A Deep Signed Directional Distance Function for Shape Representation

Ehsan Zobeidi Nikolay Atanasov

Abstract— **Predicting accurate observations efficiently from novel views is a key requirement for several robotics applications. Existing shape and surface representations, however, either require expensive ray-tracing operations, e.g., in the case of meshes or signed distance functions (SDFs), or offer only a coarse view, e.g., in the case of quadrics or point clouds. We develop a new representation that captures viewing direction and enables fast novel view synthesis. Our first contribution is a signed directional distance function (SDDF) that extends the SDF definition by measuring distance in a desired viewing direction rather than to the nearest point. As a result, SDDF removes post-processing steps for view synthesis required by SDF, such as surface extraction via marching cubes or rendering via sphere tracing, and allows ray-tracing through a single function call. SDDF also encodes by construction the property that distance decreases linearly along the viewing direction. We show that this enables dimensionality reduction in the function representation and guarantees the prediction accuracy independent of the distance to the surface. Recent advances demonstrate impressive performance of deep neural networks for shape learning, including IGR for SDF, Occupancy Networks for occupancy, AtlasNet for meshes, and NeRF for density. Our second contribution, DeepSDDF, is a deep neural network model for SDDF shape learning. Similar to IGR, we show that DeepSDDF can model whole object categories and interpolate or complete shapes from partial views.**

I. INTRODUCTION

Various representations for modeling object shapes and geometric surfaces have been proposed, including meshes, point clouds, quadrics, occupancy grids, and signed distance functions. Different representations are well suited for different purposes, such as visualization for meshes, odometry for point clouds, or collision checking for occupancy grids. The shape representation challenge in robotics applications, such as manipulation [1] and active perception [2], that involve operation in real-time, require fast ray-tracing for view prediction and occlusion checking. Shape estimation should be performed online using depth-camera or LiDAR observations, and shape models should be trained with such partial-view measurements rather than complete 3D object models. Distance prediction for collision or occlusion checking should be performed highly efficiently and independently of the distance or viewing direction to the observed surface for planning aspects of the problem. However, existing representations require post processing steps, such as surface estimation or sphere tracing, to predict observations for novel views, which may be inefficient.

To address this challenge, we propose a new 3D shape representation, called *signed directional distance function* (SDDF), and a deep neural network model for learning SDDF representations, called *DeepSDDF*. Our SDDF definition extends the popular signed distance function (SDF) [3] by returning the signed distance to a surface along a desired viewing direction, rather than to the nearest point. This makes ray tracing to predict distance with SDDF as efficient as a single function call, or a single neural network forward pass in the case of learned SDDF, and obviates the need for mesh reconstruction [4] or sphere tracing [5].

Additionally, our formulation allows training an SDDF model directly from depth camera or LiDAR data. In designing a learning architecture for SDDF, we observe that the measured distance to a surface decreases at constant unit rate along the viewing direction. Our key contribution is a neural network model for learning SDDFs that encodes this gradient property *by construction* as opposed to IGR that encourages a similar gradient condition though a loss function term. We show that this property allows reducing the dimension of the SDDF model input and makes its training more sample efficient. The gradient property also provides analytical confidence about the SDDF distance prediction accuracy *regardless of the distance to the observed surface* and make it a promising shape representation for planning aspects of the problem. In contrast, deep SDF methods usually are accurate only close to the surface. As a result of these qualities, our method provides a rendering framework that is differentiable by construction, while keeping the most efficient rendering time. Our DeepSDDF model is inspired by IGR [3], which trains a decoder-only network across multiple instances from the same object category and allows generalization to unseen instances in shape completion and shape interpolation tasks. We show that DeepSDDF is able to achieve both *instance-level* and *category-level* object shape reconstruction.

Surface and shape representations can be classified broadly into *explicit* and *implicit*. Explicit models parameterize a surface directly using point cloud [6], polygonal mesh [7], or geometric primitive [8] representations. While geometric primitive and point cloud representations allow convenient visualization, they do not provide a continuous surface representation and, hence, are ineffective for occlusion prediction. Polygonal meshes, on the other hand, offer precise and efficient visualization but learning mesh models from streaming sensor data is challenging because it requires determining a proper number of surface patches [9]. Implicit models like volumetric occupancy [10], density [11]–[14] and SDF [15] models, parameterize surfaces indirectly as a

We gratefully acknowledge support from NSF FRR CAREER 2045945 and ARL DCIST CRA W911NF-17-2-0181.

The authors are with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093, USA, e-mails: zobeidiehsan@gmail.com, natanasov@ucsd.edu.

level set of a spatial function. In these shape representations, view synthesis is not possible through a single pass. Sphere tracing [16] has been employed to enable efficient ray-tracing for SDF models. It reduces the number of SDF function evaluations but still requires at least several evaluations, with the number increasing with the distance from the observed surface. IGR [3] observes that any valid SDF must satisfy a unit-norm constraint on its gradient (Eikonal equation), and introduce an Implicit Geometric Regularization (IGR) term to encourage a trained SDF model to satisfy this constraint. In contrast, DeepSDDF satisfies its the gradient by construction. This makes our DeepSDDF model significantly more sample efficient than IGR and provides analytical confidence about the SDDF prediction accuracy independent of the distance to the surface.

In summary, we make the following contributions.

- We propose a new signed directional distance function for shape representation. Having direction in SDDF allows ray-tracing as a single function evaluation without mesh extraction or sphere tracing (very fast ray tracing independent of distance to the object). It allows training directly from distance data.
- We propose, DeepSDDF, a neural network model to learn SDDF shape representation for instance-level and category-level object surfaces and demonstrate its performance in shape completion and shape interpolation tasks. DeepSDDF satisfies the gradient property by construction, which guarantees the accuracy of SDDF predictions analytically, regardless of the distance to the object surface.

II. PROBLEM STATEMENT

We focus on learning a common shape representation for multiple instances from the same object category, e.g., cars, using distance measurement data, e.g., from a depth camera or a LiDAR scanner. A distance measurement is modeled as a collection of rays, e.g, corresponding to depth camera pixels or LiDAR beams, along which the distance from the sensor position, e.g., depth camera optical center or LiDAR sensor frame origin, is measured to the observed surface. Let $\boldsymbol{\eta}_i$ be a n-dimensional vector on the unit sphere $S^{n-1} := \{\boldsymbol{\eta} \in \mathbb{R}^n \mid \|\boldsymbol{\eta}\|_2 = 1\}$ specifying the direction of the i-th sensor ray. Denote the distance measurement from sensor position $\mathbf{p}_i \in \mathbb{R}^n$ along direction $\boldsymbol{\eta}_i$ by $d_i \in \mathbb{R} \cup \{\infty\}$. In practice, the dimension n is 2 or 3 and distance measurements of rays that do not hit a surface are set to ∞.

Problem 1. Let $\mathcal{D}_l := \left\{(\mathbf{p}_{i,l}, \boldsymbol{\eta}_{i,l}, d_{i,l})\right\}_i$ be distance measurements obtained from different instances l of the same object category. Learn a latent shape encoding $\mathbf{z}_l \in \mathbb{R}^m$ for each instance l and a function $h(\mathbf{p}, \boldsymbol{\eta}, \mathbf{z})$ that can predict the distance from any point $\mathbf{p}$ along any direction $\boldsymbol{\eta}$ to the surface of an instance with shape $\mathbf{z}$ from the same category.

III. SDDF

In this section, we propose a signed directional distance function for implicit shape representation. Sec. III-A defines SDDF and formulates the condition that the distance to a surface decreases linearly along the viewing direction. Sec. III-B and Sec. III-C study the properties of functions that satisfy the SDDF condition.

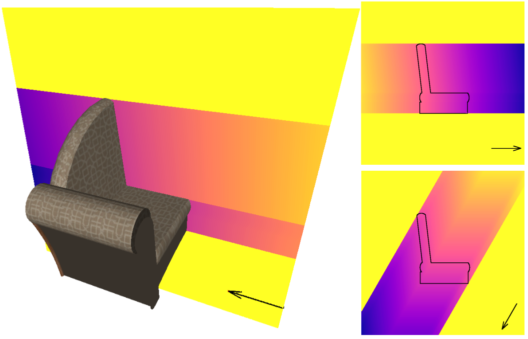

Fig. 1. SDDF of a sofa along three directions indicated by arrows, on the same plane cutting the sofa. The color at each point shows the SDDF value at that point with fixed direction from small (purple) to large (yellow). When the view rays do not intersect the object the distance is infinite (yellow regions).

A. Signed Directional Distance Function

We define a function $h(\mathbf{p}, \boldsymbol{\eta})$ to model the distance measured by a distance sensor from position $\mathbf{p}$ to the surface of a set along viewing direction $\boldsymbol{\eta}$. Both $\mathbf{p}$ and $\boldsymbol{\eta}$ are in the same coordinate frame.

Definition 1. A *signed directional distance function* (SDDF) $h : \mathbb{R}^n \times S^{n-1} \mapsto \mathbb{R}$ of set $\mathcal{O} \subset \mathbb{R}^n$ measures the signed distance from a point $\mathbf{p} \in \mathbb{R}^n$ to the set boundary $\partial\mathcal{O}$ in direction $\boldsymbol{\eta} \in S^{n-1}$:

$$h(\mathbf{p}, \boldsymbol{\eta}) := \min \left\{ d \in \mathbb{R} \;\middle|\; \mathbf{p} + d\boldsymbol{\eta} \in \partial\mathcal{O} \right\}. \qquad (1)$$

This definition is inspired by the implicit SDF representation of a set [3], [17]. SDF measures the distance from $\mathbf{p}$ to the surface $\partial\mathcal{O}$ along the direction that minimizes the distance, while SDDF measures the distance to $\partial\mathcal{O}$ in a *specific* direction $\boldsymbol{\eta}$. Also, unlike an SDF, which is negative inside the surface that it models, an SDDF is negative behind the observer's point of view. The motivation for defining SDDF is that it directly models the measurement generation process of a distance sensor allowing fast observation prediction without the need for surface reconstruction. This is relevant in robotics problems where efficient prediction of visible space is needed, such as active mapping and exploration. A key property is that, if the SDDF of a set is known, we can generate arbitrary distance views to the set boundary. In other words, we can image what a distance sensor would see from any point $\mathbf{p}$ in any viewing direction $\boldsymbol{\eta}$. The SDDF definition is illustrated in Fig. 1.

Our definition makes the distance continuous along a viewing direction with positive values before the first surface and negative values afterwards. The distance to the surface $\partial\mathcal{O}$ decreases linearly as the observer approaches $\partial\mathcal{O}$ along the viewing direction $\boldsymbol{\eta}$. To formulate this property mathematically, consider an observer located at position $\mathbf{p}_1$. Let the SDDF to the surface $\partial\mathcal{O}$ in direction $\boldsymbol{\eta}$ be $h(\mathbf{p}_1, \boldsymbol{\eta})$. Suppose that the observer moves δ units along the viewing direction $\boldsymbol{\eta}$ towards $\partial\mathcal{O}$. Now, the observer is at position $\mathbf{p}_2 = \mathbf{p}_1 + \delta\boldsymbol{\eta}$ and the SDDF $h(\mathbf{p}_2, \boldsymbol{\eta})$ in direction $\boldsymbol{\eta}$ is decreased by δ units, i.e., $\delta = h(\mathbf{p}_1, \boldsymbol{\eta}) - h(\mathbf{p}_2, \boldsymbol{\eta})$. This property is stated below, where $\nabla_{\mathbf{p}} h(\mathbf{p}, \boldsymbol{\eta})^\top \boldsymbol{\eta} = \lim_{\delta \to 0} \frac{h(\mathbf{p}+\delta\boldsymbol{\eta}, \boldsymbol{\eta}) - h(\mathbf{p}, \boldsymbol{\eta})}{\delta}$ is the gradient of h with respect to $\mathbf{p}$ projected along the direction $\boldsymbol{\eta}$.

Lemma 1. *The gradient of an SDDF $h(\mathbf{p}, \boldsymbol{\eta})$ with respect to the position $\mathbf{p}$ projected to the viewing direction $\boldsymbol{\eta}$ satisfies:*

$$\nabla_{\mathbf{p}} h(\mathbf{p}, \boldsymbol{\eta})^\top \boldsymbol{\eta} = -1. \tag{2}$$

B. SDDF Structure

Next, we study the class of functions that satisfy the SDDF gradient property in Lemma 1. We show that, while the domain $\mathbb{R}^n \times S^{n-1}$ of an SDDF has $2n-1$ degrees of freedom, the constraint in (2) removes one additional degree of freedom. To see this, we first simplify the condition in (2) by defining a function $g(\mathbf{p}, \boldsymbol{\eta}) := h(\mathbf{p}, \boldsymbol{\eta}) + \mathbf{p}^\top \boldsymbol{\eta}$. Since the viewing direction $\boldsymbol{\eta}$ is a unit vector, we have $\boldsymbol{\eta}^\top \boldsymbol{\eta} = 1$ and (2) is equivalent to the requirement that:

$$\nabla_{\mathbf{p}} g(\mathbf{p}, \boldsymbol{\eta})^\top \boldsymbol{\eta} = 0. \tag{3}$$

The zero-gradient condition in (3) suggests that $g(\mathbf{p}, \boldsymbol{\eta})$ is constant along some direction and, hence, has only $2n-2$ degrees of freedom. To show this rigorously, we rotate the coordinate system so that the viewing direction $\boldsymbol{\eta}$ lies along the last coordinate axis. For example, in $n = 3$ dimensions we can align $\boldsymbol{\eta}$ with the z-axis, showing that the third element of the gradient of $g(\mathbf{p}, \boldsymbol{\eta})$ should be zero or equivalently that $g(\mathbf{p}, \boldsymbol{\eta})$ is constant along the third dimension in the rotated reference frame.

The rotation matrix $\mathbf{R} \in SO(n)$ that rotates a unit vector $\mathbf{x} \in S^{n-1}$ to another unit vector $\mathbf{y} \in S^{n-1}$ with $\mathbf{y} \neq -\mathbf{x}$ along the sphere geodesic [18] (shortest path) is:

$$\mathbf{R} = \mathbf{I} + \mathbf{y}\mathbf{x}^\top - \mathbf{x}\mathbf{y}^\top + \frac{1}{1 + \mathbf{x}^\top \mathbf{y}} (\mathbf{y}\mathbf{x}^\top - \mathbf{x}\mathbf{y}^\top)^2. \tag{4}$$

Using (4), we can obtain an explicit expression for the rotation $\mathbf{R}_\eta$ that aligns a viewing direction $\boldsymbol{\eta}$ with the unit vector $\mathbf{e}_n = [0, \ldots, 0, 1]^\top$. Lemma 2 and 3 provide the rotation matrices for the 2D and 3D cases.

Lemma 2. *A vector $\boldsymbol{\eta} = [a, b]^\top \in S^1$ can be rotated to $\mathbf{e}_2 \in S^1$ via the rotation matrix $\mathbf{R}_{\boldsymbol{\eta}} = \begin{bmatrix} b & -a \\ a & b \end{bmatrix} \in SO(2)$.*

Lemma 3. *A vector $\boldsymbol{\eta} = [a, b, c]^\top \in S^2$ can be rotated to $\mathbf{e}_3 \in S^2$ via the rotation matrix $\mathbf{R}_{\boldsymbol{\eta}} \in SO(3)$ below:*

$$\mathbf{R}_\eta = \begin{cases} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} & \textit{if } \boldsymbol{\eta} = -\mathbf{e}_3, \\ \begin{bmatrix} 1 - \frac{a^2}{1+c} & -\frac{ab}{1+c} & -a \\ -\frac{ab}{1+c} & 1 - \frac{b^2}{1+c} & -b \\ a & b & c \end{bmatrix} & \textit{otherwise.} \end{cases} \tag{5}$$

To reveal the structure implied by the condition in (3), we express it in a rotated coordinate frame where $\mathbf{e}_n = \mathbf{R}_{\boldsymbol{\eta}} \boldsymbol{\eta}$ and $\mathbf{q} = \mathbf{R}_{\boldsymbol{\eta}} \mathbf{p}$. By the chain rule:

$$0 = \frac{\partial g}{\partial \mathbf{p}} \boldsymbol{\eta} = \frac{\partial g}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial \mathbf{p}} \boldsymbol{\eta} = \frac{\partial g}{\partial \mathbf{q}} \mathbf{R}_{\boldsymbol{\eta}} \boldsymbol{\eta} = \frac{\partial g}{\partial \mathbf{q}} \mathbf{e}_n = \frac{\partial g}{\partial q_n}. \tag{6}$$

The functions that satisfy (6) do not depend on the last element of $\mathbf{q}$. In other words, any function $g(\mathbf{p}, \boldsymbol{\eta})$ that satisfies (6) can be expressed as $g(\mathbf{p}, \boldsymbol{\eta}) = f(\mathbf{P}\mathbf{R}_{\boldsymbol{\eta}}\mathbf{p}, \boldsymbol{\eta})$ for some function $f : \mathbb{R}^{n-1} \times S^{n-1} \mapsto \mathbb{R}$ and a projection matrix $\mathbf{P} := [\mathbf{I}\ \mathbf{0}] \in \mathbb{R}^{(n-1)\times n}$, which drops the last element of $\mathbf{q} = \mathbf{R}_{\boldsymbol{\eta}}\mathbf{p}$. We summarize this property of SDDF below.

Proposition 1. *A function $h : \mathbb{R}^n \times S^{n-1} \mapsto \mathbb{R}$ is a valid SDDF, i.e., satisfies Def. 1 and the gradient condition (2) in Lemma 1, if and only if:*

$$h(\mathbf{p}, \boldsymbol{\eta}) = f(\mathbf{P}\mathbf{R}_{\boldsymbol{\eta}}\mathbf{p}, \boldsymbol{\eta}) - \mathbf{p}^\top \boldsymbol{\eta} \tag{7}$$

for some function $f : \mathbb{R}^{n-1} \times S^{n-1} \mapsto \mathbb{R}$ and $\mathbf{R}_{\boldsymbol{\eta}}$ defined in (4) with $\mathbf{x} = \boldsymbol{\eta}$ and $\mathbf{y} = \mathbf{e}_n$.

Proposition 1 shows that space of SDDF functions is a subset of real-valued functions with $2n-2$ degrees of freedom. For example, any SDDF in 2D with position $\mathbf{p} = [x, y]^\top$ and direction $\boldsymbol{\eta} = [a, b]^\top$ such that $a^2 + b^2 = 1$ can be expressed as $h(x, y, a, b) = f(bx - ay, a, b) - (ax + by)$. The gradient condition in (2) is satisfied by construction: $\nabla_{\mathbf{p}} h(\mathbf{p}, \boldsymbol{\eta})^\top \boldsymbol{\eta} = (f'[b, -a] - [a, b]) \cdot [a, b] = (f'b - a)a - (f'a + b)b = -a^2 - b^2 = -1$, where f' denotes the derivative of f with respect to its first argument evaluated at $(bx - ay, a, b)$.

C. Infinite SDDF Values

Proposition 1 allows learning SDDF shape representations from distance measurements without the need to enforce structure constraints explicitly. A remaining challenge is that the measured distance at some sensor position $\mathbf{p}$ along a ray $\boldsymbol{\eta}$ that does not hit the object surface is infinite. Since regression models cannot predict infinite values directly, we use an invertible function ϕ to squash the distance measurements to a finite range.

Lemma 4. *Let $\phi : \mathbb{R} \mapsto \mathbb{R}$ be a function with non-zero derivative, $\phi'(x) \neq 0$, for all $x \in \mathbb{R}$. Then, for any function $g : \mathbb{R}^n \times S^{n-1} \mapsto \mathbb{R}$ and vector $\boldsymbol{\eta} \in S^{n-1}$, we have:*

$$\nabla_{\mathbf{p}} g(\mathbf{p}, \boldsymbol{\eta})^\top \boldsymbol{\eta} = 0 \quad \textit{if and only if} \quad \nabla_{\mathbf{p}} \phi(g(\mathbf{p}, \boldsymbol{\eta}))^\top \boldsymbol{\eta} = 0.$$

Proof. The claim is concluded by the chain rule, $0 = \nabla_{\mathbf{p}} \phi(g(\mathbf{p}, \boldsymbol{\eta}))^\top \boldsymbol{\eta} = \phi'(g(\mathbf{p}, \boldsymbol{\eta})) \nabla_{\mathbf{p}} g(\mathbf{p}, \boldsymbol{\eta})^\top \boldsymbol{\eta}$, and since $\phi'(g(\mathbf{p}, \boldsymbol{\eta}))$ is never zero by assumption. □

The squashing function ϕ in Lemma 4 is either strictly increasing or strictly decreasing by the mean value theorem since ϕ'. In both cases, it has an inverse ϕ^{-1}. We assume ϕ is strictly increasing and define $q(\mathbf{p}, \boldsymbol{\eta}) := \phi(f(\mathbf{P}\mathbf{R}_{\boldsymbol{\eta}}\mathbf{p}, \boldsymbol{\eta}))$ such that as in Proposition 1:

$$h(\mathbf{p}, \boldsymbol{\eta}) = \phi^{-1}(q(\mathbf{p}, \boldsymbol{\eta})) - \mathbf{p}^\top \boldsymbol{\eta}. \tag{8}$$

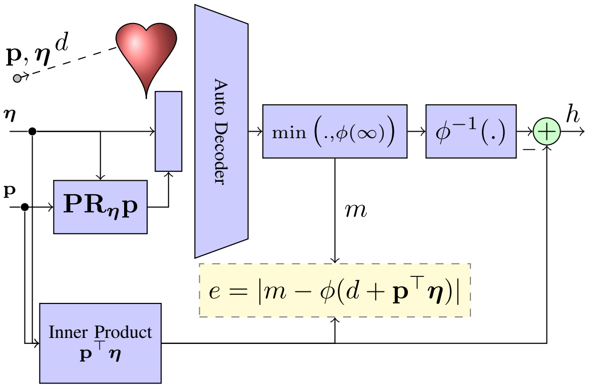
$\mathbf{p}, \boldsymbol{\eta}$
d
$\boldsymbol{\eta}$
$\mathbf{p}$
$\mathbf{PR}_{\boldsymbol{\eta}}\mathbf{p}$
Auto Decoder
$\min\left(.,\phi(\infty)\right)$
$\phi^{-1}(.)$
h
$-$
m
$e = |m - \phi(d + \mathbf{p}^\top \boldsymbol{\eta})|$
Inner Product
$\mathbf{p}^\top \boldsymbol{\eta}$

Fig. 2. Neural network model for SDDF regression. Given a position $\mathbf{p}$, viewing direction $\boldsymbol{\eta}$ and measured distance d, the model rotates $\mathbf{p}$ to new coordinates $\mathbf{R}_{\boldsymbol{\eta}}\mathbf{p}$, whose last component does not effect the SDDF value and is removed via a projection matrix $\mathbf{P}$. The projected input is processed by an autodecoder to predict a squashed distance value m, which may be converted to an SDDF value h or compared to a modified distance $d+\mathbf{p}^\top\boldsymbol{\eta}$ in the error function. Minus sign next to the plus block means that $-\mathbf{p}^\top\boldsymbol{\eta}$ will be added.

The formulation in (8) allows training a neural network model of $q(\mathbf{p}, \boldsymbol{\eta})$ with possibly infinite distance data. Due to Lemma 4, (8) is guaranteed to satisfy the SDDF property $\nabla_{\mathbf{p}} h(\mathbf{p}, \boldsymbol{\eta})^\top \boldsymbol{\eta} = -1$ by construction.

IV. DEEP SDDF MODEL

This section develops a neural network model and a cost function for learning instance-level and category-level SDDF shape models.

A. DeepSDDF Neural Network

Based on the formulation in (8), we design *DeepSDDF*, a neural network model for SDDF learning, shown in Fig. 2. The network takes position-view pairs $(\mathbf{p}, \boldsymbol{\eta})$ as input and corresponding distance measurements d as supervision. The model computes $\mathbf{PR}_{\boldsymbol{\eta}}\mathbf{p}$ analytically and then uses a fully connected autodecoder with parameters $\boldsymbol{\theta}$ to map $\mathbf{PR}_{\boldsymbol{\eta}}\mathbf{p}$ and $\boldsymbol{\eta}$ to $y = q_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta}) = \phi(f(\mathbf{PR}_{\boldsymbol{\eta}}\mathbf{p}, \boldsymbol{\eta}))$. We ensure that the autodecoder output does not exceed the maximum squashed distance, $m = \min\{y, \phi(\infty)\}$, and compare its value with the distance measurement, $\phi(d + \mathbf{p}^\top\boldsymbol{\eta})$, transformed according to (8). This neural network design guarantees that the SDDF gradient property in Lemma 2 is satisfied by construction.

B. Instance-Level DeepSDDF Training

Given distance measurements $\mathcal{D}_l$ from a single object instance l, as in Problem 1, we can learn an SDDF shape representation $h(\mathbf{p}, \boldsymbol{\eta})$ by optimizing the autodecoder parameters of the DeepSDDF model in Fig. 2. We split the training data $\mathcal{D}_l$ into sets of finite and infinite distance:

$$\begin{aligned}\mathcal{F}_l &:= \left\{(\mathbf{p}, \boldsymbol{\eta}, d) \in \mathcal{D}_l \mid d < \infty\right\},\\ \mathcal{I}_l &:= \left\{(\mathbf{p}, \boldsymbol{\eta}, d) \in \mathcal{D}_l \mid d = \infty\right\},\end{aligned} \tag{9}$$

and define an error function for training the parameters $\boldsymbol{\theta}$:

$$\begin{aligned}e(\boldsymbol{\theta}; \mathcal{F}, \mathcal{I}) := \frac{\alpha}{|\mathcal{F}|} \sum_{(\mathbf{p},\boldsymbol{\eta},d)\in\mathcal{F}} |\phi(d + \mathbf{p}^\top\boldsymbol{\eta}) - q_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta})|^p\\ + \frac{\beta}{|\mathcal{I}|} \sum_{(\mathbf{p},\boldsymbol{\eta},d)\in\mathcal{I}} r\left(\phi(\infty) - q_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta})\right)^p + \gamma\|\boldsymbol{\theta}\|_p^p,\end{aligned} \tag{10}$$

where $\alpha, \beta, \gamma > 0$ are weights, $p \geq 1$, and $r(x)$ is a rectifier, such as ReLU $r(x) = \max\{0, x\}$, GELU $r(x) = x\Phi(x)$, or softplus $r(x) = \log(1 + \exp(x))$. In the experiments, we use $p = 1$ and $r(x) = \max\{0, x\}$. The last term in (10) is used to regularize the network parameters $\boldsymbol{\theta}$ but in our experiments we set γ to zero. The first term encourages the model $q_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta})$ to predict the squashed distance values accurately. We introduced a rectifier $r(x)$ in the second term in (10) to allow the output of $q_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta})$ to exceed $\phi(\infty)$, which we observed empirically leads to faster convergence. The SDDF value predicted by the model is $h_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta}) = \phi^{-1}(\min\{q_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta}), \phi(\infty)\}) - \mathbf{p}^\top\boldsymbol{\eta}$.

C. Category-Level DeepSDDF Training

Next, we consider learning an SDDF shape model for a complete object category with L instances. Inspired by deep SDF methods [3], we introduce a latent code $\mathbf{z}_l \in \mathbb{R}^m$ to model the shape of each instance l and learn it as part of the parameters of the neural network model $q_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta}, \mathbf{z}_l)$. Given finite $\mathcal{F}_l$ and infinite $\mathcal{I}_l$ distance measurements, we optimize $\mathbf{z}_l$ independently, for each instance l, and $\boldsymbol{\theta}$ jointly, across all instances using a similar error as in (10):

$$\begin{aligned}\min_{\boldsymbol{\theta}, \{\mathbf{z}_l\}_l} \frac{\alpha}{\sum_l |\mathcal{F}_l|} \sum_l \sum_{(\mathbf{p},\boldsymbol{\eta},d)\in\mathcal{F}_l} |\phi(d + \mathbf{p}^\top\boldsymbol{\eta}) - q_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta}, \mathbf{z}_l)|^p\\ + \frac{1}{\sum_l |\mathcal{I}_l|} \sum_l \sum_{(\mathbf{p},\boldsymbol{\eta},d)\in\mathcal{I}_l} \beta r(\phi(\infty) - q_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta}, \mathbf{z}_l))^p\\ + \gamma\|\boldsymbol{\theta}\|_p^p + \sigma\frac{1}{L}\sum_l \|\mathbf{z}_l\|_p^p,\end{aligned} \tag{11}$$

where the last term regularizes the instance latent codes.

After training, a category-level SDDF shape model allows predicting the shape of a previously unseen instance at test time from a partial observation. Assume that the category-level neural network parameters $\boldsymbol{\theta}$ are already trained and we have an average category-level shape encoding $\bar{\mathbf{z}} \in \mathbb{R}^m$, e.g., obtained by using a fixed $\mathbf{z}$ for all instances l during training or simply as the mean of $\{\mathbf{z}_l\}_l$. We initialize the code of a newly observed instance with $\bar{\mathbf{z}}$ and optimize it using distance measurements $\mathcal{F}$ and $\mathcal{I}$ and the same error function as before:

$$\begin{aligned}\min_{\mathbf{z}} \frac{\alpha}{|\mathcal{F}|} \sum_{(\mathbf{p},\boldsymbol{\eta},d)\in\mathcal{F}} |\phi(d + \mathbf{p}^\top\boldsymbol{\eta}) - q_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta}, \mathbf{z})|^p\\ + \frac{\beta}{|\mathcal{I}|} \sum_{(\mathbf{p},\boldsymbol{\eta},d)\in\mathcal{I}} r(\phi(\infty) - q_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta}, \mathbf{z}))^p + \sigma\|\mathbf{z}\|_p^p.\end{aligned} \tag{12}$$

The optimized latent code $\mathbf{z}^*$ captures shape information about the instance and can be used to synthesize distances to its surface from any point $\mathbf{p}$ in any viewing direction $\boldsymbol{\eta}$ by a single forward pass through the SDDF model.

V. Evaluation

We evaluate our DeepSDDF model for shape learning in instance-level, category-level, and scene-level reconstruction. Sec. V-A presents single-instance shape modeling using real point cloud data from the YCB dataset [19]. Sec. V-B applies DeepSDDF to category-level shape modeling, demonstrating shape completion for a novel instance based on a single distance view and shape interpolation among different instances using categories from ShapeNet [20]. The shape completion accuracy of DeepSDDF is compared against the IGR model [3] for SDF estimation, which improves over DeepSDF [17] due to the Eikonal regularization but otherwise uses the same neural network. Both our model (by construction) and IGR (via loss function) capture structural constraints for SDDF and SDF, respectively, making a quantitative comparison interesting. Sec. V-C demonstrates scene-level reconstruction using real depth images from the SceneNN dataset [21]. Due to space limitation, we provide additional details and experiments, including ablation study, in [22], [23].

Metrics: We use shape reconstruction metrics introduced by Occupancy Networks [24] based on point-cloud comparison, including Chamfer distance, Completeness, and Accuracy. For all metrics *lower* is *better* and all metrics are zero for two identical point clouds. The reported times in the tables are the average ray-tracing time for single ray.

Visualization: We emphasize that our DeepSDDF model can synthesize point clouds with arbitrary resolution from arbitrary views as efficiently as a single neural-network forward pass. The generated point clouds can be converted to a different representation, such as occupancy grid, depth images, or mesh if desired.

A. *Instance-Level Shape Modeling (Real Data)*

First, we show that DeepSDDF can reconstruct instance-level shape using real data from the YCB dataset [19].

Network Architecture: We use an autodecoder with 8 layers, 512 hidden units per layer, and a skip connection from the input to layer 4.

Training Details: Depth images, segmentation masks, and camera poses for 20 object instances from YCB [19] were used for training. The masks are used to separate the (finite) rays that hit an object from the background (provided as infinite rays to our model). For each object in YCB, there are 600 views from which images are taken. The data was split randomly into a 95% training set and 5% test set. Models were trained with learning rate 0.005 that decreases every $1k$ epoch by factor of 0.5 for $10k$ epochs.

Results: Qualitative results are presented in Fig. 4. The average of quantitative results over 20 objects are provided in Table I. This experiment shows that DeepSDDF is able to train directly from real data (95% of data) and predicts the distance at novel views (5% of data) from which an object has never been observed from.

B. *Category-Level Shape Modeling*

In this section, we explore the capability of our method to represent a whole category of object shapes and compare its performance versus the deep SDF model IGR [3].

TABLE I

Average reconstruction metrics of an 8-layer DeepSDDF model trained with real data from YCB [19] on 20 objects

Chamfer-L_2	Chamfer-L_1	Completeness	Accuracy	Time (sec.)
1.121e-05	1.294e-03	6.854e-04	1.903e-03	5.829e-07

Network Architecture: We use an autodecoder with 16 layers, 512 hidden units per layer, and a skip connection from the input to layers $4, 8, 12$ to represent the SDDF model $q_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta}, \mathbf{z})$ introduced in Sec. IV with dimension of the latent shape code $\mathbf{z}$ set to 256.

Training Details: DeepSDDF and IGR were trained over 5 categories from ShapeNet [20]. Distance images with resolution 512×512 were generated as training data from 8 camera views facing the object from azimuth $\frac{k\pi}{4}$ and elevation $\frac{(-1)^k\pi}{4}$ for $k = 0, \ldots, 7$ on a sphere. Each distance image was subsampled to contain at most $100k$ finite and $100k$ infinite distance measurement rays. IGR was trained using the point-cloud obtained from all points with finite distance measurements and augmented with normals obtained using the method of [25]. Normals were not used to train DeepSDDF. DeepSDDF and IGR were trained for 1000 epochs, with 2000 random samples, and with learning rate 0.0005, 0.0001 for the network weights and latent code weights, respectively. The learning rate decreases by factor of 2 every 500 epochs for IGR (as suggested in [3]) and every 200 epochs for DeepSDDF.

Shape Completion: The shape reconstruction accuracy and timing of the DeepSDDF and IGR models are presented in Table II. Two versions of IGR are evaluated: IGR-Mesh, reconstructing a mesh using marching cubes [4] and sampling a uniform point cloud on its surface, and IGR-Sphere, performing sphere tracing via IDR [16] to generate a point cloud. The results show that our model is an order of magnitude more accurate than IGR with limited training data, despite training IGR with normals and encoding additional directional information in SDDF. This can be explained by the fact that IGR encourages the satisfaction of the SDF gradient property [3] through regularization, while DeepSDDF encodes the SDDF gradient property (Lemma 1) by construction, making DeepSDDF more data efficient. Comparing the results of IGR-Mesh and IGR-Sphere shows that the accuracy of deep SDF estimation may deteriorate with distance from the surface. We used an open-source sphere-tracing algorithm from IDR [16] with default parameters, except for $\mathit{sphere\_tracing\_iters} = 50$, which led to faster and accurate results in our experiments. The results also show that DeepSDDF is an order of magnitude faster than IGR for ray tracing because DeepSDDF requires a single forward pass, while IGR performs iterative sphere tracing.

Shape Interpolation: We also demonstrate that DeepSDDF is able to model the latent shape space of an object category continuously and meaningfully. Fig. 3 shows interpolation between the latent shape codes of two instances from the same category. The intermediate shapes do not exist in the training set and are imagined by the DeepSDDF decoder based on the interpolated latent code $\mathbf{z}$.

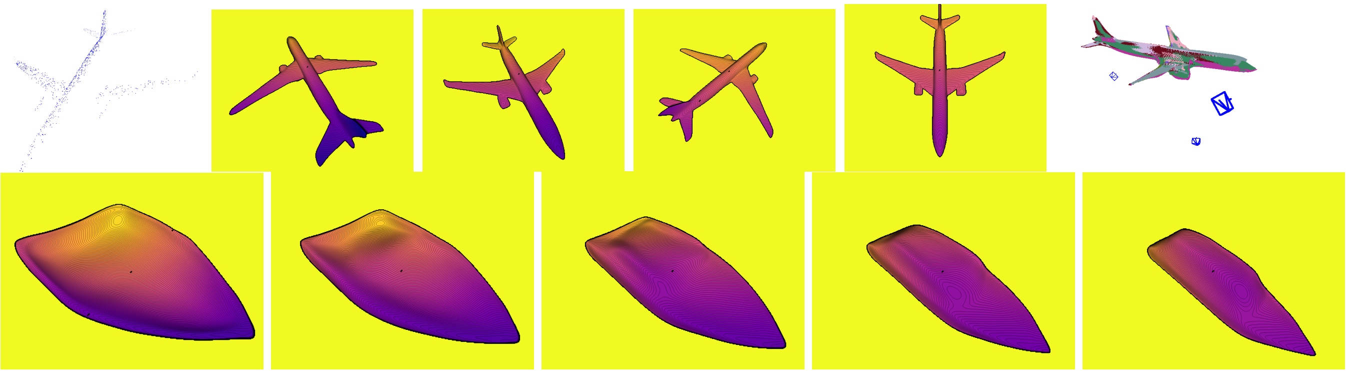

Fig. 3. SDDF shape completion (first row) using a distance measurement from an unseen airplane instance (first column). The trained DeepSDDF model can synthesize novel distance views (middle four columns) or point clouds from arbitrary camera views which are multi-view consistent (last column). The second row shows an example of shape interpolation between two boat instance in the first and last column. The middle shapes are generated with latent code obtained as the average between the two far-end instance codes. From left to right, the latent code weights with respect to the left-most instance are 1, 0.75, 0.5, 0.25, 0, respectively.

TABLE II
COMPARISON BETWEEN DEEPSDDF AND IGR [3] ON 5 SHAPENET CATEGORIES [20] WITH MESHES NORMALIZED TO UNIT-LENGTH BOUNDING BOX

Category	Method	Chamfer-L_2	Chamfer-L_1	Completeness	Accuracy	Time (sec.)
Car	DeepSDDF	1.971e-04	7.982e-03	7.609e-03	8.355e-03	1.190e-06
	IGR-Mesh	3.599e-03	3.869e-02	1.646e-02	6.092e-02	–
	IGR-Sphere	9.091e-03	6.937e-02	3.279e-02	1.059e-01	1.772e-05
Airplane	DeepSDDF	2.332e-04	7.707e-03	6.407e-03	9.008e-03	1.175e-06
	IGR-Mesh	1.774e-02	8.798e-02	2.076e-02	1.551e-01	–
	IGR-Sphere	2.840e-02	1.172e-01	2.880e-02	2.057e-01	2.317e-05
Watercraft	DeepSDDF	5.209e-04	1.288e-02	1.147e-02	1.428e-02	1.192e-06
	IGR-Mesh	7.075e-03	5.480e-02	2.751e-02	8.210e-02	–
	IGR-Sphere	3.326e-02	1.231e-01	3.855e-02	2.077e-01	2.010e-05
Sofa	DeepSDDF	3.850e-04	1.221e-02	1.093e-02	1.348e-02	1.195e-06
	IGR-Mesh	1.151e-02	6.923e-02	4.083e-02	9.764e-02	–
	IGR-Sphere	5.414e-02	1.563e-01	5.594e-02	2.567e-01	2.239e-05
Display	DeepSDDF	9.200e-04	1.789e-02	1.551e-02	2.027e-02	1.184e-06
	IGR-Mesh	8.883e-03	5.063e-02	3.493e-02	6.633e-02	–
	IGR-Sphere	2.139e-02	9.600e-02	4.030e-02	1.516e-01	1.951e-05
Avg.	DeepSDDF	4.512e-04	1.173e-02	1.038e-02	1.307e-02	1.187e-06
	IGR-Mesh	9.761e-03	6.026e-02	2.809e-02	9.241e-02	–
	IGR-Sphere	2.925e-02	1.123e-01	3.927e-02	1.855e-01	2.057e-05

C. *Scene-Level Reconstruction and Robotics Applications*

Finally, we apply DeepSDDF to learn part of a scene using real depth images from the SceneNN dataset [21]. This experiment shows that a single DeepSDDF network is capable of modeling multiple objects. Once the SDDF model is trained, it can be used to perform visibility and visible volume queries very efficiently (as a single forward network pass). Fast assessment of the visible volume of an environment from novel views is a key component in the next-best-view problem [26] and other active perception problems [2] in robotics.

Network Architecture: We use an autodecoder with 16 layers, 512 hidden units per layer, and a skip connection from the input to layers $4, 8, 12$ to represent the SDDF model $q_{\boldsymbol{\theta}}(\mathbf{p}, \boldsymbol{\eta})$ introduced in Sec. IV.

Training Details: We used 38 depth images chosen every 50 steps starting from image 0 of sequence 255 from the SceneNN dataset [21]. For each image, we randomly chose 20k colliding rays and 20k non-colliding rays, effectively subsampling the depth image resolution. The network was trained with learning rate 0.005 that decreases every $1k$ epoch by factor of 0.5 for $10k$ epochs.

Scene Reconstruction and Visibility Queries: A reconstruction of the 3D scene in SceneNN sequence 255 are shown in Fig. 4. Additionally, the SDDF function in a horizontal plane cutting the scene is shown. To illustrate the utility of an SDDF model in active perception applications, we consider two arbitrary poses (shown in Fig. 4) and evaluate the visible volume from their views. We approximate the visible volume by considering the volume corresponding to each pixel as a pyramid. For example for the yellow camera in Fig. 4 with resolution 480×640 the visible volume is 0.307 (more accurate due to higher resolution), computed by the DeepSDDF model. With resolution 240×320, the computed visible volume is 0.305.

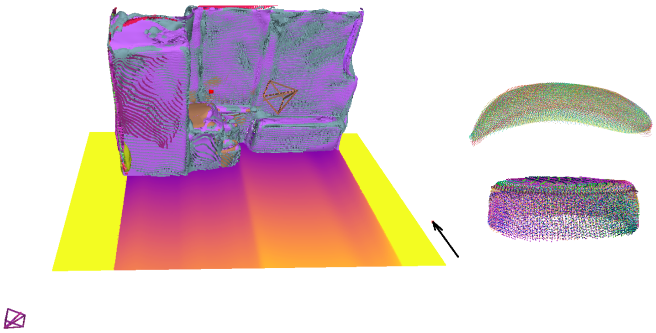

Fig. 4. Left: Scene-level SDDF reconstruction using 38 real depth images from the SceneNN dataset [21]. A point cloud surface reconstruction from 8 test views, two of which are shown as a purple camera frame (left) and yellow camera frame (close to the surface). The ground plane shows the SDDF value at each point in the viewing direction indicated by the black arrow with yellow color indicating infinite values (truncated by some max value) and red-to-blue transition indicating large to small signed distance. The red point on the surface is a sample point which is visible from the yellow camera but not from the purple camera. The DeepSDDF model can handle such visibility queries very efficiently, as a single forward pass through the neural network. Right: Two samples of novel-view point clouds synthesized by a DeepSDDF model trained using real data from YCB [19].

Similarly, the visibility of a query point $\mathbf{q}$ of interest from a camera with position $\mathbf{p}$ may be evaluated as single DeepSDDF pass i.e., checking $\|\mathbf{p} - \mathbf{q}\| > h(\mathbf{p}, \frac{\mathbf{q}-\mathbf{p}}{\|\mathbf{q}-\mathbf{p}\|})$, where h is SDDF.

D. Limitations

Assume we have a depth ray in the training set that has the direction $\boldsymbol{\eta}$ and end ray point $\mathbf{p}$. Consider an arbitrary ray that collides with the surface at point $\mathbf{p}$. For input samples, with their point element on this ray and their direction element to be the ray direction, Lemma 1, theoretically guarantees that we need one sample of these input samples. We do not need to, for example, segment this ray and collect more training samples. Since the Lemma 1 theoretically guarantees the correctness for all if one of them returns correct distance. However, this lemma does not theoretically guarantee the correctness of distances when we observe the point $\mathbf{p}$ from different directions. As a result for other rays, that collides with the surface at point $\mathbf{p}$, with different viewing direction other than $\boldsymbol{\eta}$, we need one more sample per ray. We have already observed $\mathbf{p}$ from direction $\boldsymbol{\eta}$ in training data, but we should add more input samples with different viewing directions such that the corresponding ray collides with the surface at point $\mathbf{p}$. To address this limitation, we developed a data augmentation technique in [22], [23] that adds more viewing directions for point $\mathbf{p}$ (in the training data), to training data.

VI. Conclusion

This work proposed a signed directional distance function as an implicit shape representation. Any valid SDDF was shown to satisfy a gradient condition, which should be respected by neural network models for SDDF learning. We designed DeepSDDF, a neural network model for SDDF estimation that that guarantees the gradient condition by construction. Our model enables direct supervision from depth camera or LiDAR sensors and efficient ray-tracing as a single network forward pass. DeepSDDF showed superior shape reconstruction accuracy and rendering time in instance-level and category-level shape learning experiments compared to deep SDF models. SDDF is a promising representation for robotics applications requiring accurate continuous shape and surface modeling with highly efficient visibility prediction.

References

[1] Z. Xia, Z. Deng, B. Fang, Y. Yang, and F. Sun, "A review on sensory perception for dexterous robotic manipulation," *International Journal of Advanced Robotic Systems*, vol. 19, no. 2, 2022.

[2] J. A. Placed, J. Strader, H. Carrillo, N. Atanasov, V. Indelman, L. Carlone, and J. A. Castellanos, "A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers," *IEEE Transactions on Robotics*, pp. 1–20, 2023.

[3] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit Geometric Regularization for Learning Shapes," in *International Conference on Machine Learning (ICML)*, pp. 3789–3799, 2020.

[4] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm," *Computers & Graphics*, vol. 30, no. 5, pp. 854–879, 2006.

[5] J. C. Hart, "Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces," *The Visual Computer*, vol. 12, no. 10, pp. 527–545, 1996.

[6] W. Yifan, F. Serena, S. Wu, C. Öztireli, and O. Sorkine-Hornung, "Differentiable Surface Splatting for Point-based Geometry Processing," *ACM Transactions on Graphics*, vol. 38, no. 6, 2019.

[7] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik, "Learning shape abstractions by assembling volumetric primitives," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2635–2643, 2017.

[8] D. Paschalidou, A. O. Ulusoy, and A. Geiger, "Superquadrics revisited: Learning 3d shape parsing beyond cuboids," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[9] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3d surface generation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 216–224, 2018.

[10] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[11] J. T. Kajiya and B. P. Von Herzen, "Ray tracing volume densities," *ACM SIGGRAPH Computer Graphics*, vol. 18, no. 3, pp. 165–174, 1984.

[12] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *European Conference on Computer Vision*, pp. 405–421, 2020.

[13] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections," in *IEEE/CVF Conference on Computer Visioan and Pattern Recognition (CVPR)*, 2021.

[14] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, "GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 20154–20166, 2020.

[15] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, 2011.

[16] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, B. Ronen, and Y. Lipman, "Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance," in *Advances in Neural Information Processing Systems*, vol. 33, pp. 2492–2502, 2020.

[17] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *IEEE/CVF Conference on Computer Visioan and Pattern Recognition (CVPR)*, pp. 165–174, 2019.

[18] J. Ángel Cid and F. A. F. Tojo, "A Lipschitz condition along a transversal foliation implies local uniqueness for ODEs," *Electronic Journal of Qualitative Theory of Differential Equations, arXiv:1801.01724*, vol. 36, no. 4, pp. 1–13, 2018.

[19] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The YCB object and Model set: Towards common benchmarks for manipulation research," in *International Conference on Advanced Robotics (ICAR)*, pp. 510–517, 2015.

[20] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," *arXiv:1512.03012*, 2015.

[21] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung, "SceneNN: A scene meshes dataset with annotations," in *IEEE International Conference on 3D Vision (3DV)*, pp. 92–101, 2016.

[22] E. Zobeidi and N. Atanasov, "A deep signed directional distance function for object shape representation," *arXiv: 2107.11024*, 2021.

[23] E. Zobeidi, *Machine Learning Techniques for Shape Reconstruction and Metric-Semantic Mapping*. University of California, San Diego, 2023.

[24] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4460–4470, 2019.

[25] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv: 1801.09847*, 2018.

[26] N. Atanasov, B. Sankaran, J. L. Ny, G. Pappas, and K. Daniilidis, "Nonmyopic View Planning for Active Object Classification and Pose Estimation," *IEEE Transactions on Robotics (T-RO)*, vol. 30, no. 5, pp. 1078–1090, 2014.