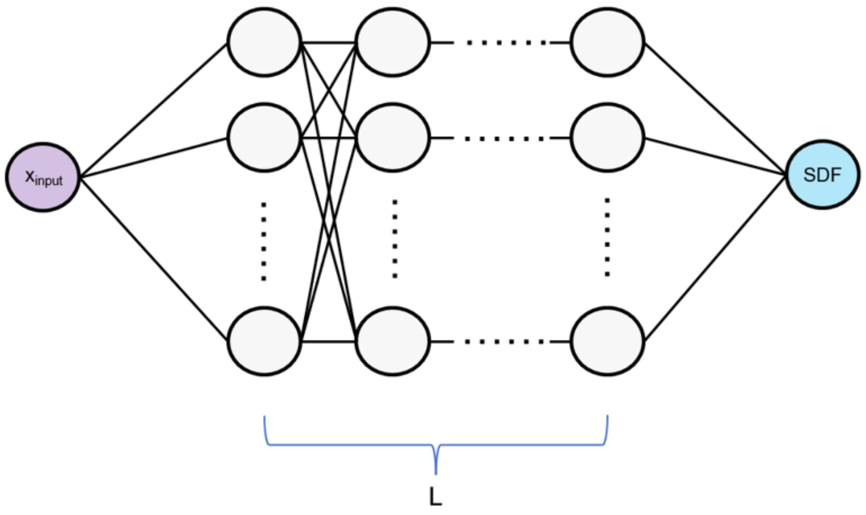


**Figure 4.1.** Depiction of coordinate-based multilayer perceptron. $x_{input} \in \mathbb{R}^3$, $SDF \in \mathbb{R}$

### 4.1.2 High-Frequency Embedding

Neural networks are inherently biased towards learning lower-frequency functions. For this reason, it is common to witness the loss of higher-frequency detail when representing surface information. We explore the use of three variations of high-frequency positional encodings found in the literature, referred to as ”on-axis”, ”off-axis” and ”random” embeddings.

The on-axis embedding was introduced by Barron et. al. [7] and transforms an input coordinate $\mathbf{x} = (x, y, z)$ as follows:

$$\begin{aligned} \eta(\mathbf{x}) = &[\sin(2^i \pi x),\ \cos(2^i \pi x)]_{0:L-1} \frown \\ &[\sin(2^i \pi y,\ \cos(2^i \pi y))]_{0:L-1} \frown \\ &[\sin(2^i \pi z),\ \cos(2^i \pi z)]_{0:L-1} \end{aligned} \tag{4.3}$$

where $\frown$ denotes sequential concatenation and L is a hyperparameter. The resulting value $\eta(\mathbf{x}) \in \mathbb{R}^{3L}$ is then passed in as the input to the MLP decoder.

The off-axis embedding is used by Ortiz et. al. [9] and extends on 4.3 by transforming the input coordinate by a matrix $A \in \mathbb{R}^{21\times3}$ containing the outward-facing unit normals of a twice-tessellated icosahedron.

$$\eta(\mathbf{x}) = [\sin(2^i A\mathbf{x}),\ \cos(2^i A\mathbf{x})]_{0:L} \tag{4.4}$$

Tancik et. al. introduce a further extension on 4.3 in the form of random fourier features[17]. For the same input vector $\mathbf{x}$, the random fourier feature encoding is as follows:

$$\eta(\mathbf{x}) = [\sin(2\pi\mathbf{b}_i^T\mathbf{x}),\ \cos(2\pi\mathbf{b}_i^T\mathbf{x})] \tag{4.5}$$

where $\mathbf{b}_j \in \mathbb{R}^{M\times3}$ is randomly sampled from an isotropic distribution. We found that, when $\theta$ was limited to $\theta_{MLP}$ i.e. the model is exclusively an MLP, the random fourier positional encoding outperformed other variants and significantly improved surface reconstruction.

### 4.1.3 Implicit Neural Features

A key component of our approach is the use of implicit neural feature grids. These comprise a rectilinear graph of nodes, each containing a vector $\mathbf{v}_i \in \mathbb{R}^C$ associated with a cartesian coordinate $\mathbf{x_i}$. For a given query point $\mathbf{x}_{query} \in \mathbb{R}^n$, we interpolate a feature vector using bi- or trilinear interpolation. Concretely, in 2D with neighbor points $\mathbf{x}_{11}$, $\mathbf{x}_{12}$, $\mathbf{x}_{21}$ and $\mathbf{x}_{22}$, we compute the $j$'th element of the interpolated feature vector $\mathbf{v}_{query}$ as follows:

$$\mathbf{v}_{query,\,j} = \frac{1}{(x_2-x_1)(y_2-y_1)} \begin{bmatrix} x_2 - x_{query} & x_{query} - x_1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_{11,\,j} & \mathbf{v}_{12,\,j} \\ \mathbf{v}_{21,\,j} & \mathbf{v}_{22,\,j} \end{bmatrix} \begin{bmatrix} y_2 - y_{query} \\ y_{query} - y_1 \end{bmatrix} \tag{4.6}$$

The 3D case is a simple extension of (4.6) with subsequent linear interpolation along the z-dimension. In the case of a prespecified rectangular input domain $\Omega \subset \mathbb{R}$, we define the coordinate indices of the grid through a rectilinear discretization of $\Omega$ as defined by lower and upper bounds $\mathbf{x}_{lower}$ and $\mathbf{x}_{upper}$, respectively. Given a defined feature grid resolution $\mathbf{r} \in \mathbb{I}^n_+$, our feature grid parameters $\theta_{grid}$ can then be defined in the 2D case as the following set of vectors

$$R_x = \{x_{lower},\, x_{lower} + \frac{1}{\mathbf{r}_x}(x_{upper} - x_{lower}),\, x_{lower} + \frac{2}{\mathbf{r}_x}(x_{upper} - x_{lower}),\; \dots,\, x_{upper}\} \tag{4.7}$$

$$R_y = \{y_{lower},\, y_{lower} + \frac{1}{\mathbf{r}_y}(y_{upper} - y_{lower}),\, y_{lower} + \frac{2}{\mathbf{r}_y}(y_{upper} - y_{lower}),\; \dots,\, y_{upper}\} \tag{4.8}$$

$$\theta_{grid} = \bigcup_{y \in R_y} \bigcup_{x \in R_x} \mathbf{v}_{x,y} \tag{4.9}$$

Let us define a function $\gamma_{\theta_{grid}} : \mathbb{R}^N \rightarrow \mathbb{R}^C$ which takes in a query point and outputs the associated interpolated feature vector. Similarly, let us define a function $\phi_{\theta_{MLP}} : \mathbb{R}^{N+C} \rightarrow \mathbb{R}$ which takes in the concatenated vector $[\mathbf{v}_{query}, \mathbf{x}_{query}]$ and outputs a predicted signed distance value. The final forward pass of the model can be defined as:

$$f_\theta(\mathbf{x}_{query}) = \phi_{\theta_{MLP}}(\mathbf{x}_{query},\, \gamma_{\theta_{grid}}(\mathbf{x}_{query})) \tag{4.10}$$

with final parameters $\theta$ defined as:

$$\theta = (\theta_{grid}, \theta_{MLP}) \tag{4.11}$$

A depiction of the full prediction pipeline with the feature grid is show in Figure 4.2. We randomly initialize our feature grid vectors in accordance with the policy used in [22].

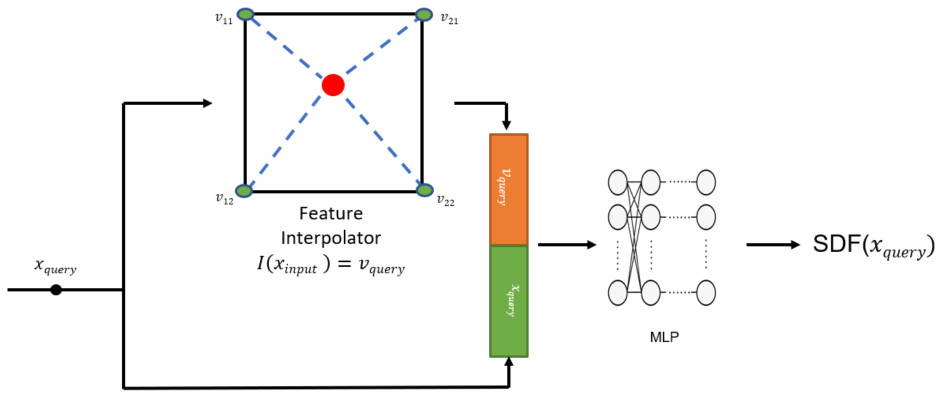


**Figure 4.2.** Depiction of model pipeline with the feature grid. $x_{query} \in \mathbb{R}^2$, $v_{query} \in \mathbb{R}^C$, and $SDF \in \mathbb{R}$

### 4.1.4 Incremental Extension of the Feature Grid

In the incremental learning case, we likely will not know the extants of our input domain ahead of time. Accordingly, we need to be able to grow our feature grid incrementally. In order to accomplish this, we define the grid as a hash-table mapping cartesian coordinates along the rectilinear grid to their respective feature vectors. At inference time, given a cartesian coordinate $\mathbf{x}_{query}$ with neighbors $\mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{21}$ and $\mathbf{x}_{22}$, we lookup each of the neighbors in our hash-table and add any coordinates with randomly initialized vectors that aren't already present. A visualization of the feature grid being expanded as a result of this process can be seen in **Figure 4.3**.

## 4.2 Loss Formulation

We use three components in constructing our loss function, each designed to enforce the fundamental properties of the signed distance function.

$$L(X,\theta) = L_{L1}(X,\theta) + \lambda_{eik} L_{eik}(X,\theta) + \lambda_{normals} L_{normals}(X,\theta) \tag{4.12}$$

where $\lambda_{eik}$ and $\lambda_{normals}$ are hyperparameters.

The first term, $L_{L1}$ is a simple L1 loss regularizing the network prediction at the surface.

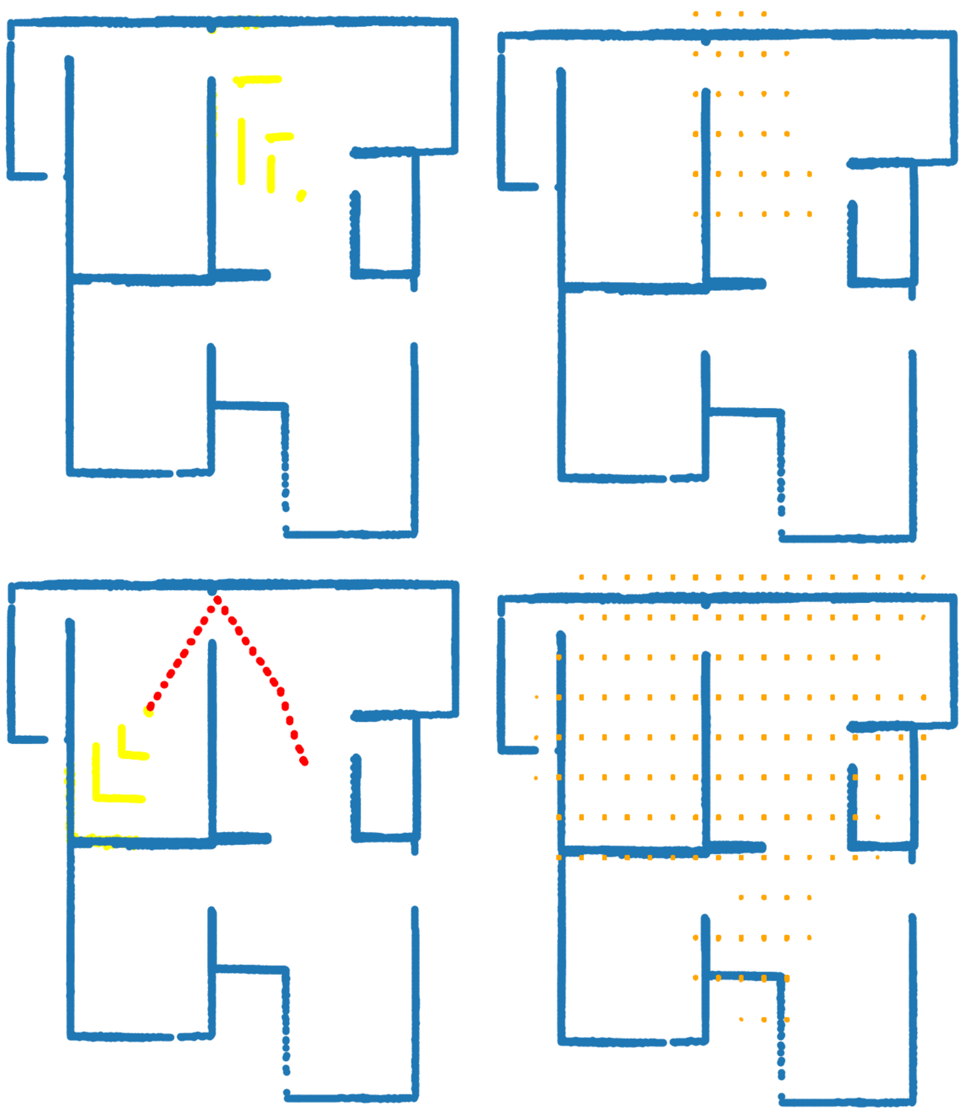


**Figure 4.3.** A visualization of of the incremental expansion of the feature grid. From top to bottom, an initial and subsequent state in a trajectory are shown. Red dots indicate previous robot positions, yellow dots indicate points in the current field of view, and orange dots indicate feature grid coordinates.

Since the implicit function is valued at zero at the surface, the term reduces to the absolute value of the network prediction.

$$L_{L1}(X,\theta) = \frac{1}{N}\sum_{i\in N} |f_\theta(\mathbf{x}_i)| \tag{4.13}$$

The second term is designed to encourage the network gradient to conform to the eikonal constraint at the surface.

$$L_{eik}(\theta) = \frac{1}{N}\sum_{i\in N} (||\nabla_{\mathbf{x}} f_\theta(\mathbf{x}_i)|| - 1)^2 \tag{4.14}$$

The third term is only added when surface normal information is available and encourages the normals of the network at the surface points to align with the gradient of the network at those points.

$$L_{normals}(\theta) = \frac{1}{N}\sum_{i\in N} ||\nabla_{\mathbf{x}} f_\theta(\mathbf{x}_i) - \mathbf{n}_i|| \tag{4.15}$$

where $\mathbf{n}_i$ is the unit normal vector of the surface at point $x_i$. We include the normal term for completeness, but for all experiments presented in this work, the normal term was left off.

## 4.3 Data Augmentation

In order to estimate the value of the signed distance field at points other than the surfaces, we implement a sampling and labeling procedure done in real time. While this is not necessary, as it has been demonstrated that a network can learn a valid signed distance field directly from a point cloud with annotated surface normals, this approach has been demonstrated to improve the quality and speed of the reconstruction, particular at points far away from the surface [4], [9].

The procedure is done in two parts: sampling and label estimation. The sampling procedure involves sampling along the rays of the depth image and generating a set of points and the associated set of signs, and the label estimation involves estimating the euclidean distance value from the set of surface points.

### 4.3.1 Sampling

Concretely, for a given depth image $D_i$ with associated surface coordinates $X_i$, we perform the following sampling operations:

First we select K random pixel coordinates and collect the unit vectors and depths associated with those pixel coordinates. For a given pixel k with associated depth measurement $d_{i,k}$, we generate $N$ stratified depth measurements in the range $[d_{min}, d_{i,k} + d_{delta}]$ where $d_{min}$ and $d_{delta}$ are tunable hyperparameters that specify the sampling bounds with respect to the surface measurement. We define these stratified points initially as offsets from the original depth measurement. Let us refer to these depth offsets as $\Delta d$. As a result we have an augmented set of data samples $D_{i,k,strat}$:

$$D_{i,k,strat} = \{d_{i,k} + \Delta d_j\},\ j \in \{1,\ ...,\ N\} \tag{4.16}$$

For a given sample $d_{i,k,j}$, the sign of the associated signed distance value can be approximated as:

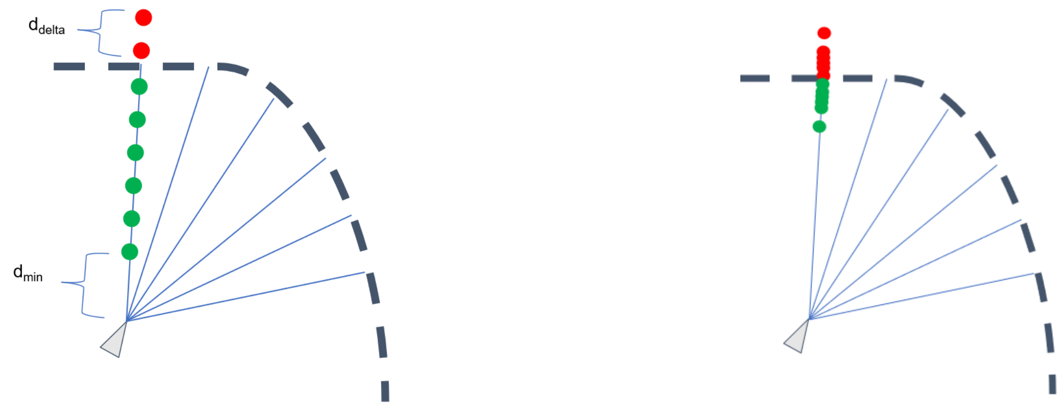
$$sign(f_{estimated}(d_{i,k,j})) = \begin{cases} +1 & \Delta d_j < 0 \\ -1 & \Delta d_j > 0 \end{cases} \tag{4.17}$$

The intention of the stratified sampling is to provide points sampled far from the surfaces.

The second data augmentation involves sampling M deviations from a 0-mean gaussian distribution.

$$D_{i,k,gauss} = \{d_{i,k} + \Delta d_j\},\ j \in \{1,\ ...,\ M\},\ \Delta d_j \sim \mathcal{N}(0, \sigma^2) \tag{4.18}$$

where $\sigma$ is a tunable hyperparameter. The associated label signs are computed according to (4.17). The intention of the gaussian sampling is to provide additional points near the surface



**(a)** Stratified **(b)** Gaussian

**Figure 4.4.** Depictions of stratified (a) and gaussian (b) sampling along a depth ray measurement in 2D. Green points signify a positive estimated signed distance value and red points signify a negative signed distance value.

to improve surface reconstruction and help the network achieve accurate gradient direction information near surface. A depiction of the two sampling methodologies can be seen in **Figure**

The two augmented datasets are then batched together, along with the original estimated surface points, to produce the final dataset:

$$D_{i,k,aug} = D_{i,k} \cup D_{i,k,strat} \cup D_{i,k,gauss} \tag{4.19}$$

The associated world frame set of points $X_{i,k,aug}$ is computed for all augmented distance values as described in equation (3.1). The set of associated signs of each point in $X_{i,k,aug}$ are referred to as $S_{i,k,aug}$. These two sets are then batched along all $K$ pixels to produce the final augmented datasets.

$$X_{i,aug} = \bigcup_{K} X_{i,k,aug} \tag{4.20}$$

$$S_{i,aug} = \bigcup_{K} S_{i,k,aug} \tag{4.21}$$

It is important to note that $S_{i,aug}$ is an approximation, as while all positive signs are known to be correct (since any distance value ranging from 0 to $d_{i,k}$ will be definition be in the free space), the negative values may be incorrect (in the case where the extended ray goes through a thin surface and into the free space on the other side). Therefore, to avoid this incorrect estimation, it is important to choose a sufficiently small $d_{delta}$.

### 4.3.2 Label Estimation

To produce our signed distance approximations for the augmented points $X_{i,aug}$ we estimate the euclidean distance values as follows:

$$EDF(x_i) \approx \min_{\mathbf{x} \in X_i} ||\mathbf{x_i} - \mathbf{x}|| \tag{4.22}$$

The final label then becomes the product of the estimated euclidean distance with the associated sign for each point:

$$f_{estimated}(x_i) = s_i * EDF(x_i),\ s_i \in S_{i,aug} \tag{4.23}$$

We then produce a final labeled dataset $\mathscr{D}$ as the following:

$$\mathscr{D} = \{(\mathbf{x}_i, f_{estimated}(\mathbf{x_i})\} \tag{4.24}$$

### 4.3.3 Augmented Data Loss

We define an additional loss term $l_{aug}$ for the augmented data.

$$L_{aug}(X, f_{estimated}(X), \theta) = \frac{1}{N} \sum_{i \in N} |f_\theta(\mathbf{x}_i) - f_{estimated}(\mathbf{x}_i)| \tag{4.25}$$

For training an augmented dataset $\mathscr{D} = (X, f_{estimated}(X)$ and $X = X_{surf} \cup X_{aug}$, we have the following complete loss formulation:

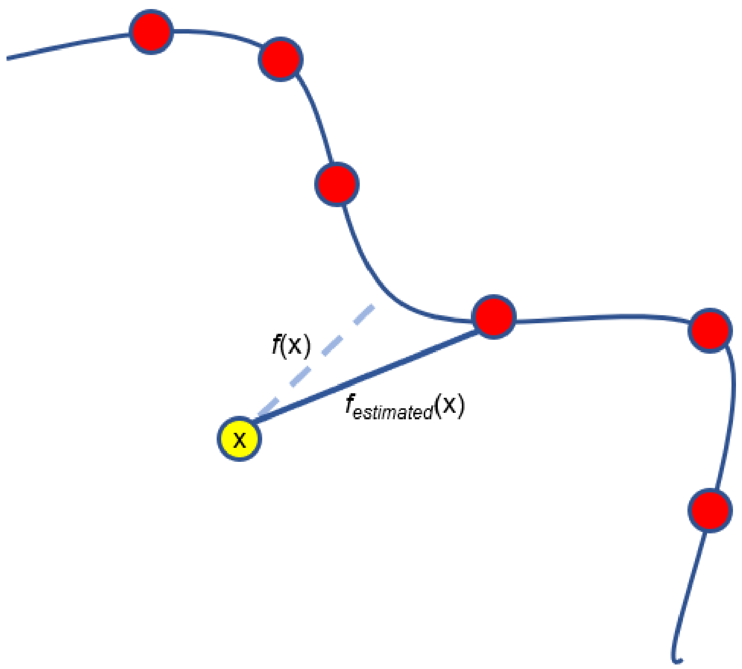


**Figure 4.5.** Batch estimation of the signed distance value of a query point $x$. Red signifies projected lidar points, $f(x)$ is the true vector denoting the signed distance value of $x$ and $f_{estiamted}(x)$ is the vector denoting the estimated signed distance value of x.

$$\begin{aligned} L(X=(X_{surf},X_{aug}),f_{estimated}(X),\theta) = {} & L_{aug}(X_{aug},f_{estimated}(X_{aug}),\theta) \\ & + L_{L1}(X_{surf},\theta) \\ & + \lambda_{eik}L_{eik}(X,\theta) \\ & + \lambda_{normals}L_{normals}(X,\theta) \end{aligned} \tag{4.26}$$

## 4.4 Batch Reconstruction

In batch reconstruction, our aim is to learn the parameter set $\theta = (\theta_{grid}, \theta_{MLP})$ sets that minimizes the loss $L$ with respect to a given dataset $\mathscr{D} = (X, f_{estimated}(X))$ comprising ground truth and augmented data collected from a trajectory.

$$\min_{\theta} \ L(\mathscr{D}, \theta) \tag{4.27}$$

### 4.4.1 Joint Optimization

The first step in the batch reconstruction process is to jointly optimize both feature grid and MLP parameters. Concretely we solve the following minimization problem using gradient descent on the neural network parameters:

$$\min_{\theta_{grid},\theta_{MLP}} \ L(\mathscr{D}, \theta) \tag{4.28}$$

### 4.4.2 Feature Grid Finetuning

After joint optimization, the next step in batch reconstruction is to finetune the feature grid with respect to the learned MLP parameters $\hat{\theta}_{MLP}$

$$\min_{\theta_{grid}} \ L(\mathscr{D}, \theta \mid \theta = (\hat{\theta}_{grid}, \hat{\theta}_{MLP})) \tag{4.29}$$

The rationale behind the process is that during the joint optimization process, a prior for encoding the features of the SDF in cartesian space is learned in the feature grid along with a prior for decoding these features using the MLP. We then fix the decoding process and exclusively update the feature grid parameters, allowing the process of encoding and decoding to be decoupled and for global updates to be made to the neural feature grid.

All optimization problems are solved using gradient descent, with the gradients being computed using autograd. All implementations are done in PyTorch[11].

Chapter 4 section 4 is, in part, coauthored with Shin, Hojoon. The thesis author was the primary author of this chapter.

## 4.5 Incremental Mapping

Our incremental training process is an extension of the batch reconstruction process. First, we warmup our decoder by performing a joint optimization on an initial dataset of N frames. Then, we generate new datasets from sets of frames and optimize our feature grid parameters with respect to these new datasets. Since the feature grid parameters are localized to geographic regions, this process allows for incremental updates to new regions of space without making global changes that will effect predictions across the input domain. An overview of the process is depicted in **Figure 4.6**.

### 4.5.1 Warmup

For an initial trajectory $\{(D_i, T_i)\}_{initial}$, we select a subset of $N$ keyframes and solve a joint optimization problem analogous to Equation (4.28).

$$\min_{\theta_{grid}, \theta_{MLP}} \; L(\mathscr{D}_{keyframe}, \theta) \;\; , \mathscr{D}_{keyframe} \subset \mathscr{D}_{initial} \tag{4.30}$$

where $\mathscr{D}_{initial}$ is the augmented SDF dataset computed from $\{(D_i, T_i)\}_{initial}$. From this joint optimization, we obtain a prior $\hat{\theta}_{MLP}$.

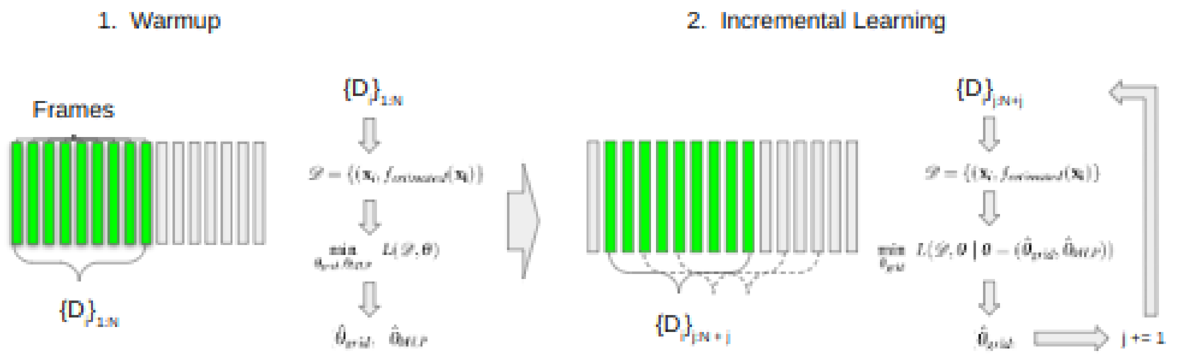

1. Warmup
Frames
$\{D\}_{1:N}$
$\{D\}_{1:N}$
$\mathscr{D} = \{(\mathbf{x}_i, f_{estimated}(\mathbf{x}_i))\}$
$\min_{\theta_{grid}, \theta_{MLP}} L(\mathscr{D}, \theta)$
$\hat{\theta}_{grid}, \hat{\theta}_{MLP}$
2. Incremental Learning
$\{D\}_{j:N+j}$
$\{D\}_{j:N+j}$
$\mathscr{D} = \{(\mathbf{x}_i, f_{estimated}(\mathbf{x}_i))\}$
$\min_{\theta_{grid}} L(\mathscr{D}, \theta \mid \theta = (\hat{\theta}_{grid}, \hat{\theta}_{MLP}))$
$\hat{\theta}_{grid}$
j += 1

**Figure 4.6.** A depiction of our incremental training approach. On the left is the full end-to-end training pipeline for the warmup. On the right is the incremental approach.

During the warmup period, we train the network for more epochs so as to learn a richer prior for the decoder. During the incremental period, we train for fewer epochs so as to allow the network to run quickly and in real-time.

### 4.5.2 Sliding Window

Let us consider a set of M new observations in the trajectory $\{(D_j, T_j)\}_{j \in M}$ and associated SDF dataset $\mathscr{D}_{new}$. We freeze the weights of the decoder MLP and then solve the following optimization problem.

$$\min_{\theta_{grid}} \; L(\mathscr{D}_{new}, \theta \mid \theta = (\hat{\theta}_{grid}, \hat{\theta}_{MLP})) \tag{4.31}$$

In selecting the set of M observations, we take a stratified sliding window approach over the set of frames in the trajectory. Concretely, we define hyperparameters $w$ and $e$, where $e$ denotes the temporal range of the sliding window and $e$ denotes the interval at which frames are sparsely sampled over that range. For sampling at time $t$, We sample the set of observations observations corresponding to indices $\{j-w,\ j-w+e, j-w+2e,\ ...,\ j\}$. For example, with $j = 200$, $w = 100$ and $e = 50$, the set of frames would include indices $\{100, 110, 120, ..., 200\}$.

This approach allows the network to train on data multiple times, as well as allowing for broader coverage of scene geometry when estimating the labels of the dataset, leading to more accurate reconstructions.

# Chapter 5

# Results

## 5.1 Evaluation of Model Architectures

We evaluate the effectiveness of different model architectures in representing 3D signed distance fields on the object level using ground truth SDF data from the SDFExplorer dataset[16]. Mesh reconstructions were all performed using the marching cubes algorithm.

### 5.1.1 Activation

Experiments on the efficacy of different activation functions were performed on the SDF explorer dataset. Experiments were run comparing sinusoidal, softplus, tanh and ReLU activation layers. All models were trained using an MLP with 8 layers and 256 hidden units, trained on 20 surfaces, each with 300K ground truth labeled datapoints. Training was done using the ADAM optimizer with a decaying learning rate over 120 epochs[5]. Objects were normalized to a unit cube and evaluations were performed on a $200^3$ voxel grid using an L1 loss metric. Quantitative results can be seen in **Table 5.1**

**Table 5.1.** Quantitative analyses of reconstruction accuracy of MLP networks using different activation layers.

| | sin | softplus | tanh | relu |
|---|---|---|---|---|
| L1 Loss | 0.163 | 0.000921 | 0.00265 | **0.000904** |

**Table 5.2.** Quantitative analyses of reconstruction accuracy of MLP networks using different embedding strategies. All networks were trained using ReLU activations.

| | No Embedding | Fourier Features | Off-Axis | On-Axis |
|---|---|---|---|---|
| L1 Loss | 0.00102 | 0.000656 | **0.000424** | 0.000625 |

### 5.1.2 Embedding Strategies

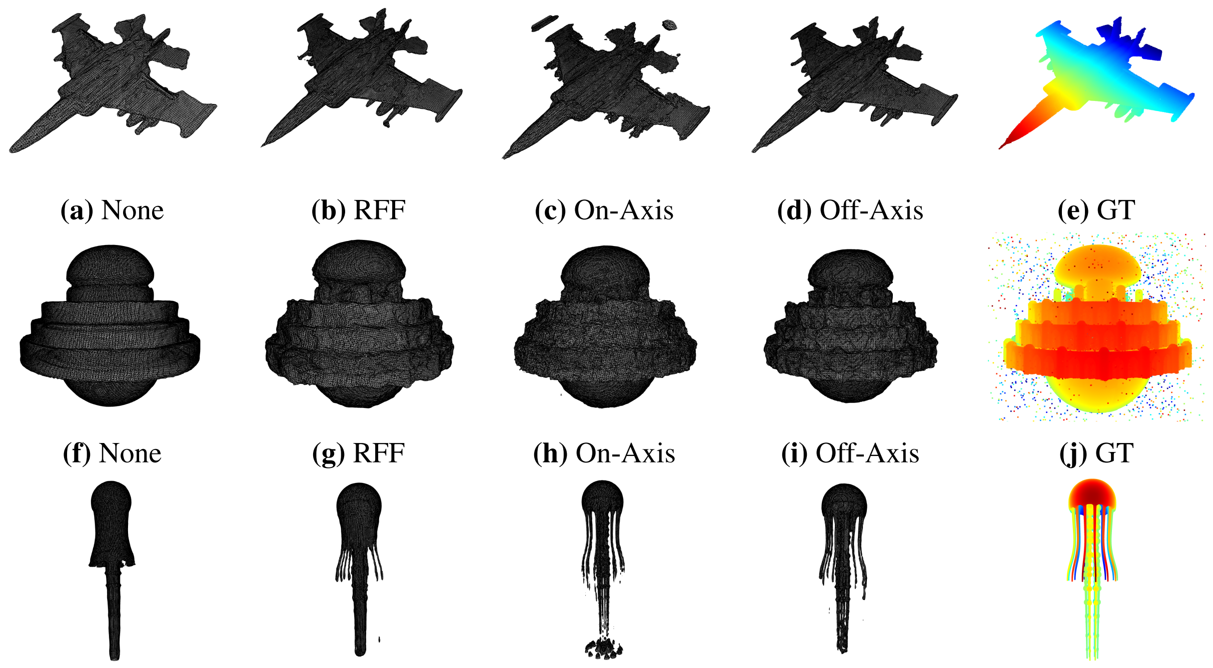
An analysis of different embedding strategies on reconstruction is performed. We analyzed four cases: no positional embedding, the ”on-axis” embedding, the ”off-axis” embedding, and randomly-generated fourier features, with embeddings for the ”on-axis” and ”off-axis” embeddings being defined with hyperparameters $L = 5$ and $L = 10$ and with the random fourier features using parameter $M = 50$. All embeddings were trained using an MLP with 8 layers and 256 hidden units and ReLU activation layers on 20 surfaces, each with 300K ground truth labeled datapoints from the SDFExplorer dataset[16]. Training was done using the ADAM optimizer with a decaying learning rate over 120 epochs[5]. Objects were normalized to a unit cube and evaluations were performed on a $200^3$ voxel grid using an L1 loss metric. Reconstructions were performed using marching cubes and can be seen in **Figure 5.1**. Quantitative results can be seen in **Table 5.2**.

## 5.2 Scene Reconstruction

We demonstrate the effectiveness of our approach for scene surface reconstruction from a trajectory of depth images on both 2D and 3D scene trajectories. We perform quantitative and qualitative evaluations of the full reconstructed signed distance fields.

### 5.2.1 House Expo

2D scenes were trained on batch datasets constructed from prerendered trajectories. SDF sampling for our method and for iSDF baselines was performed using $N_{strat} = 7$ and $N_{gauss} = 2$, with $d_{min} = 3$, $d_{delta} = 0.2$, $\sigma = 0.1$. All 120 rays of the LIDAR scan were used. Parameters were


**(a)** None **(b)** RFF **(c)** On-Axis **(d)** Off-Axis **(e)** GT

**(f)** None **(g)** RFF **(h)** On-Axis **(i)** Off-Axis **(j)** GT

**(k)** None **(l)** RFF **(m)** On-Axis **(n)** Off-Axis **(o)** GT

**Figure 5.1.** Qualitative results of training MLPs with various feature embedding strategies. As can be seen, high-frequency embeddings dramatically increase the fidelity of surface reconstructions. However, they can often lead to overfitting and noisy surfaces.

a 5-layer MLP with ReLU activations and 128 hidden units and a feature grid with individual grid units of dimensions $25 \times 25$ and feature dimension $C = 16$. Raw depth data was on the order of $10^2$, resulting in feature grid dimensions on the order of $16x10x10$. All data was scaled by 0.01 prior to being fed into the network. A value of $\lambda_{eik} = 0.2$ was used for the loss. All parameters were jointly optimized for 50 epochs, and then feature grid finetuning was performed for 50 epochs. Training was done using the ADAM optimizer.

Results for samples taken over 80 sparsely selected frames from the trajectory exploring Room 1 of the HouseExpo dataset can be seen in **Figure 5.2**. The network demonstrates the ability to infer a reasonably accurate signed distance field in areas that do not receive augmented data samples, indicating that the decoder is in fact learning general properties of the signed distance field and not just memorizing data.

The number and distribution of frames selected for training makes a significant impact on the quality of reconstruction, with comparative results being shown in **Figure 5.3**. It can be seen that fewer frames leads to not only a poor final reconstruction but a decrease in the accuracy of the self-supervised labeling, due to the decreased likelihood of capturing the nearest surface point for a given sampled datapoint.

Results for the final reconstruction can be seen in **Figure 5.4**. Trajectories were sampled sparsely in sets of 100 frames from the whole trajectories, with the same sampling and training procedure. Quantitative evaluations comparing results for training with the feature grid vs. without the feature grid can be found in **Table 5.3**. The results show that it is possible to learn high fidelity, accurate signed distance fields using our network and sampling approach, both with and without a feature grid. Our approach demonstrates that we can reconstruct high quality signed distance fields with strong gradient eikonality directly from self-supervised sampling.

### 5.2.2 Incremental Reconstruction

Results for incrementally reconstructing the HouseExpo dataset can be seen in **Figure 5.5**. All model hyperparameters are the same as in batch reconstruction. During the warmup

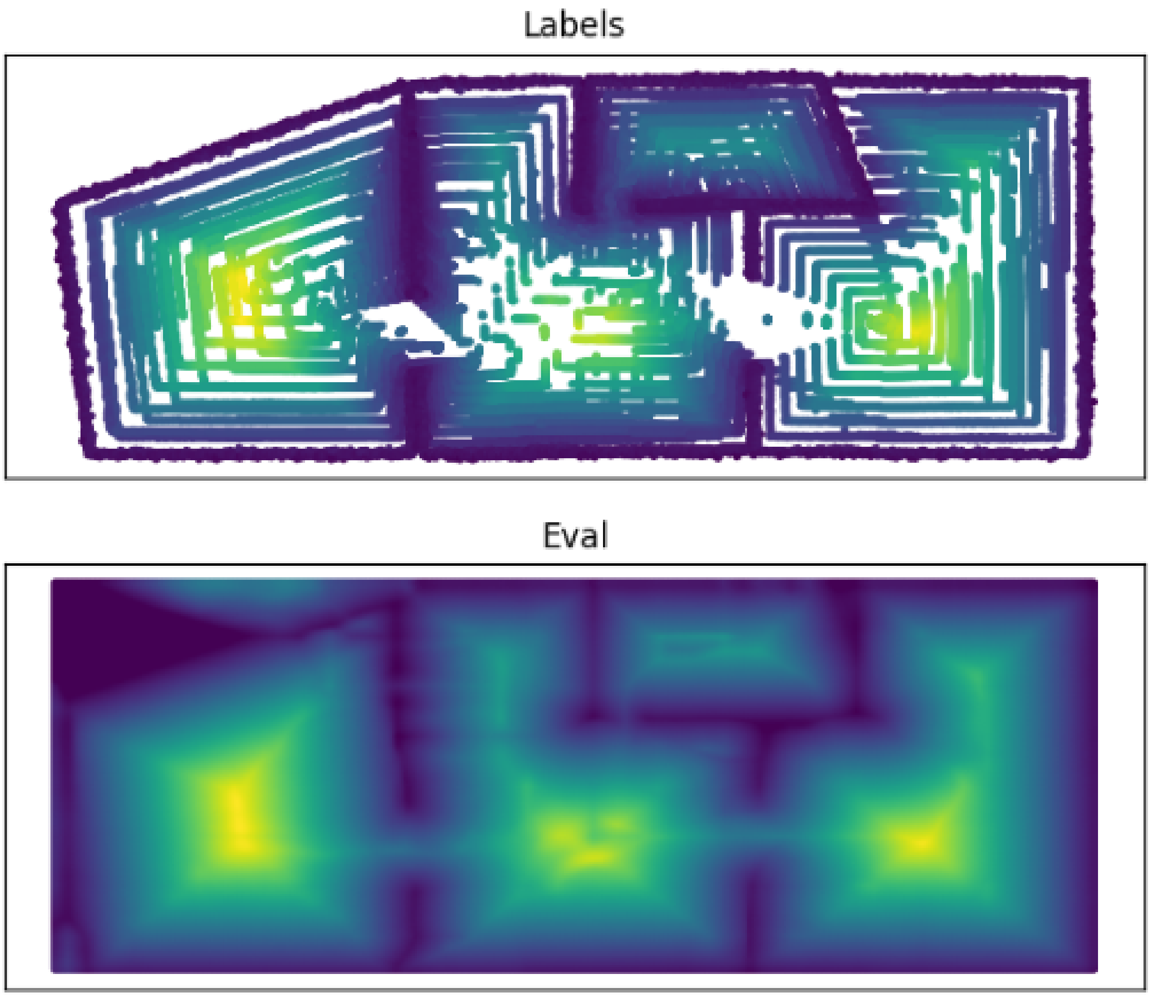
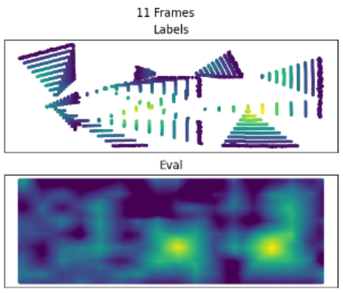


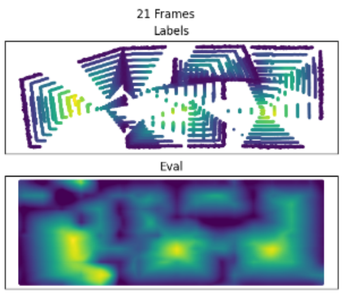
**Figure 5.2.** A visualization of sampling results for HouseExpo room 1 and the resulting trained network on a slice of 80 sparsely sampled frames. The top frame shows the estimated SDF as a result of our data augmentation process and the bottom frame shows the resulting batch reconstruction result.

**Table 5.3.** 2D Batch Reconstruction Error Metrics

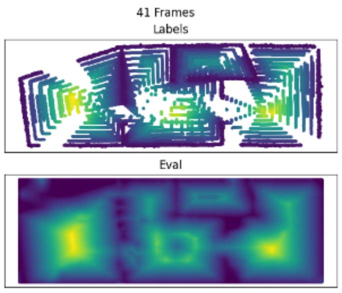
| | MLP | | MLP w Features | |
|---|---|---|---|---|
| | SDF Error | Gradient Error | SDF Error | Gradient Error |
| Room 1 | 0.044 | 0.172 | 0.011 | 0.171 |
| Room 2 | 0.041 | 0.231 | 0.031 | 0.240 |
| Room 3 | 0.093 | 0.228 | 0.123 | 0.211 |
| Room 4 | 0.068 | 0.240 | 0.093 | 0.244 |



**(a)**

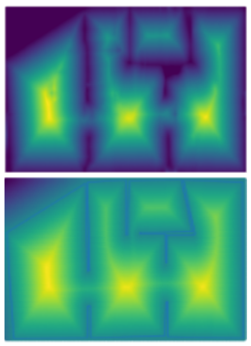


**(b)**

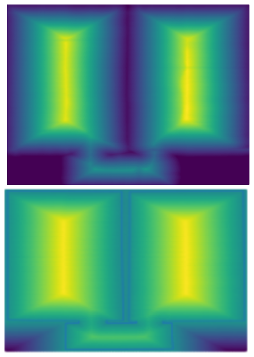


**(c)**

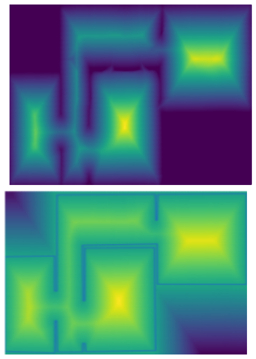
**Figure 5.3.** Demonstrates the effect of window size on the result of the reconstruction. Having not only sufficient coverage of the scene but sufficient surface data to generate an accurate reconstruction is of paramount importance in computing an accurate reconstruction.



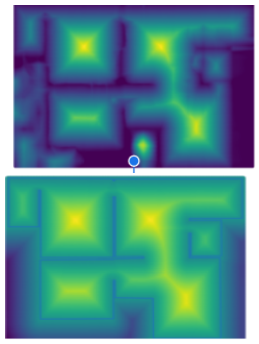
**(a)** Room 1



**(b)** Room 2



**(c)** Room 3



**(d)** Room 4

**Figure 5.4.** Top - predicted reconstructions for HouseExpo room trajectories. Bottom - ground truth signed distance fields for each of the room datasets. Predicted Signed distance field values below -0.1 are set of -0.1 for visualization purposes.

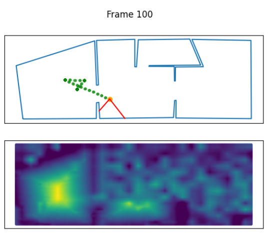
**Table 5.4.** 2D Incremental Reconstruction Error Metrics

| | Room 1 | Room 2 | Room 3 | Room 4 |
|---|---|---|---|---|
| SDF Error | 0.079 | 0.056 | 0.143 | 0.120 |
| Gradient Error | 0.431 | 0.341 | 0.272 | 0.311 |

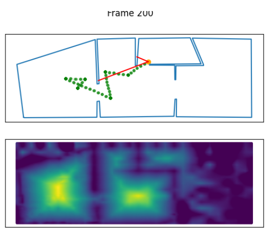
period we train for 400 epochs, and then during each subsequent training we train for 30 epochs. Train time for the incremental portion was approximately 0.5 seconds total per batch, and as a result we were able to process frames at around 20-30 FPS with a sufficiently large $e$ paramter (we chose $e = 20$ and $w = 100$. We demonstrate that we are able to learn reasonably high quality signed distance fields without suffering from the same catastrophic forgetting problems that other online signed distance representations experience.

## 5.3 3D Scene Level Batch Reconstruction

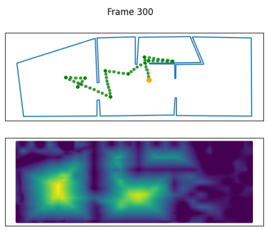
Finally, we demonstrate that our network architecture is capable of extending to 3D and producing high quality mesh reconstructions from trajectories. We sample 150 frames from the trajectory for ReplicaCAD apartment 2 and perform data augmentation with sampling parameters $N_{strat} = 12$ and $N_{gauss} = 3$, with $d_{min} = 7$, $d_{delta} = 3$ and $\sigma = 2$. For each frame, we use surface points downsampled by 5x in both pixel directions, as well as randomly sampling 100 pixels and performing the data augmentation on those 100 pixels. We train models both with and without feature grids. In all cases, we used an 8-layer MLP with 512 hidden units and ReLU activations. In the case of using a positional embedding, we used random fourier features with parameter $M = 50$. With a feature grid, we did not use a positional encoding and used a grid of dimensions 15x15x6 with feature dimension $C = 32$. The MLP models were trained for 600 epochs with learning rate decay, and the feature grid model was trained with learning rate decay for 300 epochs in the joint optimization phase and then again for 300 epochs in the feature grid finetuning phase using the same learning rate schedule. All learning decay schedules used learning rates from $5e-3$ to $1e-4$. Weights were updated using the Adam optimizer.



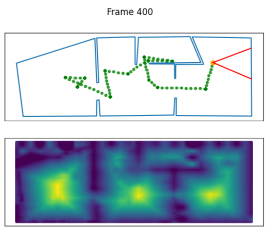
**(a)** 100



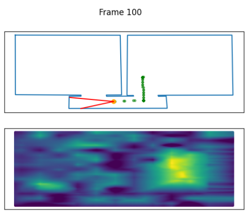
**(b)** 200



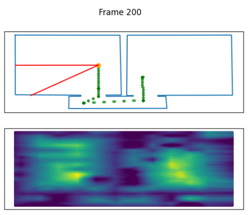
**(c)** 300



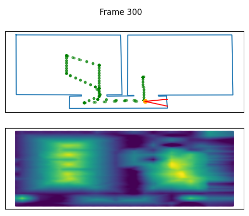
**(d)** 400



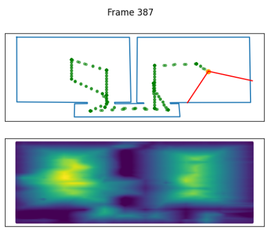
**(e)** 100



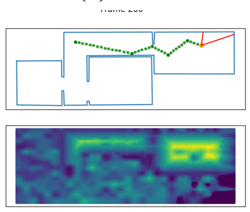
**(f)** 200



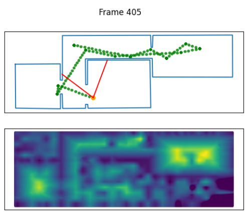
**(g)** 300



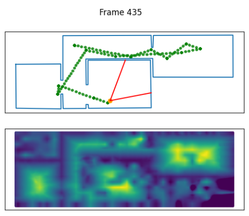
**(h)** 387



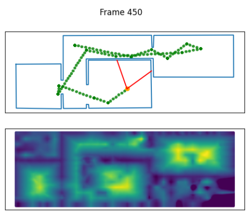
**(i)** 200



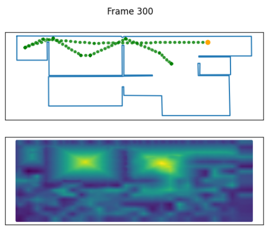
**(j)** 300



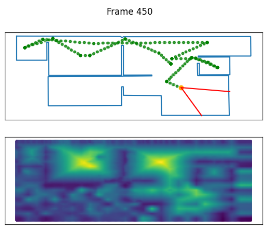
**(k)** 400



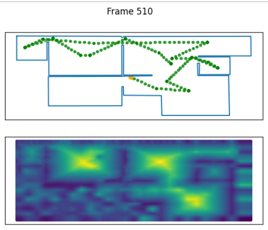
**(l)** 487



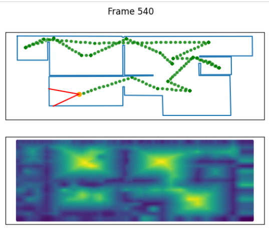
**(m)** 200



**(n)** 300



**(o)** 530



**(p)** 560

**Figure 5.5.** Incremental reconstruction results on the HouseExpo dataset. (a)-(d) are on scene 1, (e)-(h) are on scene 2, (i)-(l) are on scene 3, and (m)-(p) are on scene 4. The top depicts the trajectory and current viewing cone at a given timestamp $t$, and the bottom depicts the predicted SDF at $t$.

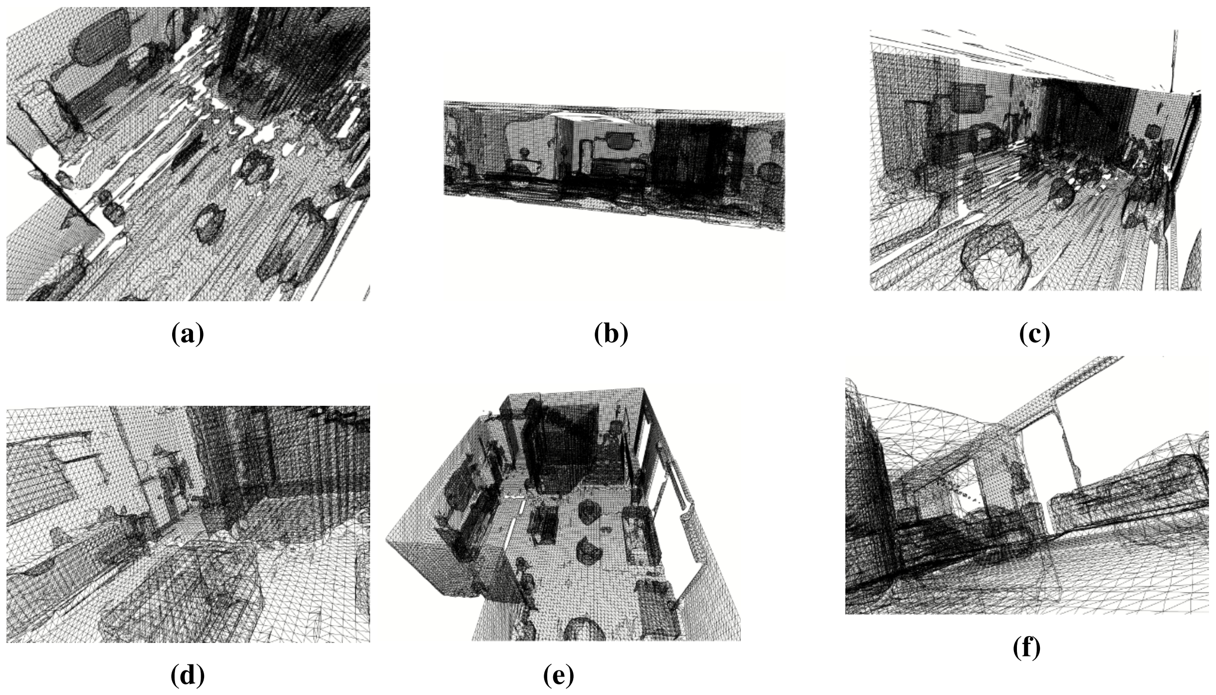


**Figure 5.6.** Visualizations of marching cubes results on batch reconstruction performed using 8-Layer ReLU MLPs. (a)-(c) reconstructions were trained without a positional embedding. (d)-(e) were trained using the same MLP architecture but with random fourier features as an encoding.

Results for the MLP-only models are visualized in **Figure 5.6**. The positional encoding is shown to enhance quality in 3D significantly. However, the best results were obtained by using the feature grid, with almost no extraneous gaps in the mesh and extremely high quality surface detail, as seen in **Figure 5.7**. This demonstrates the representational power of feature grids on a scene level, and that extremely high quality meshes can be reproduced from trajectories without going through the expensive process of differential rendering.

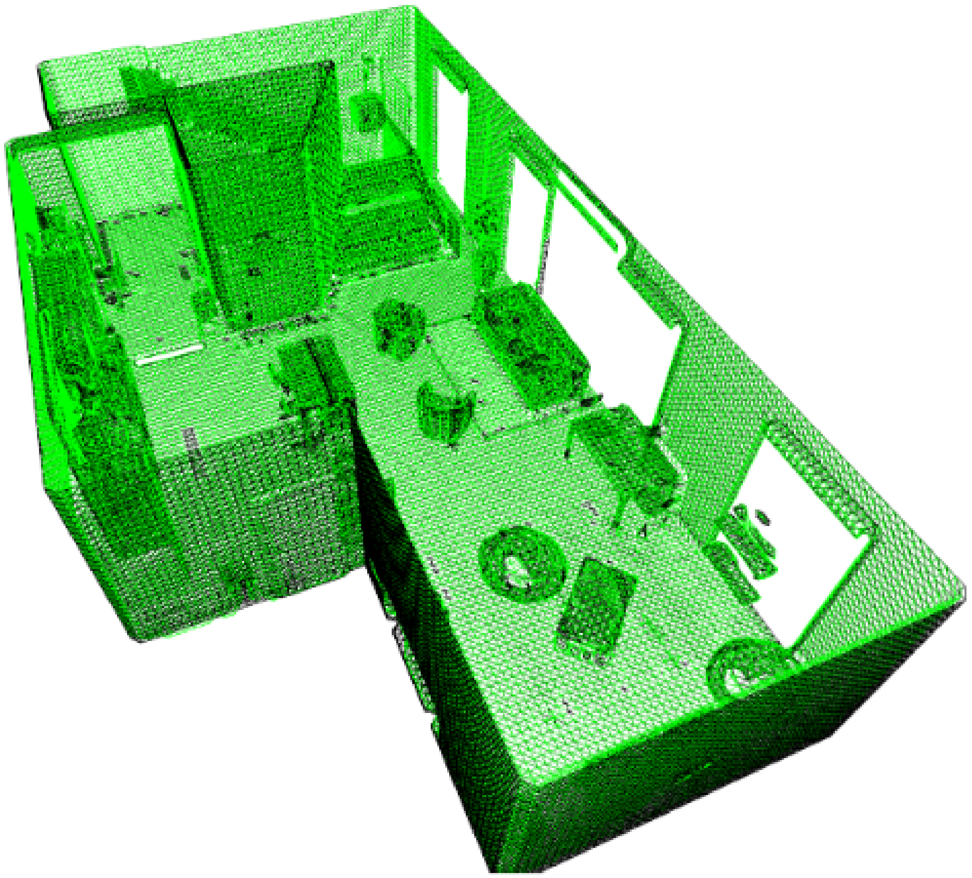


**Figure 5.7.** A visualization of our final result overlayed with the ground truth mesh. Green depicts our prediction and black the ground truth. As can be seen, the training resulted in a precise reconstruction while retaining high-frequency details without the use of a positional encoding.

# Chapter 6

# Future Work

We demonstrate a methodology for incrementally reconstructing signed distance fields from depth observations in a computationally efficient manner without suffering from catastrophic forgetting. However, there are many areas where future work could improve on our methodology.

The rectilinear organization of the feature grid is conducive to linear interpolation, but it is highly inefficient in terms of storage, as feature vectors are often dedicated to regions of complete free space. Future work should explore the use of irregular feature grids and alternative methods of interpolating feature vectors that can handle alternative geometries.

Since our data augmentation approach is constrained to sampling along rays so as to know apriori what the sign of the sampled point will be, it is biased to sample more densely closer to the observer and more sparsely away from the observer in the direction orthogonal to the rays. This is inefficient, and ideally the opposite would be true - as we get closer to the surface of interest, more samples are allocated. Alternative approaches for efficiently estimating the sign of points sampled off of the viewing direction would enable for more deliberate and informative sampling.

A very promising direction of research is learning signed directional distance functions on a scene level. The discontinuities present in such a model make it a difficult challenge for neural networks, which are only guaranteed to be able to represent continuous functions. While a handful of works have made attempts to learn some variant of a directional distance field on the object level [20][23][1], to our knowledge no work has attempted to reconstruct them for entire scenes from observed trajectories.

# Bibliography

[1] Tristan Aumentado-Armstrong, Stavros Tsogkas, Sven Dickinson, and Allan D. Jepson. Representing 3d shapes with probabilistic directed distance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19343–19354, June 2022.

[2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields, 2021.

[3] 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. *CoRR*, abs/2003.10983, 2020.

[4] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of Machine Learning and Systems 2020*, pages 3569–3579. 2020.

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[7] 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 *ECCV*, 2020.

[8] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *CoRR*, abs/2201.05989, 2022.

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

[10] 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] 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 *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[12] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *CoRR*, abs/2006.09661, 2020.

[13] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *CoRR*, abs/1906.01618, 2019.

[14] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew Davison. iMAP: Implicit mapping and positioning in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[15] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[16] Towaki Takikawa, Andrew Glassner, and Morgan McGuire. A dataset and explorer for 3d signed distance functions. *Journal of Computer Graphics Techniques (JCGT)*, 11(2):1–29, April 2022.

[17] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[18] Li Tingguang, Ho Danny, Li Chenming, Zhu Delong, Wang Chaoqun, and Max Q.-H. Meng. Houseexpo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots. *arXiv preprint arXiv:1903.09845*, 2019.

[19] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[20] Tarun Yenamandra, Ayush Tewari, Nan Yang, Florian Bernard, Christian Theobalt, and Daniel Cremers. Fire: Fast inverse rendering using directional and signed distance functions, 2022.

[21] Xiaoshuai Zhang, Sai Bi, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Nerfusion: Fusing radiance fields for large-scale scene reconstruction. *CVPR*, 2022.

[22] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[23] Ehsan Zobeidi and Nikolay Atanasov. A deep signed directional distance function for object shape representation. *CoRR*, abs/2107.11024, 2021.