This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658

Learning Scene-Level Signed Directional Distance Function with Ellipsoidal Priors and Neural Residuals

Zhirui Dai, Hojoon Shin, Yulun Tian, Ki Myung Brian Lee, Nikolay Atanasov

***Abstract*—Dense reconstruction and differentiable rendering are fundamental tightly connected operations in 3D vision and computer graphics. Recent neural implicit representations demonstrate compelling advantages in reconstruction fidelity and differentiability over conventional discrete representations such as meshes, point clouds, and voxels. However, many neural implicit models, such as neural radiance fields (NeRF) and signed distance function (SDF) networks, are inefficient in rendering due to the need to perform multiple queries along each camera ray. Moreover, NeRF and Gaussian Splatting methods offer impressive photometric reconstruction but often require careful supervision to achieve accurate geometric reconstruction. To address these challenges, we propose a novel representation called signed directional distance function (SDDF). Unlike SDF and similar to NeRF, SDDF has a position and viewing direction as input. Like SDF and unlike NeRF, SDDF directly provides distance to the observed surface rather than integrating along the view ray. As a result, SDDF achieves accurate geometric reconstruction and efficient differentiable directional distance prediction. To learn and predict scene-level SDDF efficiently, we develop a differentiable hybrid representation that combines explicit ellipsoid priors and implicit neural residuals. This allows the model to handle distance discontinuities around obstacle boundaries effectively while preserving the ability for dense high-fidelity distance prediction. Through extensive evaluation against state-of-the-art representations, we show that SDDF achieves (i) competitive SDDF prediction accuracy, (ii) faster prediction speed than SDF and NeRF, and (iii) superior geometric consistency compared to NeRF and Gaussian Splatting.**

***Index Terms*—Signed directional distance function, differentiable rendering, implicit neural field, view optimization**

I. Introduction

Finding the best 3D scene representation is a challenging problem in computer vision. The appropriate representation varies between different applications depending on the required operations, such as novel view synthesis, surface reconstruction, occupancy estimation, and occlusion checking. While explicit scene representations, e.g., based on meshes [3, 4, 5], point clouds [6, 7, 8], and voxels [9], are widely used, they are not continuous and do not support differentiation. The former hurts reconstruction accuracy and novel view synthesis, while the latter hinders their uses within downstream tasks requiring end-to-end differentiable rendering.

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

Zhirui Dai, Ki Myung Brian Lee, and Nikolay Atanasov are with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093, US (e-mails: {zhdai,kmblee,natanasov}@ucsd.edu).

Hojoon Shin is with Brain Corporation, San Diego, CA 92121, US (email: hojoon.shin@braincorp.com).

Yulun Tian is with the Robotics Department, University of Michigan, MI 48109, US (email: yulunt@umich.edu).

Recent work has focused on implicit scene representations that support differentiable geometry reconstruction and novel view synthesis. For example, occupancy networks [10] and DeepSDF [11] have shown impressive results by representing surfaces as the zero level set of an occupancy probability function or a signed distance function (SDF). Neural radiance field (NeRF) [12] and Gaussian Splatting (GS) [13] models learn geometry implicitly through 2D image rendering supervision. Although these implicit differentiable methods offer superior fidelity, they require multiple network forward passes, complicated calculations per pixel/ray, and high memory use.

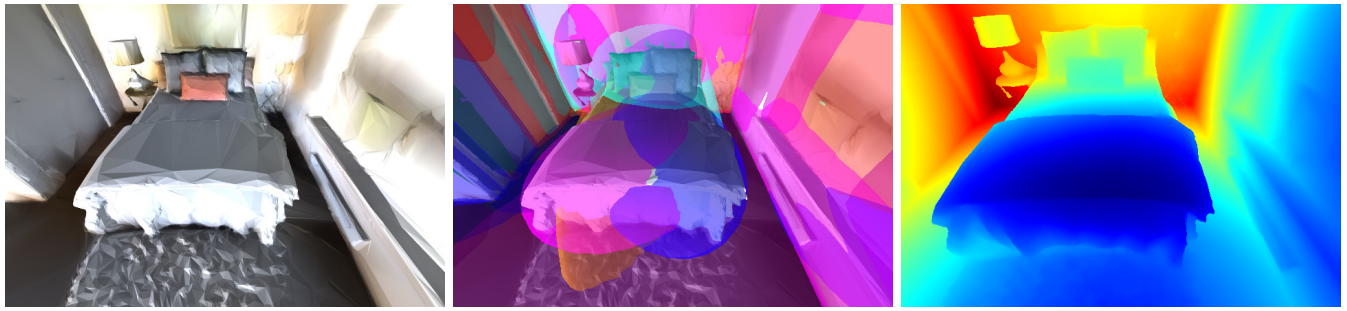
A promising recent approach that overcomes these limitations is the signed directional distance function (SDDF). SDDF is a directional formulation of SDF that takes a position and a viewing direction as input (like NeRF and unlike SDF) and provides the distance to the observed surface (like SDF and unlike NeRF). The benefits of SDDF as a scene representation are three-fold. First, SDDF models can provide fast, single forward-pass directional distance queries in a differentiable way, supporting operations such as novel view synthesis and differentiable view optimization. Second, SDDF models can be trained from different kinds of sensor data, including depth images and LiDAR scans, as long as they can be converted to ray distances. This is in contrast with SDF models, which require processing of sensor measurements to obtain SDF supervision data, and with NeRF and GS models that commonly expect camera images. Third, an SDDF model learns a geometric representation in the space of positions and directions, which allows arbitrary view synthesis and efficient occlusion queries. This is in contrast with an SDF model which requires an iterative sphere tracing algorithm [14] to compute distance in a desired direction.

However, learning scene-level geometry in the space of positions and directions is challenging. Compared to SDF, the introduction of direction as an additional input requires additional training data with diverse viewing directions to provide sufficient supervision. Another difficulty is that SDDF is sensitive to the ray positions and directions because occlusions introduce discontinuities in the observed distance. For these reasons, previous methods for learning directional distance models [15, 16] are only applicable to single-object shape modeling. In contrast, we consider learning *scene-level* SDDF, which is necessary for many applications ranging from mixed reality to robotics, where rendering and directional distance measurement are required at the scene level.

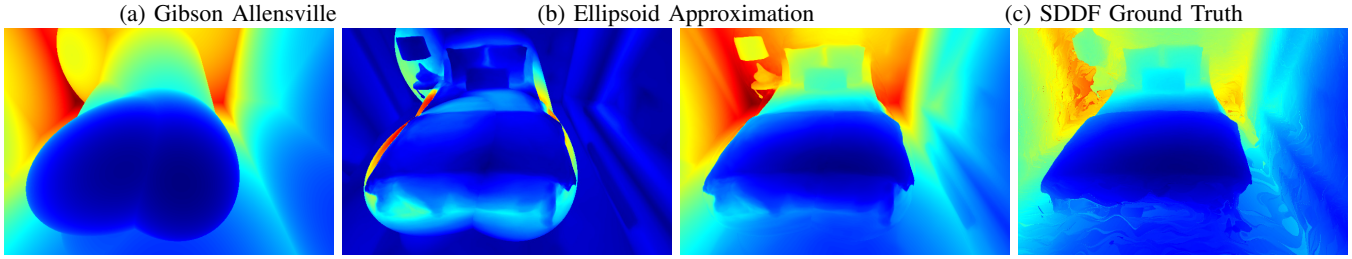
To address these challenges, we propose a method that combines the advantages of explicit and implicit representations to learn SDDF at the scene level. As shown in Fig. 1, our method first constructs an explicit ellipsoid-based prior to

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658



(a) Gibson Allensville (b) Ellipsoid Approximation (c) SDDF Ground Truth



(d) Ellipsoid-based SDDF Prior (e) SDDF Residual Prediction (f) SDDF Prediction (g) Prediction By RaDe-GS [1]

Fig. 1: (a), (c): We present a method to learn scene-level signed directional distance function (SDDF). (a), (b), (d): Our method uses ellipsoids as an initial coarse approximation of the shapes of objects in the environment. (e), (f): The ellipsoid prior is refined by a latent feature network and a shared decoder to predict the surface reconstruction residual. (f), (g): Our SDDF learning method offers single-query differentiable novel distance image synthesis without RGB supervision as an alternative to Gaussian Splat distance rendering (e.g., RaDe-GS [1]) or signed distance function sphere tracing (e.g., InstantNGP [2]).

capture the coarse structure and occlusions of the environment. Then, an implicit residual neural network model corrects the coarse ellipsoid predictions with precise details that capture the fine structure of the environment. We guarantee that both the ellipsoid prior and the residual network are differentiable, and hence, our SDDF model supports single-query novel view synthesis and differentiable view optimization. While NeRF and GS focus on photometric rendering, our focus in this paper is on geometric reconstruction with efficient depth rendering, which can benefit various applications relying on directional distances. In summary, this paper makes the following contributions.

- We introduce a new definition of SDDF suitable for scene-level representation, which extends the original SDDF definition for single objects [15] to handle occlusions in complex large-scale environments.
- We design a hybrid explicit-implicit model to approximate SDDF, consisting of an ellipsoid-based prior and an implicit neural residual, and show that the full model satisfies an Eikonal-like constraint by construction.
- We develop an algorithm to initialize the ellipsoid prior and derive the gradients with respect to its parameters in closed-form to accelerate training. Code is available at https://github.com/existentialrobotics/neural_sddf.

Our experiments show that our SDDF reconstruction method achieves competitive results against state-of-the-art, highly optimized implementations of SDF, GS, and NeRF in Replica [17] and Gibson [18] scenes, in terms of reconstruction accuracy, rendering speed, and GPU memory usage.

II. Related Work

Three bodies of work are related to ours: NeRFs, implicit geometric representations, and directional distance functions.

A. Neural Radiance Fields

The formulation of radiance fields is a recent direction in scene representation that has led to significant improvements in the synthesis of novel photometric views. The original NeRF model [12, 19] employs volumetric rendering with a neural network trained to predict the color and opacity along a view direction at a given position, which are then integrated for rendering. Such volumetric rendering, however, is computationally expensive because many samples are required for along-ray integration. To mitigate this, subsequent approaches [20, 21, 22, 23] introduce explicit data structures for faster and more accurate rendering with a smaller memory footprint. For example, PlenOctree [24] and Plenoxel [25] use voxelized data structures such as an octree [24] or a regular grid [25] to store spherical harmonics coefficients for better efficiency. Similarly, neural point methods [26, 27, 28, 29] use point clouds to store SH coefficients or neural features. Gaussian Splatting (GS) [13] and its variants such as 2D-GS [30] and rasterizing depth GS (RaDe-GS) [1] explore this direction further by using a large number of Gaussians with SH coefficients that can be explicitly rasterized to an image through a projective transform. These methods show that a hybrid model combining an explicit representation (e.g., point cloud, octree, or Gaussians) with an implicit representation (e.g., SH coefficients, or neural features) can offer a better trade-off between rendering quality and speed. Drawing from these methods, we also design a hybrid explicit-implicit scene

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658

representation. However, we focus on geometric accuracy instead of photometric accuracy. One of our key contributions in this paper is a new definition of SDDF suitable for scene-level representation. We leave the application of our SDDF representation to photometric rendering as future work.

B. Implicit Geometric Representations

Many previous scene representations consider geometric reconstruction. Distance representations such as SDF [11, 31], unsigned distance function (UDF) [32, 33], and truncated SDF (TSDF) [34], model the scene geometry explicitly and offer fast proximity queries, which are valuable in applications, such as mapping and trajectory optimization in robotics [35, 36, 37]. To scale to larger environments, an effective strategy is to use voxel hashing and incrementally estimate and store SDF in a hash table [38, 3]. Gaussian process (GP) methods [39, 40] use octrees to store incrementally estimated oriented surface points and the corresponding GP models to regress SDF. Although these methods achieve real-time incremental SDF estimation, they fail to capture small-scale details accurately.

Recent approaches such as DeepSDF [11] and implicit geometric regularization (IGR) [41] demonstrate that neural networks can learn SDF accurately, with supervision from oriented surface points and Eikonal regularization [41]. To learn a geometric representation from images instead of surface points, geometric representations can be incorporated in a radiance field and used as a means to improve photometric accuracy. For example, NeuS [42] replaces the opacity with an SDF so that the geometry (i.e., SDF) and the radiance field are learned together. RaDe-GS [1] proposes a geometric regularization term that improves both geometric and photometric accuracy. These results show that an accurate geometric representation that we pursue here is fundamentally important for both geometric and photometric accuracy. As part of our contributions, the combination of explicit geometric prior and implicit neural features achieves good balance of accuracy and efficiency in SDDF prediction, which is an important conclusion for future research on implicit geometric representations.

For geometric supervision, SDFDiff [43] and DIST [44] use depth, surface normals, or silhouette for supervision by differentiating through SDF queries in sphere tracing iterations, although the sphere tracing process [14] itself is not differentiable. Sphere tracing on SDFs can also be used to compute directional distance as we do but its iterative and cumulative nature leads not only to slower training and rendering but also to error accumulation.

C. Directional Distance Functions

Directional distance functions (DDFs) provide directional distance in a single query, and are hence more efficient and free from error accumulation compared to SDFs. Directional TSDF [45] extends TSDF in 3D to six voxel grids, each along or against the X, Y, and Z axes. Since the method uses only six discrete directions, its reconstruction accuracy may be limited, but the importance of directional information when modeling thin objects is evident. A signed ray distance function (SRDF) [46] is defined with respect to a given camera pose, as the distance of a 3D point to the scene surface along the viewing lines of the camera. This definition is useful for multiview stereo because SRDF is zero only when the query point is on the surface. Volrecon [47] uses the same term, SRDF, to instead denote simply the distance between a point and the scene surface along a ray, which is closer to our definition. This SRDF is used in volumetric rendering for multiview stereo and high-quality rendering, which shows the utility of the directional distance in capturing geometric details. However, Volrecon uses multiple transformers, and many samples along each ray, which is prohibitively slow for scaling to larger scenes.

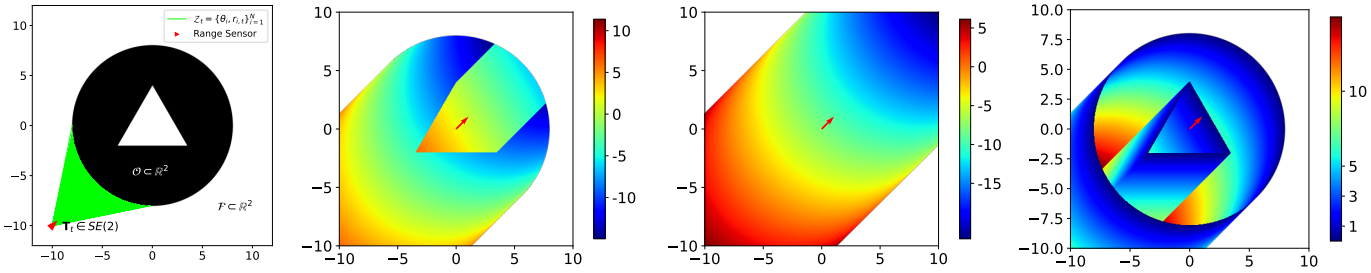
Other methods focus on learning DDFs at the object level. NeuralODF [48] and RayDF [49] learn ray-surface distances for an object bounded by a sphere that is used to parameterize the ray. Sphere-based ray parameterization cannot apply to scene-level DDF reconstruction because scenes may be unbounded. FIRe [50] combines SDF and DDF to reconstruct object shapes, where DDF renders object shapes efficiently. However, FIRe is difficult to scale to scene-level representation because it requires silhouettes for good performance, which cannot be directly captured by sensors. Pointersect [51] uses a transformer to predict the ray travel distance, the surface normal vector, and RGB color given a dense point cloud generated from multiple posed RGB-D images. Due to the dense point cloud and the transformer, Pointersect exhibits slow rendering and lower output quality when multiple surfaces exist along the ray, which limits its scalability. The network architecture of [15] is specialized in learning the SDDF at the object level. It satisfied the directional Eikonal constraint by construction. However, their definition and design are suitable only for learning a single object. The probabilistic DDF (PDDF) [16] introduces an additional "probability" network to learn the discontinuities caused by occlusions. The method is designed to learn PDDF at the object level. Although multiple objects' PDDFs can be fused into a scene PDDF, its scalability is limited, since many independent networks are needed for each object in larger scenes. Moreover, the omission of sign information obfuscates whether a query position is inside or outside the objects. Instead, as shown in Fig. 2, our definition of SDDF has fewer discontinuities than DDF, and can encode complicated geometric structures ranging from small objects to large scenes. Moreover, we design a hybrid explicit-implicit model to approximate SDDF. The explicit ellipsoidal SDDF prior not only learns a coarse SDDF approximation but also helps to efficiently handle discontinuities. The implicit neural residual further refines the SDDF prediction to capture fine geometric details.

III. Method Overview

We aim to design a method for range sensors, such as LiDARs or depth cameras, to learn an environment model capable of efficient and differentiable synthesis of arbitrary distance views. Let the occupied space in the environment be represented by a set $\mathcal{O} \subset \mathbb{R}^n$, where n is the dimension ($n \in \{2, 3\}$ in practice). Consider a set of measurements $\{\mathbf{T}_t, \mathcal{Z}_t\}_{t=1}^{T}$, where $\mathbf{T}_t \in SE(n)$ is the sensor pose at time

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658



(a) 2D Scene (b) SDDF (45°) (c) Object SDDF (45°) [15] (d) DDF (45°) [16]

Fig. 2: Example of our scene-level SDDF, the object-level SDDF of [15], and the DDF of [16] in a 2D synthetic environment. (a): a range sensor (red triangle) with pose $\mathbf{T}_t \in SE(2)$ is measuring the distance to a doughnut-like obstacle $\mathcal{O}$ (black) with a triangular hole in the middle (left plot). At time t, the sensor measurement $\mathcal{Z}_t = \{\theta_i, r_{i,t}\}_{i=1}^N$ consists of N range measurements r_i obtained along rays (green lines) cast at angles θ_i. Measurements from multiple time steps $\{\mathcal{Z}_t\}_{t=0}^{T-1}$ are collected. (b) to (d): the red arrow in the three plots on the right labels the viewing direction. The white region in the right three plots indicates invalid/infinite field values. Unlike DDF, our SDDF definition is continuous when transitioning from free to occupied space along the viewing direction. Compared with object SDDF [15], our SDDF definition reflects the geometry well, allowing scene-level reconstruction.

t and $\mathcal{Z}_t = \{\mathbf{v}_i, r_{t,i}\}_{i=1}^N$ are the measurements at time t, consisting of N viewing directions $\mathbf{v}_i \in \mathbb{S}^{n-1}$ (e.g., unit vector in the direction of each depth camera pixel or unit vector in the direction of each LiDAR scan ray) and the corresponding range measurements $r_{t,i} \in \mathbb{R}_{>0}$. Our objective is to learn a representation of the occupied space $\mathcal{O}$ in the form of a signed directional distance function.

Definition 1. The *signed directional distance function* (SDDF) of a set $\mathcal{O} \subset \mathbb{R}^n$ is a function $f : \mathbb{R}^n \times \mathbb{S}^{n-1} \to \mathbb{R} \cup \{\pm\infty\}$ that measures the signed distance from a point $\mathbf{p} \in \mathbb{R}^n$ to the set boundary $\partial\mathcal{O}$ along a direction $\mathbf{v} \in \mathbb{S}^{n-1}$, defined as:

$$f(\mathbf{p}, \mathbf{v}; \mathcal{O}) := \begin{cases} \min\{d > 0 \mid \mathbf{p} + d\mathbf{v} \in \partial\mathcal{O}\}, & \mathbf{p} \notin \mathcal{O}, \\ \max\{d \leq 0 \mid \mathbf{p} + d\mathbf{v} \in \partial\mathcal{O}\}, & \mathbf{p} \in \mathcal{O}. \end{cases} \quad (1)$$

The SDDF definition is illustrated in Fig. 2. For comparison, the signed distance function (SDF) of a set $\mathcal{O}$ is defined as the shortest distance from $\mathbf{p} \in \mathbb{R}^n$ to the boundary $\partial\mathcal{O}$:

$$f_{\text{SDF}}(\mathbf{p}; \mathcal{O}) := \begin{cases} \min_{\mathbf{y} \in \partial\mathcal{O}} \|\mathbf{p} - \mathbf{y}\|_2, & \mathbf{p} \notin \mathcal{O}, \\ -\min_{\mathbf{y} \in \partial\mathcal{O}} \|\mathbf{p} - \mathbf{y}\|_2, & \mathbf{p} \in \mathcal{O}. \end{cases} \quad (2)$$

The SDF and SDDF of $\mathcal{O}$ are related as follows:

$$f_{\text{SDF}}(\mathbf{p}; \mathcal{O}) = \min_{\mathbf{v} \in \mathbb{S}^{n-1}} f(\mathbf{p}, \mathbf{v}; \mathcal{O}). \quad (3)$$

It is well known [41, 40] that SDFs satisfy an Eikonal equation $\|\nabla_{\mathbf{p}} f_{\text{SDF}}(\mathbf{p}; \mathcal{O})\|_2 = 1$, which is useful for regularizing or designing the structure of models for estimating SDF. The next proposition shows that SDDF satisfies a similar property.

Proposition 1. *Suppose an SDDF $f(\mathbf{p}, \mathbf{v}; \mathcal{O})$ is differentiable at $\mathbf{p} \in \mathbb{R}^n$. Then, it satisfies a directional Eikonal equation:*

$$\mathbf{v}^\top \nabla_{\mathbf{p}} f(\mathbf{p}, \mathbf{v}; \mathcal{O}) = -1. \quad (4)$$

Proof. When the ray hits the same surface point $\mathbf{q} = \mathbf{p} + d\mathbf{v}$ as $\mathbf{p}$ moves along $\mathbf{v}$, we have:

$$\mathbf{v}^\top \nabla_{\mathbf{p}} f(\mathbf{p}, \mathbf{v}; \mathcal{O}) = \lim_{\delta \to 0} \frac{f(\mathbf{p} + \delta\mathbf{v}, \mathbf{v}) - f(\mathbf{p}, \mathbf{v})}{\delta}$$
$$= \lim_{\delta \to 0} \frac{(d - \delta) - d}{\delta} = -1. \quad \blacksquare$$

This directional Eikonal equation indicates that as the position $\mathbf{p}$ moves towards (away from) the intersected surface along the direction $\mathbf{v}$, the SDDF value decreases (increases) at unit rate. This property is useful for designing the structure or regularizing a neural network representation of SDDF.

Previous work [15] proposed an object SDDF that is defined as $\min\{d \in \mathbb{R} \mid \mathbf{p} + d\mathbf{v} \in \partial\mathcal{O}\}$. However, this definition is only suitable for learning the shape of a single object because when there is another object behind the viewing position $\mathbf{p}$, the object SDDF will ignore the object that the viewing direction $\mathbf{v}$ is pointing at. Fig. 2c shows an example of this issue.

PDDF [16] addresses this problem by learning directional distance without sign, defined as $\min\{d \geq 0 \mid \mathbf{p} + d\mathbf{v} \in \partial\mathcal{O}\}$. However, as shown in Fig. 2d, this definition introduces several discontinuities, which is not favorable for learning a directional distance model from data.

Our definition of SDDF, shown in Fig. 2b, guarantees that the distance continuously changes from positive to negative values when a viewing ray $\mathbf{v}$ enters the occupied space $\mathcal{O}$ from *outside in*. However, when the ray leaves $\mathcal{O}$ from *inside out*, there is a discontinuity in the distance. Fortunately, such a discontinuity does not occur in practice because sensors remain within free space, only observing $\mathcal{O}$ from outside-in. Thus, our definition can accurately model occlusions while retaining the sign to distinguish whether the query position is in free space, unlike previous formulations of DDF [16].

Yet, learning a solely continuous, implicit representation of our new SDDF definition remains challenging. Unlike an SDF, SDDF is sensitive to both position and direction, as small

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658

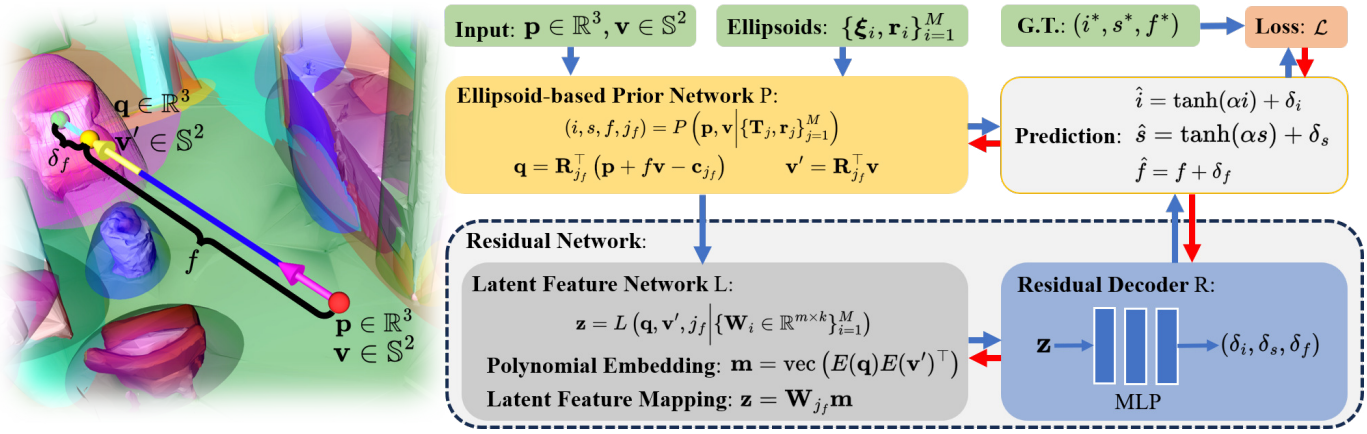


Fig. 3: **Method overview.** Given a query ray from position $\mathbf{p} \in \mathbb{R}^3$ in direction $\mathbf{v} \in \mathbb{S}^2$, an ellipsoid-based **P**rior network P uses M ellipsoids $\{\boldsymbol{\xi}_i, \mathbf{r}_i\}_{i=1}^M$ to learn the rough shape of the environment such that it can determine the closest ellipsoid intersected by the ray and predict an SDDF prior. Then, with the intersection point $\mathbf{q} \in \mathbb{R}^3$ and ray direction $\mathbf{v}' \in \mathbb{S}^2$ in the ellipsoid's local frame, a **L**atent network L generates a latent feature $\mathbf{z} \in \mathbb{R}^m$, which is decoded by the **R**esidual decoder R into residual predictions $(\delta_i, \delta_s, \delta_f)$, i.e. the difference between the ground truth and the prior. Finally, we compose the SDDF prediction as $\hat{f} = f + \delta_f$. Blue arrows show the data flow in the forward pass, while red arrows represent the backward pass.

input perturbations may lead to hitting a different obstacle surface, resulting in a significant change in SDDF value. The inherent discontinuity and the added input dimensions for the ray direction demand more training data over different ray directions. Explicit representations have advantages in handling such discontinuities with less data, but purely explicit representations struggle to achieve high-fidelity reconstruction and differentiable view synthesis, as they are discrete. In contrast, implicit representations are good at learning geometric details and allow differentiation.

Therefore, as shown in Fig. 3, our model combines both explicit and implicit representations. First, in Sec. IV, an explicit ellipsoid-based **P**rior network $P(\mathbf{p}, \mathbf{v})$ is introduced to predict a coarse SDDF prior $f(\mathbf{p}, \mathbf{v})$. Then, in Sec. V, we present a residual network consisting of a **L**atent feature network L and a **R**esidual decoder R that predicts an SDDF correction $\delta_f(\mathbf{p}, \mathbf{v})$, so that the combination accurately models the true SDDF $f^*(\mathbf{p}, \mathbf{v})$ as $\hat{f}(\mathbf{p}, \mathbf{v}) = f(\mathbf{p}, \mathbf{v}) + \delta_f(\mathbf{p}, \mathbf{v})$.

IV. Ellipsoid-based Prior Network

To take advantage of an explicit representation for occlusion modeling, we design an ellipsoid-based prior network P. The prior uses a set of ellipsoids to approximate the structure of the environment based on the range measurements and leaves the task of learning fine details to the residual network R.

A. SDDF of a Single Ellipsoid

First, for simplicity, consider a single ellipsoid given by:

$$\mathcal{E} = \{\mathbf{y} \in \mathbb{R}^3 \mid (\mathbf{y} - \mathbf{c})^\top \mathbf{R}\mathbf{Q}_0^{-2}\mathbf{R}^\top (\mathbf{y} - \mathbf{c}) \leq 1\}, \quad (5)$$

where $\mathbf{c} \in \mathbb{R}^3$ and $\mathbf{R} \in SO(3)$ are the position and orientation, $\mathbf{Q}_0 = \mathrm{diag}(\mathbf{r})$, and $\mathbf{r} \in \mathbb{R}^3_+$ are the radii of the ellipsoid.

We parameterize the ellipsoid pose as:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{c} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \mathbf{T}_0 \exp\left(\boldsymbol{\xi}^\wedge\right), \quad \boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\theta}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^\top & 0 \end{bmatrix}, \quad (6)$$

where $\mathbf{T}_0 \in SE(3)$ is initialized and fixed, and $\boldsymbol{\xi} = (\boldsymbol{\rho}, \boldsymbol{\theta}) \in \mathbb{R}^6$ is learnable. In (6), the function $\boldsymbol{\theta}^\wedge$ maps a vector $\boldsymbol{\theta} \in \mathbb{R}^3$ to a corresponding skew-symmetric matrix, and exp is the matrix exponential function.

To ensure that the radii $\mathbf{r} \in \mathbb{R}^3_+$, we parameterize it as $\mathbf{r} = \mathbf{r}_0 \exp(\mathbf{s})$, where $\mathbf{r}_0 \in \mathbb{R}^3_+$ is initialized and fixed, $\mathbf{s} \in \mathbb{R}^3$ is learnable, and exp is applied element-wise.

Then, we derive the SDDF of an ellipsoid $\mathcal{E}$ in closed form.

Proposition 2. *Consider a ray from position $\mathbf{p} \in \mathbb{R}^3$ in direction $\mathbf{v} \in \mathbb{S}^2$ and ellipsoid $\mathcal{E} \subset \mathbb{R}^3$. If the ray does not intersect $\mathcal{E}$, the ellipsoid SDDF is $f(\mathbf{p}, \mathbf{v}; \mathcal{E}) = \infty$. Otherwise:*

$$f(\mathbf{p}, \mathbf{v}; \mathcal{E}) = -\frac{\det \mathbf{Q}_0 \sqrt{i(\mathbf{p}, \mathbf{v})} + \mathbf{p}'^\top \mathbf{Q}_1^2 \mathbf{v}'}{\mathbf{v}'^\top \mathbf{Q}_1^2 \mathbf{v}'}, \quad (7)$$

where

$$i(\mathbf{p}, \mathbf{v}) = \mathbf{v}'^\top \mathbf{Q}_1^2 \mathbf{v}' - \mathbf{w}'^\top \mathbf{Q}_0^2 \mathbf{w}' \quad (8)$$

is an intersection indicator, $\mathbf{Q}_1 = \det(\mathbf{Q}_0)\mathbf{Q}_0^{-1}$, $\mathbf{p}' = \mathbf{R}^\top(\mathbf{p} - \mathbf{c})$, $\mathbf{v}' = \mathbf{R}^\top \mathbf{v}$, and $\mathbf{w}' = \mathbf{p}' \times \mathbf{v}'$.

The proof of Proposition 2 is provided in Supplemental II-A. The expression in (7) is one of the two solutions of the quadratic equation formed by combining the ray equation $\mathbf{q} = \mathbf{R}^\top(\mathbf{p} + f(\mathbf{p}, \mathbf{v}; \mathcal{E})\mathbf{v} - \mathbf{c})$ and the ellipsoid equation $\mathbf{q}^\top \mathbf{Q}_0^{-2} \mathbf{q} = 1$. It corresponds to the first intersection point along the ray direction, which aligns with the SDDF definition in Definition 1. The intersection indicator $i(\mathbf{p}, \mathbf{v})$ in (8) is positive when the line through $\mathbf{p}$ in direction $\mathbf{v}$ intersects $\mathcal{E}$ and negative otherwise. When the line is tangent to the ellipsoid, we have $i(\mathbf{p}, \mathbf{v}) = 0$. We also introduce a sign indicator:

$$s(\mathbf{p}, \mathbf{v}) = \mathbf{p}'^\top \mathbf{Q}_1^2 \mathbf{p}' - \det \mathbf{Q}_0^2, \quad (9)$$

which is negative when $\mathbf{p} \in \mathcal{E}$ and positive otherwise. We use the above equation instead of $\mathbf{p}'^\top \mathbf{Q}_0^{-2} \mathbf{p}' - 1$ as the sign indicator for better numerical stability.

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658

In order to specify a reasonable SDDF prior when the line does not intersect the ellipsoid, i.e., when $i(\mathbf{p},\mathbf{v}) < 0$, we change the SDDF from ∞ to:

$$f(\mathbf{p},\mathbf{v};\mathcal{E}) = -\frac{\mathbf{p}'^{\top}\mathbf{Q}_1^2\mathbf{v}'}{\mathbf{v}'^{\top}\mathbf{Q}_1^2\mathbf{v}'}. \tag{10}$$

The expression in (10) gives the distance from $\mathbf{p}'$ along $\mathbf{v}'$ to a virtual plane at the ellipsoid origin with normal vector $\mathbf{Q}_1^2\mathbf{v}'/\left\|\mathbf{Q}_1^2\mathbf{v}'\right\|_2$. This guarantees that $f(\mathbf{p},\mathbf{v};\mathcal{E})$ changes smoothly when $i(\mathbf{p},\mathbf{v})$ changes sign.

The expression in (7) is valid only when the ray from $\mathbf{p}$ in the direction of $\mathbf{v}$ intersects $\mathcal{E}$. If the ellipsoid $\mathcal{E}$ is behind the ray, i.e., $\mathbf{p} \notin \mathcal{E}$ and $s(\mathbf{p},\mathbf{v}) > 0$, then (7) is negative but $f(\mathbf{p},\mathbf{v};\mathcal{E}) = \infty$. We introduce a validity function $v(\mathbf{p},\mathbf{v})$, which is negative when $\mathcal{E}$ is behind the view ray. This allows us to combine (7) and (10) with consideration of validity to obtain a modified SDDF prior for a single ellipsoid:

$$f(\mathbf{p},\mathbf{v};\mathcal{E}) = \begin{cases} -\frac{\det\mathbf{Q}_0\sqrt{\beta}+\mathbf{p}'^{\top}\mathbf{Q}_1^2\mathbf{v}'}{\mathbf{v}'^{\top}\mathbf{Q}_1^2\mathbf{v}'}, & v(\mathbf{p},\mathbf{v}) \geq 0, \\ \infty, & v(\mathbf{p},\mathbf{v}) < 0, \end{cases} \tag{11}$$

where $\beta = \max(i(\mathbf{p},\mathbf{v}),0) + \epsilon$, $\epsilon > 0$ is a small value introduced for numerical stability of backward propagation, and the validity indicator function is defined as:

$$v(\mathbf{p},\mathbf{v}) = -\frac{\det\mathbf{Q}_0\sqrt{\beta}+\mathbf{p}'^{\top}\mathbf{Q}_1^2\mathbf{v}'}{\mathbf{v}'^{\top}\mathbf{Q}_1^2\mathbf{v}'}s(\mathbf{p},\mathbf{v}), \tag{12}$$

which requires that the sign of a valid SDDF and the sign of the indicator in (9) should agree. A 2D example in Fig. 4 shows that the ellipsoid SDDF prior $f(\mathbf{p},\mathbf{v};\mathcal{E})$ in (11) reflects Definition. 1 correctly.

The ellipsoid prior is also compatible with unsigned DDF, which is advantageous when dealing with non-watertight or thin objects where sign is ill-defined. In this case, one of the axis radii may be set to 0 so that the 3D ellipsoid becomes a 2D disk. Additional details are presented in Supplemental I.

B. Fusing Multiple Ellipsoid SDDFs

To model scenes with multiple objects at different locations, we consider a set of M ellipsoids $\mathcal{E}_j$ for $1 \leq j \leq M$. Sec. IV-A showed how to determine the SDDF $f_j(\mathbf{p},\mathbf{v})$, intersection indicator $i_j(\mathbf{p},\mathbf{v})$, and sign indicator $s_j(\mathbf{p},\mathbf{v})$ for a single ellipsoid in (11), (8) and (9), respectively. If $\mathbf{p} \in \mathcal{E}_j$, then $s_j(\mathbf{p},\mathbf{v}) \leq 0$. Hence, to determine whether $\mathbf{p}$ is contained in any ellipsoid, we can find the minimum $s_j(\mathbf{p},\mathbf{v})$. When a ray $(\mathbf{p},\mathbf{v})$ intersects an ellipsoid, we have $i_j(\mathbf{p},\mathbf{v}) \geq 0$. Hence, to determine whether any ellipsoid is intersected, we can find the maximum $i_j(\mathbf{p},\mathbf{v})$. Finally, the SDDF of a union of ellipsoids is equal to the minimum of the individual ellipsoid SDDFs but with the intersected ellipsoids prioritized. In summary, the intersection indicator, the sign indicator, and the SDDF value of a union of M ellipsoids are:

$$i(\mathbf{p},\mathbf{v};\cup_j\mathcal{E}_j) = \max_j i_j(\mathbf{p},\mathbf{v}), \tag{13}$$

$$s(\mathbf{p},\mathbf{v};\cup_j\mathcal{E}_j) = \min_j s_j(\mathbf{p},\mathbf{v}), \tag{14}$$

$$f(\mathbf{p},\mathbf{v};\cup_j\mathcal{E}_j) = \begin{cases} \min\limits_{j:i_j(\mathbf{p},\mathbf{v})\geq 0} f_j(\mathbf{p},\mathbf{v}), & \exists i_j(\mathbf{p},\mathbf{v}) \geq 0, \\ \min_j f_j(\mathbf{p},\mathbf{v}), & \text{otherwise}. \end{cases} \tag{15}$$

Thus, given M ellipsoids with pose and radii $\mathbf{T}_j$, $\mathbf{r}_j$ for $1 \leq j \leq M$, we define our ellipsoid-based prior network as:

$$(i,s,f,j_f) = P\left(\mathbf{p},\mathbf{v} \;\middle|\; \{\mathbf{T}_j,\mathbf{r}_j\}_{j=1}^{M}\right), \tag{16}$$

where j_f is the index of the ellipsoid selected by (15). This ellipsoid index is used to condition the latent feature computation, which will be described in Sec. V. We refer to P in (16) as a network because its outputs are differentiable with respect to $\mathbf{p}$, $\mathbf{v}$, $\mathbf{T}_j$, and $\mathbf{r}_j$. We provide the analytical gradients in Supplemental II-B.

Because our prior network uses ellipsoids to approximate SDDF, it satisfies the Eikonal equation in (4) by construction.

Proposition 3. *The SDDF $f(\mathbf{p},\mathbf{v};\cup_j\mathcal{E}_j)$ in (15) computed by the ellipsoid-based prior network satisfies the SDDF directional Eikonal equation in (4).*

We provide the proof of Proposition 3 in Supplemental II-C. Proposition 3 shows that the SDDF of a single ellipsoid satisfies the Eikonal equation, and the fusion operations in (13)–(15) preserve this property. The network P provides a coarse geometric prior but does not yield accurate predictions. We address this next.

V. Residual Network

In this section, we design a residual network to predict a correction term $\delta_f(\mathbf{p},\mathbf{v})$ so that the SDDF prediction of our combined prior and residual model, $\hat{f}(\mathbf{p},\mathbf{v}) = f(\mathbf{p},\mathbf{v}) + \delta_f(\mathbf{p},\mathbf{v})$, is accurate even for sets with complicated shapes. Intuitively, we expect the prior network P to learn the rough shape of the environment, encoded in $f(\mathbf{p},\mathbf{v})$, while the residual network R captures the details, encoded in $\delta_f(\mathbf{p},\mathbf{v})$.

A. Latent Feature Network

The residual network needs to estimate the SDDF correction term $\delta_f(\mathbf{p},\mathbf{v})$ based on the local shape of the surface that the ray $(\mathbf{p},\mathbf{v})$ interacts with. We convert $\mathbf{p}$ and $\mathbf{v}$ to the local coordinate frame of the first intersected ellipsoid and train a latent feature network whose output $\mathbf{z}$ is decoded into a SDDF correction by a residual decoder network $\delta_f = R(\mathbf{z})$.

Given a ray $(\mathbf{p},\mathbf{v})$, the prior network in (16) provides (i,s,f) and the index j_f of the selected ellipsoid. Using the pose of the j_f-th ellipsoid $\mathbf{R}_{j_f} \in SO(3)$, $\mathbf{c}_{j_f} \in \mathbb{R}^3$, we obtain the intersection point $\mathbf{q}$ in the ellipsoid frame:

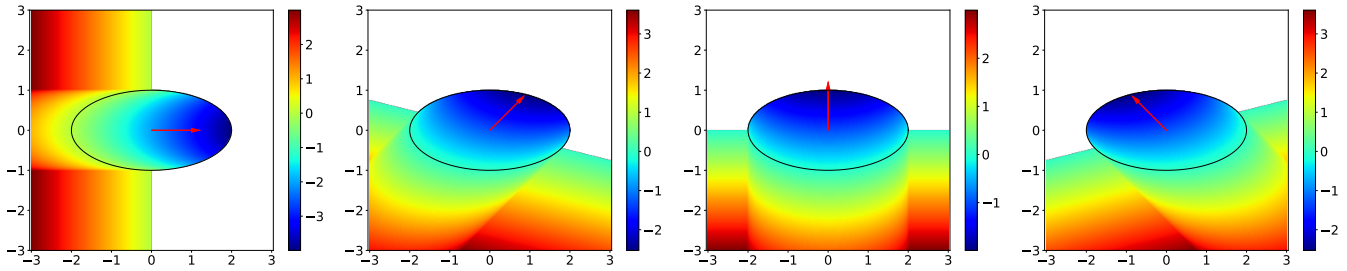
$$\mathbf{q} = \mathbf{R}_{j_f}^{\top}(\mathbf{p} + f\mathbf{v} - \mathbf{c}_{j_f}) = \mathbf{p}' + f\mathbf{v}'. \tag{17}$$

We train a latent feature network $\mathbf{z} = L(\mathbf{q},\mathbf{v}',j_f)$ with the intersection point $\mathbf{q}$, the local viewing direction $\mathbf{v}'$, and the ellipsoid index j_f as inputs and a latent feature $\mathbf{z}$ as output.

We observe that the ellipsoid SDDF in (7) can be interpreted as a nonlinear function of multiple multivariate polynomials involving $\mathbf{q}$ and $\mathbf{v}'$, where each variable's maximum degree is 2. While we use the ellipsoid prior network to exactly compute the SDDF value for ellipsoids, we hypothesize that the residual SDDF correction for complex shapes can also be approximated as a function of similar polynomial terms of $\mathbf{q}$ and $\mathbf{v}'$, which learns certain implicit features of second-order surfaces suitable for residual correction. Inspired by

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658



(a) 0° (b) 45° (c) 90° (d) 135°

Fig. 4: 2D visualization of the single ellipsoid SDDF $f(\mathbf{p}, \mathbf{v}; \mathcal{E})$ in (11) for fixed $\mathbf{v}$ and varying $\mathbf{p}$. According to (11), the SDDF prior for a single ellipsoid is finite when $\mathbf{p}$ is inside or when the ray intersects the ellipsoid or the virtual plane when $\mathbf{p}$ is outside the ellipsoid. Otherwise, the SDDF prior is infinite, indicating that the ellipsoid is behind the ray. The virtual plane is placed at the ellipsoid center with normal vector $\mathbf{Q}_1^2\mathbf{v}/\left\|\mathbf{Q}_1^2\mathbf{v}\right\|_2$, which varies with the viewing direction $\mathbf{v}$.

this observation, we design the network L to transform a polynomial embedding vector into a latent feature vector:

$$\mathbf{z} = L\left(\mathbf{q}, \mathbf{v}', j_f \mid \{\mathbf{W}_i\}_{i=1}^{M}\right) = \mathbf{W}_{j_f}\mathbf{m} \in \mathbb{R}^m, \tag{18}$$

$$\mathbf{m} = \mathrm{vec}\left(E(\mathbf{q})E(\mathbf{v}')^\top\right) \in \mathbb{R}^{100}, \tag{19}$$

$$E(\mathbf{p}) = \left[p_x^2, p_x p_y, p_x p_z, p_y^2, p_y p_z, p_z^2, p_x, p_y, p_z, 1\right]^\top, \tag{20}$$

where $E : \mathbb{R}^3 \to \mathbb{R}^{10}$ is a degree-2 monomial embedding, $\mathrm{vec}(\cdot)$ concatenates the columns of the input matrix, $\mathbf{m}$ is a vector of degree-2 monomials, and $\mathbf{W}_i \in \mathbb{R}^{m\times 100}$. The expression in (20) constructs a 10-dimensional vector of all degree-2 monomials of the input $\mathbf{q} \in \mathbb{R}^3$ and $\mathbf{v}' \in SO(3)$ correspondingly. Then, (19) uses the outer product of the two monomial vectors to generate a 100-dimensional vector of degree-2 monomials involving both $\mathbf{q}$ and $\mathbf{v}'$. Finally, (18) uses a learnable weight matrix $\mathbf{W}_{j_f}$ specific to the ellipsoid index j_f to transform the vector of monomials $\mathbf{m}$ into a latent feature vector $\mathbf{z} \in \mathbb{R}^m$.

B. Residual Decoder

The residual decoder is a multi-layer perceptron $R : \mathbb{R}^m \to \mathbb{R}^3$ that decodes the latent feature vector $\mathbf{z} \in \mathbb{R}^m$ into three residual predictions $(\delta_i, \delta_s, \delta_f)$. Then, the final predictions of the intersection indicator, sign indicator, and SDDF value are

$$\hat{i}(\mathbf{p}, \mathbf{v}) = \tanh\left(\alpha i(\mathbf{p}, \mathbf{v})\right) + \delta_i(\mathbf{p}, \mathbf{v}), \tag{21}$$

$$\hat{s}(\mathbf{p}, \mathbf{v}) = \tanh\left(\alpha s(\mathbf{p}, \mathbf{v})\right) + \delta_s(\mathbf{p}, \mathbf{v}), \tag{22}$$

$$\hat{f}(\mathbf{p}, \mathbf{v}) = f(\mathbf{p}, \mathbf{v}) + \delta_f(\mathbf{p}, \mathbf{v}), \tag{23}$$

where $\alpha > 0$ is a hyperparameter. Since the prior network produces $i(\mathbf{p}, \mathbf{v}) \in \mathbb{R}$ and $s(\mathbf{p}, \mathbf{v}) \in \mathbb{R}$ but, as described later in Sec. VI-A, we provide supervision $i^*(\mathbf{p}, \mathbf{v}) \in \{-1, 1\}$ and $s^*(\mathbf{p}, \mathbf{v}) \in \{-1, 1\}$ for (21) and (22), and use $\tanh$ to squash the output of the prior intersection indicator and sign indicator. The residuals δ_i and δ_s are applied after the squashing to prevent clipping their gradients during training.

Complementing Proposition 3, we show that the joint prior-residual SDDF prediction $\hat{f}$ in (23) still satisfies the SDDF directional Eikonal equation by construction.

Proposition 4. *The SDDF $\hat{f}(\mathbf{p}, \mathbf{v})$ in (23) computed by the combination of the prior and the residual networks satisfies the SDDF directional Eikonal equation in (4).*

The proof of Proposition 4 is presented in Supplemental III-A. The intuition behind the proof is that the input position $\mathbf{q}$, which is the intersection point between the ray and the ellipsoid, does not change when $\mathbf{p}$ moves along the ray direction $\mathbf{v}$ as long as the same ellipsoid is selected by the prior network. Therefore, the residual SDDF correction $\delta_f(\mathbf{p}, \mathbf{v})$ remains unchanged when $\mathbf{p}$ moves along $\mathbf{v}$, leading to a zero directional derivative along $\mathbf{v}$. Because our SDDF model satisfies the Eikonal equation by construction, we do not need an extra loss term to regularize the network and can use fewer parameters in the model, making it more efficient to train and evaluate.

VI. Training

A. Dataset Generation and Augmentation

We convert the sensor poses and range measurements $\{\mathbf{T}_t, \mathcal{Z}_t\}_{t=1}^{T}$, $\mathcal{Z}_t = \{\mathbf{v}_i, r_{t,i}\}_{i=1}^{N}$, described in Sec. III, into a dataset $\mathcal{D} = \{\mathbf{p}_j, \mathbf{v}_j, f_j^*, i_j^*, s_j^*\}_j$ suitable for training our SDDF model. Here, $\mathbf{p}_j \in \mathbb{R}^3$ is the origin of the ray, $\mathbf{v}_j \in \mathbb{S}^2$ is the direction of the ray provided in $\mathcal{Z}_t$, f_j^* is an SDDF measurement, $i_j^* \in \{-1, 1\}$ is an intersection indicator measurement, and $s_j^* \in \{-1, 1\}$ is a sign indicator measurement. From each ray measurement $\mathbf{v}_i, r_{t,i}$, obtained from sensor position $\mathbf{p}_t$ and orientation $\mathbf{R}_t$, we generate a corresponding data sample as $\mathbf{p}_j = \mathbf{p}_t$, $\mathbf{v}_j = \mathbf{R}_t\mathbf{v}_i$, $f_j^* = r_{t,i}$, $i_j^* = 1$, $s_j^* = 1$. These data are unbalanced in the sign indicator and the SDDF sign because all the samples have $s_j^* = 1$ and $f_j^* \geq 0$. Therefore, we need to augment the data with negative samples. For each sample $(\mathbf{p}_j, \mathbf{v}_j, , f_j^*, i_j^*, s_j^*)$, we generate a corresponding negative sample $(\mathbf{p}', \mathbf{v}_j, -\epsilon, 1, -1)$, where we extend the ray to a point $\mathbf{p}' = \mathbf{p}_j + (f_j^* + \epsilon)\mathbf{v}_j$ slightly behind the observed surface with a small offset $\epsilon > 0$. Although the intersection indicator i_j^* is also unbalanced, it is inefficient to generate non-intersecting rays, and our experiments show that augmenting the intersection indicator does not improve our reconstruction results.

B. Ellipsoid Initialization

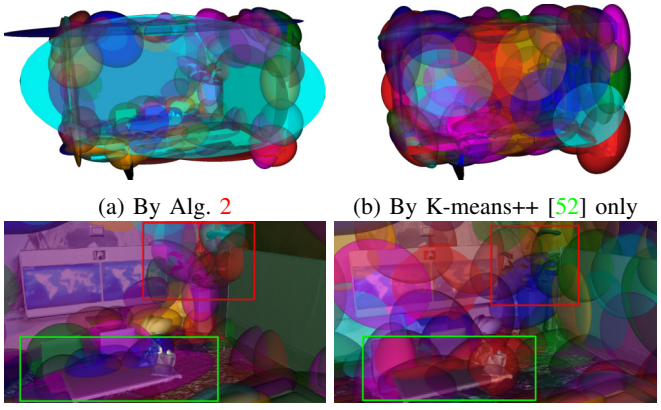
We present an initialization strategy for the M ellipsoids, needed by the ellipsoid-based prior network P in Sec. IV.

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658

Algorithm 1 Single Ellipsoid Initialization

procedure SINGLEELLIPSOIDINIT($\mathcal{X} = \{\mathbf{x}_k\}_{k=1}^K$)
  $\mathbf{c} \leftarrow \frac{1}{K}\sum_{k=1}^K \mathbf{x}_i$ ▷ center
  $\mathbf{X} \leftarrow \begin{bmatrix}\mathbf{x}_1 \cdots \mathbf{x}_K\end{bmatrix}^\top - \mathbf{1}\mathbf{c}^\top$
  Eigen decomposition on $\frac{1}{K}\mathbf{X}^\top\mathbf{X} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$
  $\mathbf{R} \leftarrow \mathbf{Q}\,\mathrm{diag}\left(\begin{bmatrix}1 \cdots \det \mathbf{Q}\end{bmatrix}\right)$ ▷ rotation
  $\begin{bmatrix}\lambda_1 \cdots \lambda_n\end{bmatrix} \leftarrow \mathrm{diag}\,(\mathbf{\Lambda})$ ▷ n is the space dimension
  $r_i \leftarrow \max\left(r_{\min}, \alpha\sqrt{|\lambda_i|}\right)$ ▷ radii
  return $\mathbf{R}, \mathbf{c}, \mathbf{r}$



(a) By Alg. 2 (b) By K-means++ [52] only

(c) Closer view of (a) (d) Closer view of (b)

Fig. 5: Comparison of ellipsoid initialization algorithms. The left column is generated by Alg. 2 and the right column is by the K-means++ [52] algorithm only (i.e. L3 to L5 of Alg. 2). (a) Using Alg. 2, a few ellipsoids are used to approximate planar surfaces like ceiling, wall, and ground. (b) Using K-means++ [52] only, too many ellipsoids are used to approximate planar surfaces. (c) and (d) show close-ups of the indoor objects. The table (in the green box) and the plant (in the red box) are better approximated by ellipsoids from Alg. 2.

Using the augmented dataset $\mathcal{D}$ constructed above, we obtain a point cloud:

$$\mathcal{X} = \{\mathbf{x}_k\}_{k=1}^K = \{\mathbf{p}_j + f_j^*\mathbf{v}_j \mid f_j^* \geq 0\}_j \cup \{\mathbf{p}_j \mid f_j^* < 0\}_j,$$

which is the collection of surface points and points inside obstacles. Leaving the number of ellipsoids M as a hyperparameter, we use K-means++ [52] to divide $\mathcal{X}$ into M clusters $\{\mathcal{X}_m\}_{m=1}^M$. For each cluster, we initialize an ellipsoid using principal component analysis of the points in the cluster as shown in Alg. 1. In practice, we set a minimum allowed radius $r_{\min} = 0.005$ to avoid numerical problems and scale the ellipsoids by $\alpha = 3$ to ensure that the points $\mathcal{X}_m$ are mostly covered by the m-th ellipsoid.

However, in typical indoor environments, as shown in Fig. 5, this approach allocates too many ellipsoids to planar surfaces, which could be fit by a single flat ellipsoid. Hence, we propose Alg. 2 to create a better multi-ellipsoid initialization. After using K-means++ [52] to divide $\mathcal{X}$ into M clusters and create an ellipsoid $(\mathbf{R}_m, \mathbf{c}_m, \mathbf{r}_m)$ for each cluster m by Alg. 1, we build an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node is a flat ellipsoid $i \in \mathcal{V}$ (i.e. the mean projection β of $\mathbf{x}_k \in \mathcal{X}_m$ to the shortest axis is smaller than threshold $\beta_{\max}$), and

Algorithm 2 Multi-Ellipsoid Initialization

procedure MULTIELLIPSOIDINIT($\mathcal{X} = \{\mathbf{x}_k\}_{k=1}^K, M, S$)
  ▷ S is the number of ellipsoid neighbors
  $\{\mathcal{X}_m\}_{m=1}^M \leftarrow$ K-MEANS++($\mathcal{X}, M$)
  for $m = 1 \cdots M$ **do**
    $\mathbf{R}_m, \mathbf{c}_m, \mathbf{r}_m \leftarrow$SINGLEELLIPSOIDINIT($\mathcal{X}_m$)
    $j \leftarrow \arg\min_{1 \leq j \leq n} r_{m,j}$
    $\mathbf{n}_m \leftarrow \mathbf{R}_m\mathbf{e}_j$
    $\beta_m \leftarrow \frac{1}{|\mathcal{X}_m|}\sum_{\mathbf{p}\in\mathcal{X}_m} \mathbf{n}_m^\top(\mathbf{p} - \mathbf{c}_m)$
  $\mathcal{V} \leftarrow \emptyset, \mathcal{E} \leftarrow \emptyset$ ▷ undirected graph
  for $i = 1 \cdots M, \beta_i < \beta_{\max}$ **do** ▷ flat ellipsoid
    $\mathcal{K}_i \leftarrow$ K-NEARESTNEIGHBOR($\{\mathbf{c}_m\}_{m=1}^M, i, S$)
    for $j \in \mathcal{K}_i, \beta_i < \beta_{\max}$ **do**
      $\eta \leftarrow \frac{1}{2}\left(|(\mathbf{c}_i - \mathbf{c}_j)^\top\mathbf{n}_i| + |(\mathbf{c}_i - \mathbf{c}_j)^\top\mathbf{n}_j|\right)$
      if $\eta < \eta_{\max}$ **then** ▷ i and j are coplanar
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{i, j\}$ and $\mathcal{E} \leftarrow \mathcal{E} \cup \{(i, j)\}$
  $\{\mathcal{V}_i\}_{i=1}^C \leftarrow$ FINDCONNECTEDCOMPONENTS($\mathcal{V}, \mathcal{E}$)
  $\mathcal{R} \leftarrow \emptyset, \mathcal{X}' \leftarrow \emptyset$
  for $i = 1 \cdots C$ **do** ▷ merge ellipsoids
    $\mathbf{R}, \mathbf{c}, \mathbf{r} \leftarrow$SINGLEELLIPSOIDINIT($\bigcup_{m\in\mathcal{V}_i}\mathcal{X}_m$)
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{(\mathbf{R}, \mathbf{c}, \mathbf{r})\}$
    $\mathcal{X}' \leftarrow \mathcal{X}' \cup \bigcup_{m\in\mathcal{V}_i}\mathcal{X}_m$
  $\{\mathcal{X}'_m\}_{m=1}^{M-C} \leftarrow$ K-MEANS++($\mathcal{X} - \mathcal{X}', M - C$)
  for $m = 1 \cdots M - C$ **do** ▷ initialize other ellipsoids
    $\mathbf{R}, \mathbf{c}, \mathbf{r} \leftarrow$SINGLEELLIPSOIDINIT($\mathcal{X}'_m$)
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{(\mathbf{R}, \mathbf{c}, \mathbf{r})\}$
  return $\mathcal{R}$ ▷ M ellipsoid poses and radii

two nodes (i, j) are connected only when they are coplanar neighbors. Then, we find the number of ellipsoids, C, for these planar surfaces by finding the connected components $\{\mathcal{V}_i\}_{i=1}^C$ of $\mathcal{G}$. We merge the ellipsoids in each component $\mathcal{V}_i$ running Alg. 1 on $\bigcup_{m\in\mathcal{V}_i}\mathcal{X}_m$. Finally, the remaining points $\bigcup_{m\notin\mathcal{V}}\mathcal{X}_m$ are divided into $M-C$ clusters by K-means++ [52] and the ellipsoid of each cluster is initialized by Alg. 1. As shown in Fig. 5, the planar surfaces are correctly detected and approximated by only a few ellipsoids, leaving more ellipsoids for other more complex shapes.

C. Loss Function for Ellipsoid-based Prior Network

Given a prediction $\mathbf{x} = (i, s, f)$ from the ellipsoid-based prior network P and the corresponding label $\mathbf{x}^* = (i^*, s^*, f^*)$ from the measurement data $\mathcal{D}$, we compute a weighted sum of Huber losses:

$$\begin{aligned} l_0(x, y) &= \begin{cases} 0.5(x-y)^2, & \text{if } |x-y| < 1, \\ |x-y| - 0.5, & \text{otherwise,} \end{cases} \\ l(x, y) &= \left(w^+\mathbb{1}(y \geq 0) + w^-\mathbb{1}(y < 0)\right) l_0(x, y), \\ \mathcal{L}_P(\mathbf{x}, \mathbf{x}^*) &= l(i, i^*) + l(s, s^*) + l(f, f^*). \end{aligned} \tag{24}$$

The role of P is to make the ellipsoids cover the objects in the scene so that the occlusions between objects are captured. Thus, larger weights are used for negative f^* and negative s^*: $w_f^- = 1.65$ and $w_s^- = 10$. The other weights are set to 1.

D. Loss Function for Residual Network

The residual network is trained using the same loss function as the prior network in (24) but with different weights. Unlike the prior network, the residual network focuses on learning

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658

the geometric details. Thus, higher weights are used for the SDDF prediction $\hat{f}$. Given the prediction $\hat{\mathbf{x}} = (\hat{i}, \hat{s}, \hat{f})$ and the corresponding label $\mathbf{x}^* = (i^*, s^*, f^*)$ from the measurement data $\mathcal{D}$, we have

$$\begin{aligned} \mathcal{L}_R(\hat{\mathbf{x}}, \mathbf{x}^*) &= l(\hat{i}, i^*) + l(\hat{s}, s^*) + l(\hat{f}, f^*), \\ \mathcal{L}(\hat{\mathbf{x}}, \mathbf{x}, \mathbf{x}^*) &= \mathcal{L}_P(\mathbf{x}, \mathbf{x}^*) + \mathcal{L}_R(\hat{\mathbf{x}}, \mathbf{x}^*), \end{aligned} \tag{25}$$

where for $\mathcal{L}_R$, $w^+_{\hat{f}} = 1.0, w^-_{\hat{f}} = 1.1$, and other weights 0.1.

VII. Application to Viewpoint Optimization

Our SDDF model is differentiable and, hence, enables continuous viewpoint optimization. This property is useful for several applications, e.g., to guide navigation in virtual reality or to enable a robot to explore an unknown environment. We first consider determining the next-best view and then scale up to optimization of a trajectory of several views. Our SDDF model can predict a point cloud measurement from any desired sensor pose $(\mathbf{p}_t, \mathbf{R}_t)$ as:

$$\mathcal{P}_t = \{\mathbf{p}_t + \hat{f}_i \mathbf{R}_t \mathbf{v}_i\}_{i=1}^N, \tag{26}$$

where $\{\mathbf{v}_i\}_{i=1}^N$ are the ray directions in the sensor frame and $\hat{f}_i$ are the SDDF predictions for each ray. The utility of a point-cloud measurement for the purpose of exploration or environment coverage can be evaluated in terms of the visible region volume. We use the following loss to measure the (negative) size of the visible volume:

$$\mathcal{L}_v\left(\{\hat{f}_i\}_{i=1}^N\right) = -\frac{1}{2N}\sum_{i=1}^{N}\left(\max\{\hat{f}_i, 0\}\right)^2. \tag{27}$$

Moreover, consecutive measurements $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$ should have a small overlap in order to observe a large area. Therefore, we design the following overlap loss:

$$\mathcal{L}_o\left(\mathcal{P}_t, \mathcal{P}_{t+1}\right) = -\frac{\sum_{\mathbf{p}\in\mathcal{P}_t, \mathbf{q}\in\mathcal{P}_{t+1}} \min\{\|\mathbf{p}-\mathbf{q}\|_2, d_{\max}\}}{|\mathcal{P}_t||\mathcal{P}_{t+1}|}, \tag{28}$$

where $d_{\max} > 0$ is a distance threshold of no overlap.

In practice, we must also ensure that the sensor is not in collision with any obstacles. To do so, we introduce a set of risk detection rays, which are uniformly sampled from the sphere that contains the robot, and obtain their SDDF predictions $\{\hat{f}^r_i\}_{i=1}^M$. The risk loss is defined as:

$$\mathcal{L}_r\left(\{\hat{f}^r_i\}_{i=1}^M\right) = \frac{1}{M}\sum_{i=1}^{M}\max\{d_{\text{safe}} - \hat{f}^r_i, 0\}, \tag{29}$$

where $d_{\text{safe}} > 0$ is a safe distance threshold, chosen such that $\hat{f}^r_i < d_{\text{safe}}$ implies a potential collision.

We optimize the camera pose $(\mathbf{p}_{t+1}, \mathbf{R}_{t+1})$ at time $t+1$ by minimizing the weighted sum $\mathcal{L} = w_o\mathcal{L}_o + w_v\mathcal{L}_v + w_r\mathcal{L}_r$, where w_o, w_v, and w_r are weights.

Considering multi-view trajectory optimization, it is inefficient to optimize every pose in a continuous trajectory because two views that are close to each other are likely to overlap significantly, i.e., with minimal change in the viewpoint. Therefore, we consider optimizing certain waypoints on the trajectory. We incrementally optimize n poses $\{\mathbf{p}_i, \mathbf{R}_i\}_{i=0}^n$ generated by an off-the-shelf planning algorithm, such as RRT* [53], such that for each pose $\{\mathbf{p}_i, \mathbf{R}_i\}_{i>0}$, we optimize the loss:

$$\mathcal{L}' = w_o\mathcal{L}_o\left(\mathcal{P}_0 \bigcup_{j=1}^{j<i} \mathcal{P}'_j, \mathcal{P}_i\right) + w_v\mathcal{L}_v + w_r\mathcal{L}_r, \tag{30}$$

where $\mathcal{P}'_j$ is the predicted point cloud at the optimized waypoint $(\mathbf{p}'_j, \mathbf{R}'_j)$. During the optimization, we downsample $\mathcal{P}_0 \bigcup_{j=1}^{j<i} \mathcal{P}'_j$ with a stride of $i-j$, labeled as $\tilde{\mathcal{P}}_j$, such that $\tilde{\mathcal{P}}_j$ has a constant size. This incremental optimization strategy provides two benefits. First, it reduces the use of GPU memory and makes the along-trajectory multi-view optimization problem solvable. Second, it allows to parallelize the trajectory optimization and execution.

VIII. Evaluation

We evaluate our SDDF model in terms of accuracy, computational efficiency, and usefulness for viewpoint optimization. First, we compare against three state-of-the-art baselines on synthetic and real datasets. Then, an ablation study examines the effectiveness of each module in our model by varying the number of ellipsoids, data augmentation and initialization strategies, and the use of polynomial embedding. Finally, we demonstrate the utility of SDDF in viewpoint optimization.

A. Comparison with Baselines

Baselines: We compare our method against three baselines: Nerfacto (with and without depth loss, check Supplemental IV-B for details) [19], RaDe-GS [1], and SDF-Instant-NGP [2]. These methods were not initially designed for SDDF prediction but can be used to predict SDDF as follows. For SDF, we implement sphere tracing [14] to find the closest point on the surface along the query direction. For Nerfacto and RaDe-GS, we render a depth image at the query viewpoint, project the depth image to a point cloud, and compute the distance. We compare the mean absolute error (MAE) in SDDF prediction and the computational cost in terms of GPU memory, model size, and inference time.

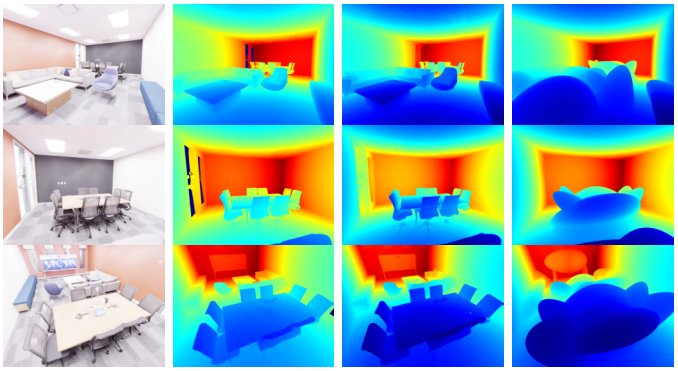
Datasets: A total of 15 datasets (14 synthesized and 1 real) are used for comparison. We obtained data from six scenes from Replica ("Hotel" and "Office 0-5") [54], the Allensville scene from Gibson [18], and scene 0000-00 from ScanNet [55].

From the seven synthetic scenes, we synthesize two sensor types, LiDAR and RGB-D, for a total of 14 datasets. For both sensor types, measurements are synthesized from a grid of free space positions in the scene. LiDAR data is synthesized with 360 horizontal and 180 vertical angular increments, ranging between $[-\pi, \pi) \times [-\pi/2, \pi/2]$. RGB-D data is synthesized from six uniform camera orientations, with an image resolution of 640×480, and horizontal and vertical fields of view of $94° \times 77°$. For each scene, we sample 20 random test viewpoints (40 for Allensville), which were visually confirmed to be representative of possible camera poses in real applications.

ScanNet [55] provides real RGB-D images captured along a trajectory. During training, the RGB images are linearly interpolated to align with the depth images of resolution

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658



(a) Scene (b) SDDF G.T. (c) Prediction (d) SDDF Prior

Fig. 6: SDDF prediction by our method. Columns a) and b) show RGB images and ground-truth SDDF. As shown in c), our model remains accurate with varying view distance due to satisfying the directional Eikonal constraint. As shown in d), the ellipsoidal prior provides a coarse approximation of the shape of objects in the scene, such as walls, chairs, and tables to be used by the residual network.

640×480. We found that the quality of the reference mesh from ScanNet is insufficient for generating SDDF ground truths; thus, we used this dataset for qualitative comparison only. The views for comparison are generated by randomly selecting and perturbing waypoints of the trajectory ($\mathcal{U}_{[-0.2,0.2]}$ for each dimension of the translation and $\mathcal{U}_{[-\pi,\pi]}$ for the yaw angle) so that the rendered views are representative.

Experiment Setup: The LiDAR and RGB-D datasets are used as follows. For our method, we consider both the LiDAR and RGB-D cases. When using the RGB-D datasets with our method, we only use the depth images and apply $20\times$ downsampling ($5\times$ for Replica Hotel) to achieve a similar ray density as the LiDAR dataset. Similarly, for SDF-Instant-NGP, we consider both LiDAR and RGB-D datasets and use all range measurements in either the LiDAR or depth images, along with additional data augmentations of near-surface samples and free-space samples necessary for stable training [2]. For Nerfacto and RaDe-GS, we only consider the RGB-D case, as they do not support rendering LiDAR data. RaDe-GS is trained with and without RGB (i.e., either depth-only or RGB-D), whereas Nerfacto is only trained with RGB-D, as RGB is necessary for training Nerfacto. RaDe-GS is trained with $2\times$ downsampling due to GPU memory limitations, while Nerfacto is trained using the full dataset.

Our method uses 128 ellipsoids (256 for Gibson Allensville) to fit an SDDF prior of the scene. In contrast, RaDe-GS uses 5% of the range measurements (3% for Allensville due to GPU memory limits) to initialize one Gaussian for each measurement, which yields many more Gaussians than our ellipsoids. Further training details are provided in Supplemental IV.

Accuracy of SDDF Predictions: We first compare the SDDF predictions made using different methods. Qualitative results are shown in Figs. 6c and 7 (more results in Figs. 12-14 in Supplemental V-A). Our SDDF model accurately learns the geometry of the scene, as shown in the three qualitative examples in Fig. 6. The overall model remains robust to varying viewing distances because the latent feature for the residual network is constructed from the intersection point and the viewing direction in a way that satisfies the Eikonal constraint (Proposition 4). The ellipsoidal prior (Fig. 6d) is optimized to approximate the shapes of objects in the scene, based on which the residual network recovers fine details with high fidelity, as seen in Fig. 6c. This ellipsoid-based network design allows readily combining different sensor measurements of the same surface into a neural representation.

Fig. 7 shows qualitative comparisons against RaDe-GS [1] (RGB-D and depth-only), Nerfacto [19], and SDF-Instant-NGP [2] with sphere tracing. In Fig. 7c, RaDe-GS exhibits erroneous artifacts around the toilet, at the corner, or near the fridge, where there are limited RGB-D observations. Meanwhile, our method accurately reconstructs these areas. Nerfacto also exhibits artifacts due to insufficient data as shown in Fig. 7e. Since no explicit representations like our ellipsoids or the Gaussians in RaDe-GS are used, Nerfacto predicts SDDF based on the learned volume density, which is optimized for photometric rendering rather than scene geometry, leading to large distance prediction errors.

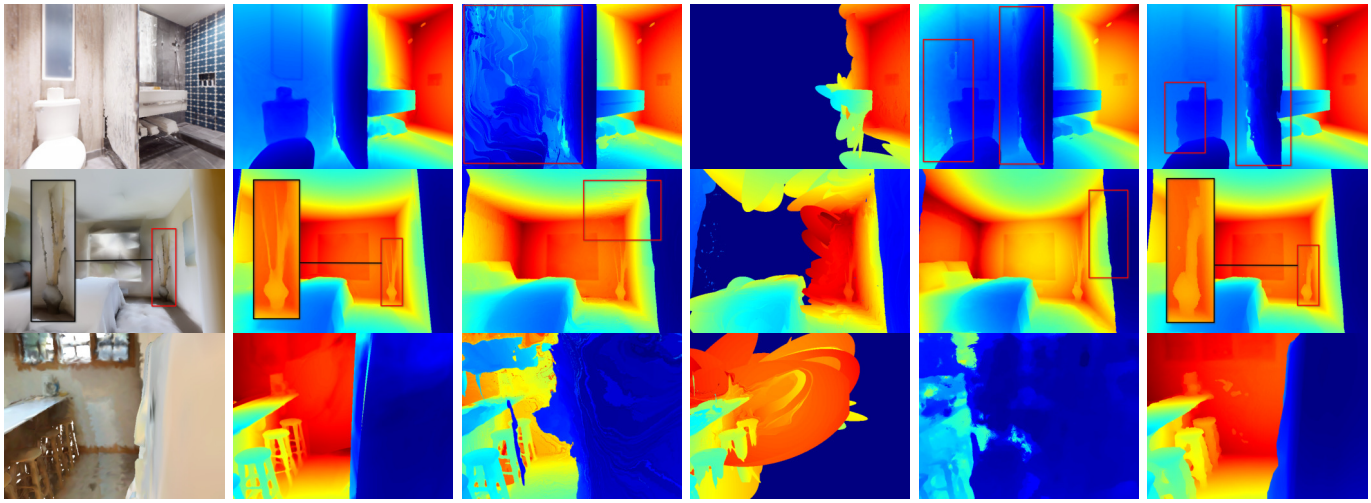
Meanwhile, sphere tracing on SDF-Instant-NGP [2] does not exhibit significant artifacts. However, this baseline tends to learn smoother shapes that lack sharper details, such as the plant shown in the second row or the chairs in the third row of Fig. 7f. Moreover, the first row of Fig. 7f shows that the errors accumulated during sphere tracing become significant at boundaries when the SDF model does not predict accurate SDF consistently. These qualitative observations are consistent in *real* ScanNet data [55], as shown in Fig. 7 and Fig. 14 in Supplemental V-A.

The quantitative results in Table I show that our method reaches the state of the art for both LiDAR and RGB-D datasets. Our method has slightly higher errors on RGB-D datasets because our method only uses depth information from the RGB-D datasets, whereas RaDe-GS uses both RGB and depth data. Moreover, our method is trained with $20\times$ downsampling, whereas RaDe-GS uses $2\times$ downsampling and hence approximately $10\times$ more depth data. Consequently, RaDe-GS uses a significantly higher number of parameters as seen in Table II. Nerfacto and SDF-Instant-NGP do not downsample the data, but generally perform worse than ours. Training with depth loss [56] does not improve the distance predictions by Nerfacto. We examine the results of Nerfacto and depth-Nerfacto further in Supplemental IV-B. With RGB-D datasets, RaDe-GS [1] has the lowest MAE among models that use both RGB and depth. However, as shown in Table I and in Fig. 7d, when RaDe-GS is trained using depth data only, it shows significantly larger errors in larger and more complicated scenes. We omit quantitative evaluation on ScanNet data (Fig. 14 in Supplemental V-A), due to the unavailability of a reliable ground-truth mesh.

Computational Efficiency: Table II shows the computational cost of all methods, in terms of training time, inference time, number of parameters and maximum GPU memory usage during training and testing. Our method exhibits the lowest memory usage for training and testing, and the second lowest number of parameters after SDF-Instant-NGP [2]. This

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658



(a) Scene (b) SDDF (ours) (c) RaDe-GS [1] (d) w/o RGB [1] (e) Nerfacto [19] (f) SDF [2]

Fig. 7: Qualitative comparison of SDDF predictions. Row 1: Replica Hotel (synthesized). Row 2: Gibson Allensville (synthesized). Row 3: ScanNet scene 0000-00 (real). In areas with limited sensor measurements, RaDe-GS [1] fails to learn the geometry, with RGB (c) or without RGB (d), yielding artifacts. Nerfacto [19] in e) shows significant artifacts and large distance prediction error. SDF-Instant-NGP [2] in f) tends to learn a smoother approximation, missing sharp boundaries.

TABLE I: Mean absolute error (cm) of SDDF prediction

		Allensville	Hotel	Office 0	Office 1	Office 2	Office 3	Office 4
LiDAR	SDF-Instant-NGP [2]	**1.137**	1.224	**0.825**	0.724	1.342	1.608	**1.037**
	SDDF (ours)	1.350	**0.997**	1.092	**0.694**	**1.236**	**1.588**	1.132
RGB-D	RaDe-GS [1]	**1.737**	**0.857**	**0.438**	**0.348**	**1.258**	**0.827**	**0.498**
	Nerfacto [19]	83.433	58.272	63.011	68.382	69.144	74.168	88.479
	depth-Nerfacto [19]	85.655	61.706	65.074	69.767	74.422	79.227	76.535
Depth Only	RaDe-GS (w/o RGB) [1]	25.002	45.948	1.208	6.654	3.582	**0.955**	**0.500**
	SDF-Instant-NGP [2]	**1.106**	1.297	**0.737**	0.711	1.120	1.395	0.953
	SDDF (ours)	1.490	**1.247**	1.120	**0.696**	**1.081**	1.568	1.206

TABLE II: Max. GPU memory usage, model size, and inference time on Replica-Hotel dataset [54] (Intel 14900K CPU and NVIDIA RTX-3090 GPU). Training time is the time took by the model to reach MAE < 1.5 cm.

	SDDF	RaDe-GS	Nerfacto	SDF
Training GRAM (G)	**3.3**	18.7	4.7	6.2
Testing GRAM (G)	**1.7**	6.7	3.6	1.8
No. Parameters (M)	2.7	113.7	16.4	**1.7**
Training Time (min)	100	30	N/A	**3**
Inference Time (ms/frame)	69	**7**	254	103

is because we use much fewer ellipsoids than Gaussians in RaDe-GS [1], which has the highest GPU memory usage. The lower GPU memory usage of our method is beneficial for scaling up to larger scenes, or deployment on mobile robots with limited hardware resources. Although Nerfacto has relatively low GPU memory usage, it has the longest inference time, and its prediction error is much higher than other methods.

In terms of inference time, our method is the second best, with RaDe-GS significantly outperforming our method. We attribute this to the highly optimized, full-CUDA implementation of RaDe-GS, and expect that it is possible to achieve similar performance with improvements in the implementation. Our method is 1.5× faster to infer the directional distance than the SDF-Instant-NGP-based sphere-tracing method, and 4× faster than Nerfacto. This is because our method only requires a single forward pass, whereas the other methods require multiple passes. It is also worth noting that our method is faster than SDF-Instant-NGP despite their CUDA optimized implementation. However, because of the extra direction input, our model needs more time to train, which is a future direction for improvement.

B. Ablation Study

We examine the performance of our model in four variations. We first consider using twice or half the number of ellipsoids for each scene. To investigate the contribution of negative sample augmentation, we compare against an SDDF model trained without negative samples. In addition, we train SDDF models with ellipsoids initialized by K-means++ [52] only. To demonstrate the benefit of our polynomial embedding, we compare it against Fourier embedding [57]:

$$\mathbf{m} = [\sin(2\pi \mathbf{B_p p}), \cos(2\pi \mathbf{B_p p}), \sin(2\pi \mathbf{B_v v}), \cos(2\pi \mathbf{B_v v})], \tag{31}$$

where $\mathbf{B_p}, \mathbf{B_v} \in \mathbb{R}^{25\times 3}$ are random Fourier features sampled from $\mathcal{N}(0, 1)$. We use (31) in place of (20) and (19), while (18) is kept so that the model size remains unchanged. The results are summarized in Table III. Lastly, we illustrate the advantages of our SDDF definition compared to related formulations [15, 16].

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658

TABLE III: Ablation experiments on LiDAR datasets. Mean absolute error (cm) of SDDF prediction is reported.

	Allensville	Hotel	Office 0	Office 1	Office 2	Office 3	Office 4
twice number of ellipsoids	1.414	1.013	**1.034**	0.716	1.291	1.649	1.250
half number of ellipsoids	1.477	1.013	1.072	0.706	**1.218**	1.727	1.192
without negative samples	5.379	2.639	1.954	0.867	2.734	8.598	2.406
K-means++ [52] only ellipsoid initialization	1.405	1.033	1.060	0.703	1.279	1.636	1.248
Fourier embedding	1.842	1.303	1.544	0.930	1.771	2.513	1.631
default	**1.350**	**0.997**	1.092	**0.694**	1.236	**1.588**	**1.132**

Table III shows that although our method is robust to the number of ellipsoids, using too few or too many ellipsoids causes problems. With too few ellipsoids, the model cannot approximate the shape of objects in the scene well, whereas with too many ellipsoids, the model learns spurious occlusions between the ellipsoids. During testing, we also found that too many ellipsoids cause the optimizer to move some redundant ellipsoids out of the scene.

Negative SDDF sample augmentation plays a key role in optimizing ellipsoids. Fig. 8 shows that with negative SDDF samples, the ellipsoid priors are optimized to cover the scene well. In contrast, without negative samples, the resulting ellipsoids are generally smaller and regress into the interior of objects, which leads to missing ray intersections. Missing ray intersections cause a wrong ellipsoid to be selected at a further location, leaving the burden to the residual decoder R to learn larger residuals.

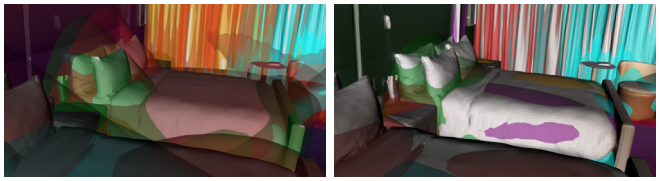
Appropriate ellipsoid initialization is also important. As shown in Table III, initializing the ellipsoids solely with K-means++ [52] causes larger errors, as K-means++ generally uses an excessive number of ellipsoids for planar surfaces.

We also consider using the Fourier embedding [57] in place of the polynomial embedding introduced in this work. The Fourier embedding may seem useful because it can learn high-frequency features. However, the results in Table III show that polynomial embedding provides better generalization for different scenes with smaller errors. Our investigation indicates that this is because the Fourier embedding requires different spatial frequency hyperparameters for each scene to achieve comparable performance, whereas the polynomial embedding does not require such hyperparameter tuning.

Direct comparison against DDF [16] was not possible due to the unavailability of source code, while the SDDF method of [15] can handle only object-level reconstruction. We illustrate the advantage of our SDDF formulation in a simple 2D scene in Fig. 9. The first row shows that our SDDF definition has fewer discontinuities than DDF [16], while the SDDF of [15] fails to capture the two circles due to its object-level representation of the box. The discontinuities in DDF [16] translate into a larger learning error than our SDDF, as we demonstrate in the second row of Fig. 9 using an MLP model with hidden dimensions $[16, 8, 8, 1]$. Faster convergence to lower error in both training and validation with the same model indicates that our SDDF formulation is more amenable to learning than DDF.

C. Application to Viewpoint Optimization

The first row of Fig. 10 demonstrates gradient-based next-best view optimization using our SDDF model. The second row shows the combined surface area observed by the two camera views and the overlapped area between the two views. The visualization shows that our method can reduce the overlap between the two camera views and increase the observed area. In this example, the area observed by the two camera views increases to $36.15m^2$ $(+90.3\%)$ from $19.00m^2$ with the initial viewpoints. In the third row of Fig. 10, we show an example of scaling up to a trajectory. By optimizing the waypoints, the observed area increases from $124.55m^2$ to $177.50m^2$ $(+37.7\%)$. We show more results of viewpoint optimization in different scenes in Supplemental V-B. These results illustrate a capability that SDDF enables for the first time as a proof of concept. While the same trajectory optimization can be done with SDF and differentiable sphere tracing, this will be much slower and less stable due to the multiple sphere tracing iterations.



(a) Default (b) W/o negative samples

Fig. 8: Ablation study of negative samples augmentation. With negative sample augmentation in a), the ellipsoids cover the objects in the scene. Without negative sample augmentation (b), the objects are not covered by ellipsoids in many places.

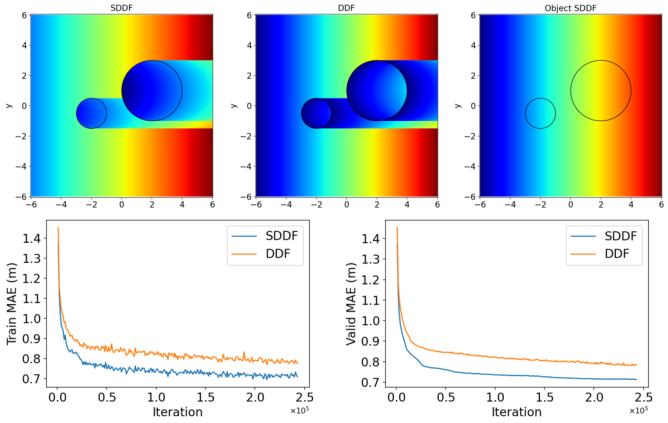
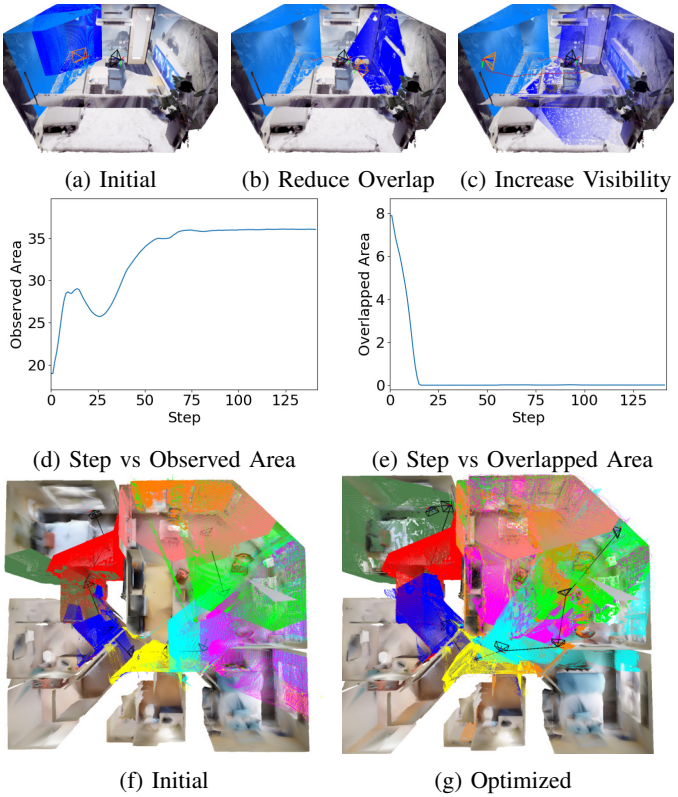


Fig. 9: Comparison of SDDF against DDF [16] and object SDDF [15] in a 2D scene with two circles in a square box. Top row: ground truth targets with a fixed viewing direction $\mathbf{v} = (-1, 0)$. Bottom row: training and validation MAE (m) when learning using an MLP.

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658



(a) Initial (b) Reduce Overlap (c) Increase Visibility

(d) Step vs Observed Area (e) Step vs Overlapped Area

(f) Initial (g) Optimized

Fig. 10: Visualization of differentiable view optimization. In (a), (b) and (c), the black camera C_t is at time t. The orange camera C_{t+1} is at time $t+1$ with optimized pose. The cyan point cloud is $\mathcal{P}_t$ and the blue one is $\mathcal{P}_{t+1}$. The trajectory of C_{t+1} during the optimization process is colored in red. Starting from the setup in (a), C_{t+1} is optimized to reduce the overlap between $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$ as shown in (b). In (c), the camera C_{t+1} gets a bigger view not overlapped with C_t's than (b). The risk loss ensures that C_{t+1} stays away from the wall while trying to get a larger view. (d) and (e) show the curves of total observed area and overlapped area, respectively. (f) and (g) show an example of optimizing multiple trajectory waypoints. With our method, the observed area along the trajectory is significantly larger.

D. Limitations

Our method has several limitations. The training of SDDF requires enough diversity in ray directions, which necessitates data augmentation techniques when insufficient sensor observations are available. Diversity in viewing directions is especially important for our model to excel with sparser measurements, as evident in the downsampling in our experiments. As shown in the results, our model performs worse with RGB-D measurements, although the RGB-D camera generates denser observations than the LiDAR in our experiments. This is due to the RGB-D camera having a smaller vertical field of view and, hence, less diversity in viewing directions. This suggests a future research direction of efficiently enhancing the variety of viewing directions subject to a limited sensing budget. One approach is to synthesize additional viewing directions based on the existing measurements using methods like hidden point removal (HPR) [58]. However, HPR requires computing the convex hull of the point cloud, which can be computationally expensive for large-scale scenes. It is important to consider more efficient ways to collect diverse viewing directions in future work. Another direction is to train the model with losses that encourage generalization to unseen viewing directions. For example, we can synthesize viewing rays hitting the same surface point but from different directions and enforce consistency in the predicted SDDF values for these rays.

It is also important to speed up the training of the SDDF model to make it scale to even larger scenes and be applicable in real-time reconstruction scenarios. One possible direction is to explore the combination of differentiable sphere tracing [44] with scene-level SDF and SDDF learning, where the SDF and SDDF models improve each other to accelerate convergence.

Although our method is not very sensitive to the number of ellipsoids, the algorithmic addition and pruning of ellipsoids will obviate the need to choose the correct number and improve robustness to different scenes. The current implementation is not well optimized, although it is faster than SDF-based sphere tracing. Addressing these limitations is an interesting direction for future work.

IX. Conclusion

In this work, we introduced a new definition of SDDF suitable for scene-level representation and differentiable rendering, and developed a hybrid explicit-implicit model to learn SDDF. Our method uses ellipsoids to obtain a coarse geometric prior and a residual network with latent features, obtained from the ellipsoids, to correct the differences between the prior and the ground truth. Our experiments demonstrate that SDDF is a promising scene representation for fast novel view rendering and gradient-based viewpoint optimization.

References

[1] B. Zhang, C. Fang, R. Shrestha, Y. Liang, X. Long, and P. Tan, "RaDe-GS: Rasterizing Depth in Gaussian Splatting," in *arXiv preprint: arXiv 2406.01467*, 2024. 2, 3, 9, 10, 11

[2] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," *ACM Trans. Graph.*, 2022. 2, 9, 10, 11

[3] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-board MAV Planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017. 1, 3

[4] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images," in *ECCV*, 2018. 1

[5] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1689–1696. 1

[6] J. Behley and C. Stachniss, "Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments," *Robotics: Science and Systems XIV*, 2018. 1

[7] F. Lu, G. Chen, Y. Liu, Y. Zhan, Z. Li, D. Tao, and C. Jiang, "Sparse-to-Dense Matching Network for Large-Scale LiDAR Point Cloud Registration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 1

[8] Y. Zeng, J. Hou, Q. Zhang, S. Ren, and W. Wang, "Dynamic 3D Point Cloud Sequences as 2D Videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 1

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658

[9] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees," *Autonomous Robots*, 2013. 1

[10] 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)*, 2019. 1

[11] 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 Vision and Pattern Recognition (CVPR)*, 2019. 1, 3

[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 (ECCV)*, 2020. 1, 2

[13] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Transactions on Graphics*, 2023. 1, 2

[14] J. C. Hart, "Sphere Tracing: a Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces," *The Visual Computer*, 1996. 1, 3, 9

[15] E. Zobeidi and N. Atanasov, "A Deep Signed Directional Distance Function for Shape Representation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024. 1, 2, 3, 4, 11, 12

[16] T. Aumentado-Armstrong, S. Tsogkas, S. Dickinson, and A. Jepson, "Representing 3D Shapes with Probabilistic Directed Distance Fields," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 3, 4, 11, 12

[17] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica Dataset: A Digital Replica of Indoor Spaces," *arXiv preprint arXiv:1906.05797*, 2019. 2

[18] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson Env: Real-World Perception for Embodied Agents," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 9

[19] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa, "Nerfstudio: A Modular Framework for Neural Radiance Field Development," in *SIGGRAPH*, 2023. 2, 9, 10, 11

[20] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "NeRF++: Analyzing and Improving Neural Radiance Fields," in *arXiv preprint: arXiv 2010.07492*, 2020. 2

[21] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[22] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. Valentin, "FastNeRF: High-Fidelity Neural Rendering at 200FPS," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[23] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, "Baking Neural Radiance Fields for Real-Time View Synthesis," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[24] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "PlenOctrees for Real-time Rendering of Neural Radiance Fields," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[25] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance Fields without Neural Networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[26] K.-A. Aliev, A. Sevastopolsky, M. Kolos, D. Ulyanov, and V. Lempitsky, "Neural Point-Based Graphics," in *European Conference on Computer Vision (ECCV)*, 2020. 2

[27] R. Rakhimov, A.-T. Ardelean, V. Lempitsky, and E. Burnaev, "NPBG++: Accelerating Neural Point-Based Graphics," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[28] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann, "Point-NeRF: Point-based Neural Radiance Fields," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[29] Q. Zhang, S.-H. Baek, S. Rusinkiewicz, and F. Heide, "Differentiable Point-Based Radiance Fields for Efficient View Synthesis," *SIGGRAPH Asia*, 2022. 2

[30] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2D Gaussian Splatting for Geometrically Accurate Radiance Fields," in *SIGGRAPH*, 2024. 2

[31] C. Chen, Y.-S. Liu, and Z. Han, "NeuralTPS: Learning Signed Distance Functions Without Priors From Single Sparse Point Clouds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. 3

[32] X. Long, C. Lin, L. Liu, Y. Liu, P. Wang, C. Theobalt, T. Komura, and W. Wang, "NeuralUDF: Learning Unsigned Distance Fields for Multi-View Reconstruction of Surfaces with Arbitrary Topologies," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[33] Y.-T. Liu, L. Wang, J. Yang, W. Chen, X. Meng, B. Yang, and L. Gao, "NeUDF: Learning Neural Unsigned Distance Fields With Volume Rendering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 3

[34] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D Reconstruction at Scale Using Voxel Hashing," *ACM Trans. Graph.*, 2013. 3

[35] K. M. B. Lee, Z. Dai, C. L. Gentil, L. Wu, N. Atanasov, and T. Vidal-Calleja, "Safe Bubble Cover for Motion Planning on Distance Fields," in *arXiv preprint: arXiv 2408.13377*, 2024. 3

[36] L. Wu, K. M. B. Lee, C. Le Gentil, and T. Vidal-Calleja, "Log-GPIS-MOP: A Unified Representation for Mapping, Odometry, and Planning," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 4078–4094, 2023. 3

[37] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian Optimization for Motion Planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013. 3

[38] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D Reconstruction at Scale using Voxel Hashing," *ACM Transactions on Graphics (TOG)*, 2013. 3

[39] B. Lee, C. Zhang, Z. Huang, and D. D. Lee, "Online Continuous Mapping using Gaussian Process Implicit Surfaces," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019. 3

[40] L. Wu, K. M. B. Lee, L. Liu, and T. Vidal-Calleja, "Faithful Euclidean Distance Field From Log-Gaussian Process Implicit Surfaces," *IEEE Robotics and Automation Letters (RA-L)*, 2021. 3, 4

[41] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit Geometric Regularization for Learning Shapes," in *International Conference on Machine Learning (ICML)*, 2020. 3, 4

[42] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-View Reconstruction," in *International Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 3

[43] Y. Jiang, D. Ji, Z. Han, and M. Zwicker, "SDFDiff: Differentiable Rendering of Signed Distance Fields for 3D Shape Optimization," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2026.3688658

[44] S. Liu, Y. Zhang, S. Peng, B. Shi, M. Pollefeys, and Z. Cui, "DIST: Rendering Deep Implicit Signed Distance Function With Differentiable Sphere Tracing," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 13

[45] M. Splietker and S. Behnke, "Directional TSDF: Modeling Surface Orientation for Coherent Meshes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019. 3

[46] P. Zins, Y. Xu, E. Boyer, S. Wuhrer, and T. Tung, "Multi-View Reconstruction Using Signed Ray Distance Functions (SRDF)," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[47] Y. Ren, F. Wang, T. Zhang, M. Pollefeys, and S. Süsstrunk, "VolRecon: Volume Rendering of Signed Ray Distance Functions for Generalizable Multi-View Reconstruction," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[48] T. Houchens, C.-Y. Lu, S. Duggal, R. Fu, and S. Sridhar, "NeuralODF: Learning Omnidirectional Distance Fields for 3D Shape Representation," in *arXiv preprint: arXiv 2206.05837*, 2022. 3

[49] Z. Liu, B. Yang, Y. Luximon, A. Kumar, and J. Li, "RayDF: Neural Ray-Surface Distance Fields with Multi-View Consistency," in *Neural Information Processing Systems (NeurIPS)*, 2023. 3

[50] T. Yenamandra, A. Tewari, N. Yang, F. Bernard, C. Theobalt, and D. Cremers, "FIRe: Fast Inverse Rendering using Directional and Signed Distance Functions," in *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024. 3

[51] J.-H. R. Chang, W.-Y. Chen, A. Ranjan, K. M. Yi, and O. Tuzel, "Pointersect: Neural Rendering with Cloud-Ray Intersection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[52] D. Arthur and S. Vassilvitskii, "k-means++: The Advantages of Careful Seeding," in *ACM-SIAM Symposium on Discrete Algorithms*, 2007. 8, 11, 12

[53] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," in *arXiv preprint: arXiv 1105.1186*, 2011. 9

[54] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe, "The Replica Dataset: A Digital Replica of Indoor Spaces," in *arXiv preprint: arXiv 1906.05797*, 2019. 9, 11

[55] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 9, 10

[56] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, "Depth-supervised NeRF: Fewer Views and Faster Training for Free," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 10

[57] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains," in *Neural Information Processing Systems (NeurIPS)*, 2020. 11, 12

[58] S. Katz, A. Tal, and R. Basri, "Direct Visibility of Point Sets," *ACM Trans. Graph.*, 2007. 13

Zhirui Dai (Graduate Student Member, IEEE) is a Ph.D. student of Electrical and Computer Engineering at the University of California San Diego, La Jolla, CA, USA. He obtained a B.S. in Physics from Fudan University, Shanghai, China, in 2019 and an M.S. in Electrical and Computer Engineering from the University of California San Diego in 2021. His research focuses on mobile robot autonomy and particularly on mapping and task planning.

Hojoon Shin is an autonomy engineer at Brain Corporation. He obtained a B.S. in Mechanical and Aerospace Engineering from Seoul National University, Seoul, South Korea, in 2021 and an M.S. in Mechanical Engineering from the University of California San Diego in 2023. His current work focuses on autonomous robotics, with particular interest in navigation and motion planning.

Yulun Tian (Member, IEEE) is an Assistant Professor of Robotics at the University of Michigan, Ann Arbor, MI, USA. Prior to this appointment, he was a Postdoctoral Scholar at the Contextual Robotics Institute, University of California San Diego, La Jolla, CA, USA. He received the B.A. degree in Computer Science from UC Berkeley, Berkeley, CA, USA, in 2017, and the S.M. and Ph.D. degrees in Aeronautics and Astronautics from Massachusetts Institute of Technology, Cambridge, MA, USA (2019 and 2023). His work received the 2024 Best Dissertation Award from the IEEE RAS Technical Committee for Multi-Robot Systems, the 2022 King-Sun Fu Memorial Best Paper Award from the IEEE Transactions on Robotics, a 2021 Honorable Mention from the IEEE Transactions on Robotics, and a 2020 Honorable Mention from the IEEE Robotics and Automation Letters. His current research interest includes spatial perception, trustworthy autonomy, and multi-agent systems.

Ki Myung Brian Lee (Member, IEEE) is a Postdoctoral Scholar at the Contextual Robotics Institute, University of California San Diego, La Jolla, CA, USA. He received the B.Eng. (Hons I) degree in mechatronics (space) from the University of Sydney, Camperdown, NSW, Australia, and the Ph.D. degree in robotics from the University of Technology Sydney, Ultimo, NSW, Australia, in 2023. He was recognized as an RSS Pioneer of 2023, and was the recipient of the UTS Research Excellence Scholarship. His current research aims to develop novel representations of environments and tasks that accelerate planning and control. More broadly, he is interested in mobile robot autonomy in previously unseen environments.

Nikolay Atanasov (S'07-M'16-SM'23) is an Associate Professor of Electrical and Computer Engineering at the University of California San Diego, La Jolla, CA, USA. He obtained a B.S. degree in Electrical Engineering from Trinity College, Hartford, CT, USA in 2008 and M.S. and Ph.D. degrees in Electrical and Systems Engineering from the University of Pennsylvania, Philadelphia, PA, USA in 2012 and 2015, respectively. Dr. Atanasov's research focuses on robotics, control theory, and machine learning with emphasis on active perception problems for autonomous mobile robots. He works on probabilistic models for simultaneous localization and mapping (SLAM) and on optimal control and reinforcement learning algorithms for minimizing probabilistic model uncertainty. Dr. Atanasov's work has been recognized by the Joseph and Rosaline Wolf award for the best Ph.D. dissertation in Electrical and Systems Engineering at the University of Pennsylvania in 2015, the Best Conference Paper Award at the IEEE International Conference on Robotics and Automation (ICRA) in 2017, the NSF CAREER Award in 2021, and the IEEE RAS Early Academic Career Award in Robotics and Automation in 2023.

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on May 24,2026 at 21:38:10 UTC from IEEE Xplore. Restrictions apply.
© 2026 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.