14

Map Representations with Differentiable Volume Rendering

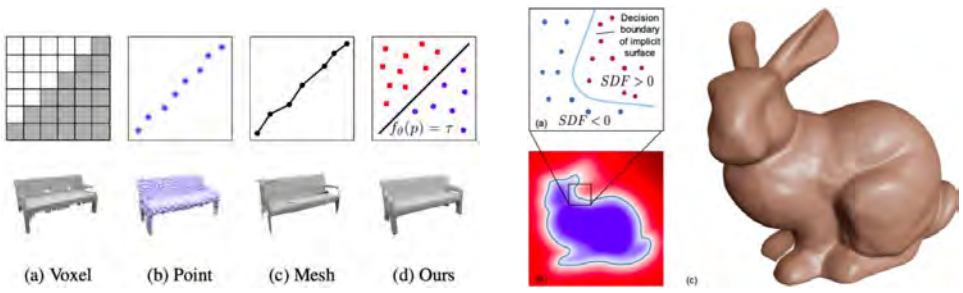
Hidenobu Matsuki and Andrew J. Davison

The choice of underlying scene representation significantly impacts the capability and efficiency of SLAM systems. As discussed in Chapter 5, each scene or map representation in SLAM has its own advantages and disadvantages, depending on the specific downstream task, such as camera ego-motion tracking, dense scene reconstruction, or object-level decomposition. Recently, 3D scene representations that can be optimized via differentiable rendering, such as Neural Radiance Field (NeRF) [770] or 3D Gaussian Splatting (3DGS) [560], have emerged as prominent choices for high-quality inverse rendering and novel view synthesis within the computer graphics research community, driven by advances in graphics hardware and automatic differentiation frameworks. Initially focused on offline novel view synthesis with known camera poses, these representations have gradually been applied to various other tasks in 3D reconstruction and scene understanding [1199], such as accurate and dense geometry reconstruction of large-scale scenes [825, 1236, 1157], semantic fusion [1286, 1287, 594], and more. Early implementations required days of optimization, but by leveraging explicit data structures (e.g., voxels or point clouds) to enable fast data access, optimization times have fallen to near real time. More recently, 3D Gaussian Splatting has been introduced, offering super-fast and high-quality rendering capabilities, effectively replacing Neural Radiance Fields in many applications. These novel graphics primitives are now also being integrated into SLAM research. Although the idea of combining differentiable rendering with 3D occupancy maps dates back to Thrun et al.'s "forward sensor model" in the early 2000s [1093], modern hardware and algorithms are opening entirely new possibilities. This chapter provides an overview of NeRF and 3DGS as scene representations for SLAM, discussing their advantages and disadvantages, and suggesting future research directions. We begin with a historical survey of 3D scene representations in Section 14.1, explore NeRF (Section 14.2) and 3DGS (Section 14.3) in detail, including their integration into SLAM. We further discuss potential future directions in Section 14.4, and conclude this chapter in Section 14.5.

14.1 Introduction

14.1.1 Learnable 3D Representations

With the success of deep learning in the 2D image domain [268, 450, 451], the research community gradually started to explore learning-based approaches for 3D tasks. Early works directly utilized 3D data represented by classical data structures such as points [897] and occupancy grids [935], processing them for downstream tasks. However, these classical representations are discrete and often not amenable for deep learning; they often struggle to represent watertight and continuous surfaces with flexible topology in a compact manner, which limits the gradient-based optimization essential for neural network training. To create more optimization-friendly 3D representations, some early works introduced 3D representations using neural networks as scalar function approximators to define occupancy or signed distance functions [766, 853, 199], which can be trained through direct 3D supervision. DeepSDF [853], for example, directly regresses a signed distance function (SDF) from a 3D coordinate and optionally a latent code using 8-layer Multi-Layer Perceptron (MLP) as shown in Figure 14.1. These methods demonstrated continuous and compact 3D shape reconstruction without being constrained by topology and discretization errors. However, supervised training on 3D data requires costly ground truth data annotations, making dataset collection non-scalable. This limitation has motivated research efforts toward leveraging easier-to-obtain 2D image information and varying levels of supervision for 3D scene understanding, which has led to 3D reconstruction via differentiable rendering.



(a) Occupancy Networks [766] (b) DeepSDF [853]

Figure 14.1 Learnable 3D Representations. Both ONet [766] and DeepSDF [853] regress continuous implicit surface functions modelled by neural network. TODO: Reuse permission

14.1.2 Differentiable Rendering

Supervised learning on 3D data poses challenges due to the difficulty and expense of obtaining large-scale ground truth annotations. To overcome these issues, a graphics rendering pipeline has been integrated into neural network training, leading to the advancement of Differentiable Rendering (DR). DR provides this capability by allowing gradients to flow from 2D images back into the 3D scene parameters (Figure 14.2). In a graphics pipeline, a 3D model may be represented in one of two ways:

- **Surface data**, which explicitly encodes object boundaries (e.g., polygons or implicit surfaces).
- **Volume data**, which describes continuous fields such as density or color, representing an object's internal properties and material distribution.

The rendering processes for these representations are called **surface rendering** and **volume rendering**, respectively. Accordingly, different rendering methods are used for each representation (Figure 14.3):

- Surface rendering often employs a method called **rasterization**, which projects surface geometry onto a 2D plane. It leverages hardware acceleration, making it highly efficient for explicit surfaces.
- Volume rendering is typically implemented via **ray marching**, which casts rays through a volume, sampling values along each ray's path. While well-suited for continuous fields, it requires higher computational cost.

Although early DR approaches used softened rasterization for meshes [702, 549, 687], volume rendering has gradually adopted as a more powerful approach for gradient-based optimization. By providing gradients over 3D space —even before identifying exact surface positions— volume rendering makes neural network training easier and thus is widely adopted as a DR backbone. Following NeuralVolume [695], works like differentiable volumetric rendering [813] and Scene Representation Networks [1021] use differentiable ray marching to represent implicit surfaces. Although initially limited to simple shapes, these methods laid the groundwork for more complex and realistic 3D reconstructions.

14.1.3 3D Scene Representation with Differentiable Rendering

A major breakthrough occurred in 2020 when Mildenhall et al. [770] introduced NeRF, a technique for scene reconstruction that optimizes a MLP using volumetric ray-marching from posed color images. NeRF employs a single MLP as the scene representation, performing volume rendering by marching along pixel rays and querying the MLP for opacity and color. Given that volume rendering is naturally

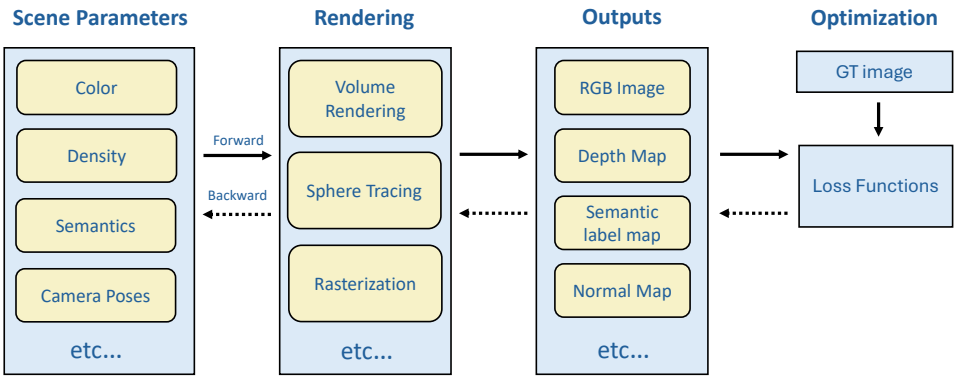
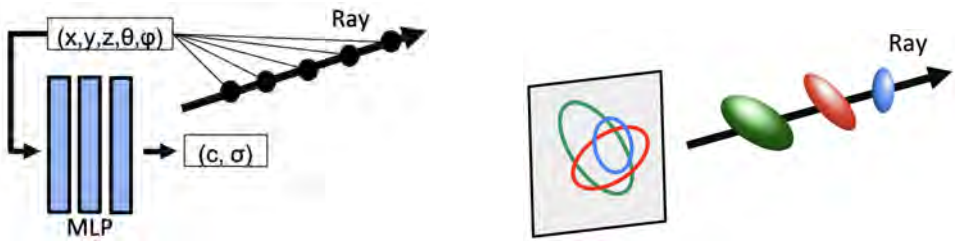


Figure 14.2 Overview of differentiable rendering pipeline. DR allows to optimize 3D scene parameters from multi-view 2D images in a end-to-end manner thanks to its consistent gradient propagation.



(a) Ray marching (NeRF) (b) Rasterization/Splatting (3DGS)

Figure 14.3 Comparison of the rendering methods in NeRF and 3D Gaussian Splatting. In ray-marching based approaches, the inference results of points sampled at fixed intervals along a ray are blended, whereas in rasterization, each primitive along the ray is projected onto the image plane and then blended.

differentiable, the MLP is optimized to minimize rendering loss using multiview information, achieving high-quality novel view synthesis. NeRF's MLP inputs include spatial location and viewing direction, combined with positional encoding to capture high-frequency scene details. Despite achieving high-quality novel view synthesis, NeRF's primary bottleneck is its training speed. To address this, recent developments have introduced explicit volume structures, such as voxel grids [1055, 349] or hash functions [790], to enhance performance. These advances have shown that the key to high-quality novel view synthesis lies in differentiable volumetric rendering rather than the neural network itself. They have demonstrated that it is possible to achieve rendering quality comparable to NeRF without relying on an MLP. However, even in these improved systems, per-pixel ray marching continues to be a significant bottleneck for rendering speed.

In parallel to the explosion of NeRF and volume rendering, differentiable point-based rendering techniques have also been researched [605, 959, 564]. These methods use point-based representation such as a point cloud [605, 959] or metaball [564], and define a differentiable rendering pipeline using a CNN based rasterizer or approximated renderer. In 2023, Kerbl et al. [1314] proposed a radically different point-based rendering approach for radiance field capture based on 3D Gaussian Splatting [560] (3DGS). 3DGS represents a scene using numerous semi-transparent Gaussian blobs that are rendered via alpha-blending. Unlike NeRF, 3DGS employs differentiable rasterization (Figure 14.3). Similar to traditional graphics rasterization, 3DGS iterates over the primitives to be rasterized rather than marching along rays. This method leverages the natural sparsity of a 3D scene, resulting in a representation that is expressive enough to capture high-fidelity 3D scenes while offering significantly faster rendering with similar training speed to the most efficient volumetric NeRF.

In the following sections, we explore the technical details of NeRF and 3DGS (Figure 14.4). Despite their shared goal of radiance field capture, these methods differ significantly in scene representation and rendering pipeline. We begin with an overview of NeRF, focusing on its ray-marching-based differentiable rendering formulation and its various extensions, which utilize diverse data structures and evolve into generalized field representations known as Neural Fields. Following this, we examine 3DGS and its rasterization-based rendering pipeline, highlighting the key differences from NeRF.

Figure 14.4 Novel View Synthesis Examples of MipNeRF360 [65] (left [TODO: Reuse permission]) and 3DGS [560] (right). Both methods can capture high fidelity radiance field from posed RGB images taken in real-world. The images are from [65, 560]

14.2 Neural Radiance Fields (NeRF)

In this section, we explain the technical details of NeRF, its derivative methods, and its application to visual SLAM. This technical background provides the fundamental

knowledge needed to understand the motivations and differences with respect to 3D Gaussian Splatting in the next section.

14.2.1 Method Overview

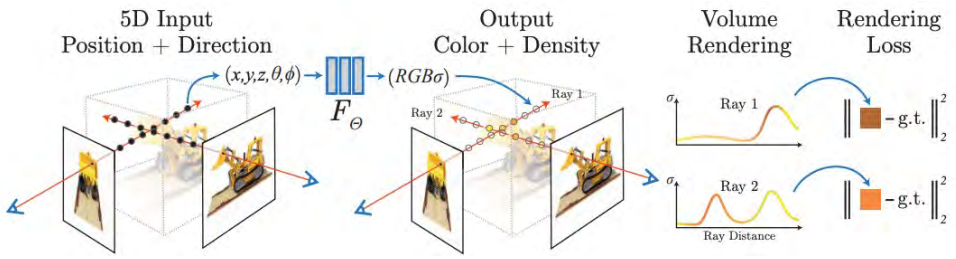


Figure 14.5 Overview of NeRF optimization pipeline. Radiance Field is represented by MLP and 2D images are rendered via differentiable volume rendering. MLP parameters are optimized to minimize the photometric error between rendered color image and ground truth color image. The image is from [770]. [TODO: Reuse permission]

The NeRF model is expressed as a function $f_\Theta(\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$, with MLP weights Θ, in-scene 3D coordinates $\mathbf{x} = (x, y, z)$, viewing direction $\mathbf{d} = (\theta, \phi)$, color $\mathbf{c} = (r, g, b)$ and volume density σ . To synthesize novel views, the NeRF workflow involves casting camera rays through the scene, generating sampling points for each pixel. The local color and density at each sampling point are computed using the NeRF MLP(s), and volume rendering is used to synthesize the 2D image (Figure 14.5). The expected color $C(\mathbf{r})$ of camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is calculated using the following integral formulation:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})\mathrm{d}t \tag{14.1}$$

Here, $\mathrm{d}t$ denotes the differential distance traveled by the ray at each integration step and t_n and t_f denote near and far bounds. The terms $\sigma(\mathbf{r}(t))$ and $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$ represent the volume density and color at point $\mathbf{r}(t)$ along the camera ray with viewing direction $\mathbf{d}$, respectively. The accumulated transmittance from t_n to t is given by $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s)), \mathrm{d}s\right)$. The continuous integral is estimated numerically with quadrature by dividing the ray into N evenly-spaced bins and sampling uniformly between bins and randomly within each bin. The volume rendering equation using the quadrature rule is formulated as:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \alpha_i T_i \mathbf{c}_i, \quad \text{where } T_i = \exp\Big(-\sum_{j=1}^{i-1} \sigma_j \delta_j\Big). \tag{14.2}$$

$\delta_i = t_i$ and t_{i+1} denote the interval between consecutive samples t_i and t_{i+1}, while σ_i and $\mathbf{c}_i$ indicate the density and color evaluated at sample point i along the ray, respectively. Additionally, $\alpha_i = (1 - \exp(-\sigma_i \delta_i))$ denotes the opacity resulting from alpha compositing at sample point i.

To optimize the model, a quadratic photometric error between the rendered image and the ground truth color image is used:

$$L = \sum_{r \in R} \|\hat{C}(\mathbf{r}) - C_{gt}(\mathbf{r})\|_2^2 \ . \tag{14.3}$$

where $C_{gt}(r)$ denotes the ground truth color for the pixel associated with $\mathbf{r}$ in the training image, and R denotes the batch of rays used for synthesizing the target image. In the context of SLAM, depth rendering through alpha-blending of opacity is often employed, to additionally minimize discrepancies between the expected depth and sensor or predicted depth.

14.2.2 Data Structures in NeRF

The original NeRF paper [770] uses a single MLP to represent the entire radiance field and requires a forward inference to evaluate the color and density at specific 3D location. This means all the network parameters are involved with the evaluation and the forward inference and backward training is significantly slow. To achieve faster inference and training, several approaches introduced explicit data structure either partially or completely, and achieved fast data access.

Voxel/Hash Grid. Early works introduced a voxel grid to speed up the forward inference. Later on, Plenoxel [349] and DirectVoxGo [1055] introduced voxel feature-grid as trainable properties and encode radiance field into it. Compared to MLP based NeRF, color and density at specific location is queried by simple index lookup and trilinear interpolation, which makes the whole pipeline significantlly faster (from days to 10mins). Instant-NGP [790](Figure 14.6) further introduced multi-resolution hash encoding, efficiently encodes volumetric features in a hash table, and together with highly optimized CUDA implementation, it achieves NeRF training in a few seconds.

Points and Surfels. Point-based representations have also been used in conjunction with NeR (Figure 14.7). Point-NeRF [1205] optimizes point-clouds with features and tiny MLP decoder, and render the image via a volume rendering framework. It demonstrated that the point-based representation has a flexibility of spatial allocation compared to regular voxel grids, which allows the model to skip empty spaces, resulting in a speed-up of a factor of 3 over vanilla NeRF. Similar to point clouds, surfels can also be used to represent radiance field [374].

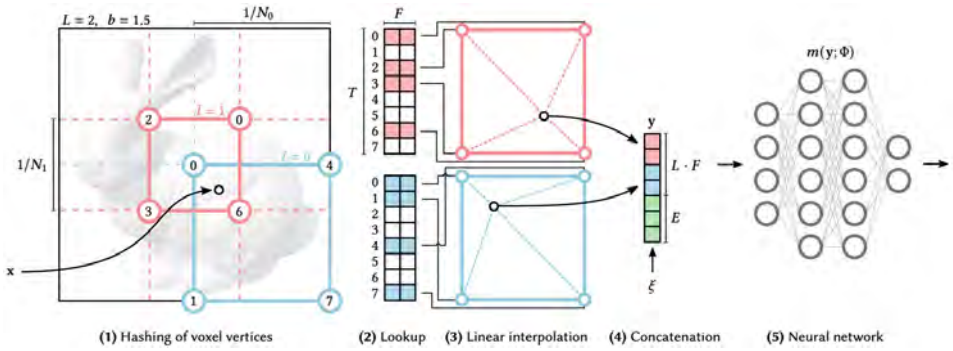


Figure 14.6 An example of grid-based NeRF (Instant-NGP [790]). The radiance field is modelled by a combination of multi-resolution hash grid and tiny MLP decoder. The image is from [790]

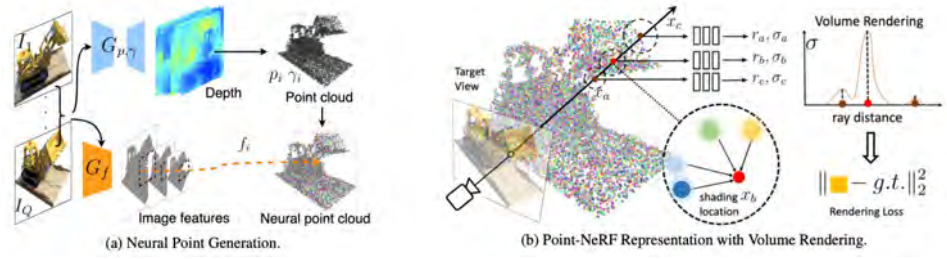


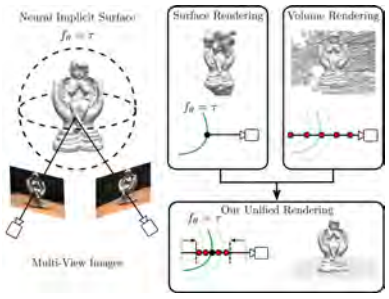
Figure 14.7 An example of point-based NeRF representation (Point-NeRF [1205]). The radiance field is modeled by a combination of points with trainable features and tiny MLP decoder. The image is from [1205]. [TODO: Reuse permission]

14.2.3 Neural Fields

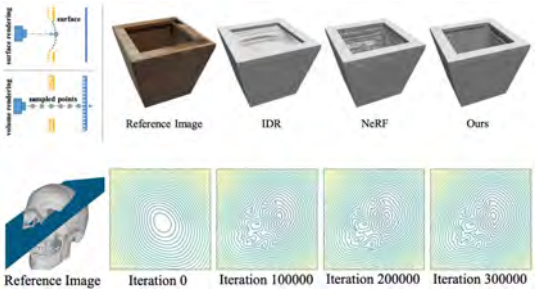
While initial NeRF work primarily focuses on representing a radiance field — mapping 3D positions to color and density for photo-realistic scene capture— the underlying concept can be extended to various other outputs, depending on the downstream task. Such generalized field representations are termed *Neural Fields*. Here, we introduce a few major examples.

Surface Representations (SDF and Occupancy). To achieve more geometrically accurate 3D reconstructions from RGB images, Neural Fields can utilize representations like Signed Distance Functions (SDF) and Occupancy Fields (Figure 14.8). Techniques such as NeuS [1157] and VolSDF [1236] leverage SDF to represent surfaces. SDF encodes the distance of a point from the nearest surface, with the sign indicating whether the point is inside or outside the object. By integrating SDF into NeRF, it is possible to achieve precise surface modeling. Techniques like UNISURF [825] employ occupancy fields to represent the presence or absence of

surfaces in the 3D space. Occupancy fields divide the space into a grid, where each cell indicates whether it is occupied by a surface. This method can be combined with NeRF to handle complex scenes effectively. To extract meshes from these representations, techniques such as marching cubes can be applied to convert the continuous surface representation into a discrete mesh format.



(a) UNISURF [825].[TODO: Reuse permission]



(b) NeuS [1157].

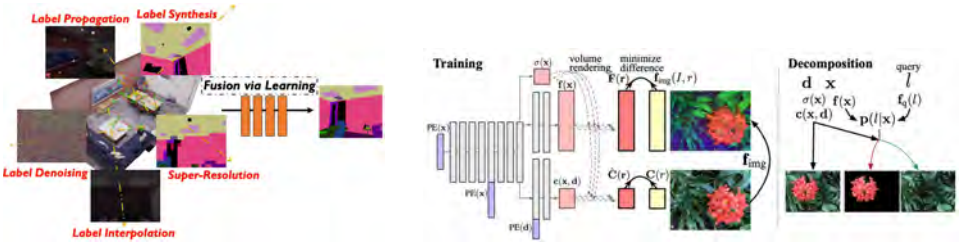
Figure 14.8 Neural Field for surface reconstruction. UNISURF represents occupancy field and NeuS represents continuous Signed Distance Functions via MLP which can be rendered via volume rendering. The images are from [825, 1157]

Semantics. Incorporating semantic information into Neural Fields enables the decomposition of scenes into meaningful components, enhancing scene understanding and interaction (Figure 14.9). Early works such as [1286, 1287] extend NeRF by outputting semantic labels in addition to color and density. This is achieved by training the model with multi-view semantic images, enabling it to predict the semantic category of each part of the scene. These fields fuse high-dimensional features from multiple views to produce a 3D consistent semantic feature field. More generally, NeRF can fuse 2D features from pre-trained foundation networks and perform more advanced semantic tasks such as scene decomposition and interactive editing via language prompt or clicking [594, 1115, 752].

14.2.4 NeRF/Neural Fields for Visual SLAM

Neural Fields were initially designed for offline novel view synthesis using posed color images. However, their ability to produce high-fidelity reconstructions has driven interest in applying them to Visual SLAM, which requires the real-time joint estimation of camera pose and scene representation.

Neural Field-based 3D representations offer several advantages over classical dense SLAM methods. First, end-to-end optimization via differentiable rendering simplifies 2D-to-3D data fusion, eliminating the need for multiple hand-designed fusion steps as in TSDF-based SLAM, where consistency is not always guaranteed.



(a) SemanticNeRF [1286] (b) Distilled Feature Field [594]

Figure 14.9 Neural Field for semantic scene reconstruction. By using additional semantic head for MLP, NeRF can output multi-view consistent semantic label/features which can be trained by per-view semantic prediction. The images are from [1286, 594]

Second, Neural Fields can model complex light fields, handling transparent objects and reflections more flexibly for photometric error minimization. Third, depending on the data structure, they enable scene completion and compression, leveraging priors inherent in the neural representation.

iMAP [1052](Figure 14.10) was the first work in this area, utilizing a purely MLP-based map representation similar to the original NeRF. The key insight of iMAP is that an MLP-based approach benefits from map compression and scene completion due to the inherent smoothness prior in MLPs. Similar to the development of NeRF research, the integration of classical and neural representations has also been investigated in the SLAM context. NICE-SLAM [1304] was the first to introduce a hybrid representation combining a multi-resolution voxel feature grid with a tiny MLP. Similar grid-type representations have been adopted by other works [1227, 532, 1153]. More recently, PointNeRF-style point primitives have also been applied to SLAM [973]. Among the various technical components, the choice of underlying data structure has the greatest influence on reconstruction quality, computational efficiency, and real-time performance in SLAM. Given this, we next examine the various data structures used in Neural-Field SLAM methods and detail their key characteristics and trade-offs.

14.2.4.1 Data Structure Characteristics and Trade-Offs in Neural Field SLAM

The choice of underlying data structure significantly impacts the properties of Neural Field–based SLAM. Table 14.1 provides an overview of various Neural-Field SLAM methods. Below, we detail the major characteristics and trade-offs associated with these representations.

Trade off between Compression and Inference speed. While the initial Neural Field SLAM work uses a single-MLP, more recent methods use (partially or entirely) more explicit data structures similar to the classical graphic primitives. MLP approximates the entire radiance field function via a network's parameters

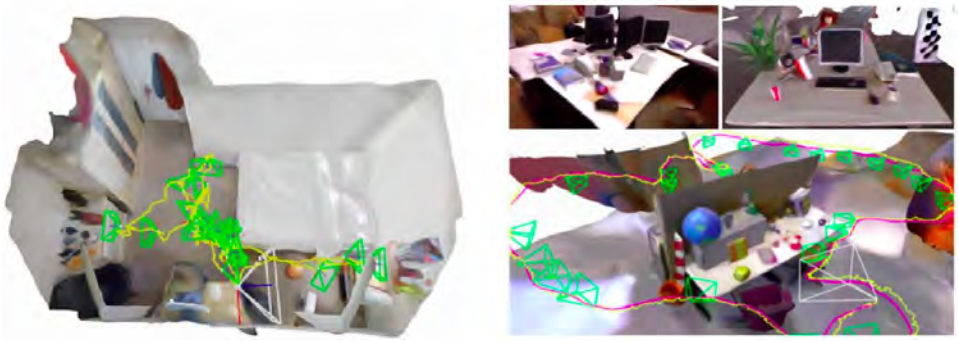


Figure 14.10 Qualitative results of iMAP [1052]. The method firstly demonstrated that MLP and differentiable rendering framework can be used for real-time joint estimation of camera pose and scene geometry from RGB-D video stream.

	MLP	Voxel/Hash	Point/Surfel
Compression	Good	Moderate	Moderate
Inference speed	Limited	Good	Moderate
Scene Completion	Good	Good	Moderate
Dynamic Allocation	Limited	Limited	Good
Forgetting Problem	Limited	Moderate	Moderate

Table 14.1 *A comparison of underlying data structure for Neural Field.*

and has huge benefits in terms of compression, but the physical meaning of the parameters is hard to interpret. On the other hand, explicit data structures like voxel grids are more interpretable and allow fast data access and local update of the map, but they require more memory. To that end, the map representation exhibits a typical time-space complexity trade-off. For SLAM tasks in particular, real-time operation is paramount and therefore explicit data structures are often preferable in practice.

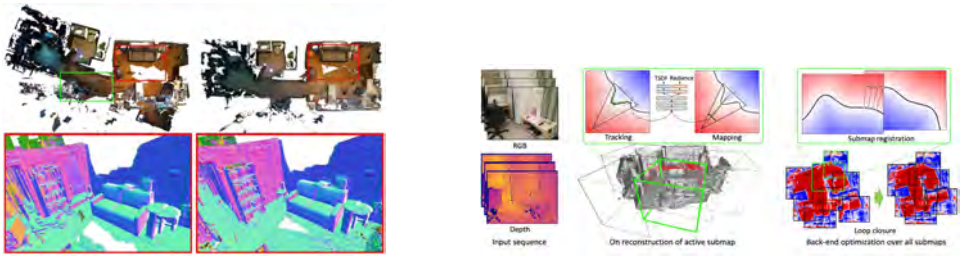
Scene Completion. iMAP demonstrated that MLP-based 3D reconstruction has hole-filling capability of the unobseved region, thanks to the continuous nature of the function. Multi-resolution grid also has similar property (Figure 14.11). This is because the positional encoding or multi-resolution grid resolution smooths the signal within the frequency or grid resolution. This is not not observed by point-type representation where the primitive is discrete and does not have global smoothing.

Dynamic Allocation of Graphic Primitives. Unlike offline scene reconstruction tasks, SLAM requires the online update of state variables based on run-time observations to adjust for dynamic changes, such as scene boundaries and loop closures. In Neural Fields, MLP and voxel-type representations are difficult to adjust for these updates due to their fixed boundaries and positional encoding/voxel resolution, which requires dynamic allocation of multiple local NeRF models [1074]

(a) iMAP [1052] (b) NICE-SLAM [1304]

Figure 14.11 Hole-filling effect of NeRF-based SLAM. Thanks to the use of multi-resolution frequency or grid encoding and continuous MLP function, the representation has a capability to smooth local geometry and provides plausible hole-filling effect of small unobserved regions. The images are from [1052, 1304]. [TODO: Reuse permission for NICE SLAM (by Marc)]

(Figure 14.12). On the other hand, point representations offer more flexibility in primitive allocation and can handle loop closures more effectively [680].



(a) Loopy-SLAM [680] (b) MIPS-Fusion [1074]

Figure 14.12 Loop Closure on NeRF-based SLAM methods. While voxel-based approaches require multiple local submap of Neural Field to handle camera pose update by loop closure, discritized point-based representation can more seamlessly change the primitive allocation. Images are from [680, 1074]. [TODO: Reuse permission]

Forgetting Problem. In both purely MLP-based and hybrid representations, the existence of a global function implies that any local parameter update impacts the overall results. Consequently, the map must retain past information as a training signal within the optimization window and requires an explicit keyframe database. This issue, often referred to as the "forgetting problem," can be mitigated by reducing reliance on or completely removing the global function, typically represented by an MLP.

14.3 3D Gaussian Splatting

Figure 14.13 Novel view rendering results of 3DGS. (Left:) RGB rendering. (Right:) Shaded Gaussians. 3DGS represents the scene by allocating a large number of semi-transparent 3D Gaussians.

While NeRF performs differentiable ray-marching, 3DGS performs differentiable rasterization for volume data. Similar to regular graphics rasterization, by iterating over the primitives to be rasterized rather than marching along rays, 3DGS leverages the natural sparsity of a 3D scene and achieves a representation which is expressive enough to capture high-fidelity 3D scenes while offering significantly faster rendering (Figure 14.13). The input consists of a set of color images with corresponding poses estimated by structure from motion (SFM). In addition, 3DGS takes the sparse point cloud generated during the SFM process, which is used for constructing a set of initial 3D Gaussians as graphical primitives.

14.3.1 Method Overview

Each 3D Gaussian is parameterized by its 3D mean position $\boldsymbol{\mu} \in \mathbb{R}^3$, covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$, opacity o, and direction-dependent color $\mathbf{c}$ represented by spherical harmonics (SH). Given a scaling matrix $\boldsymbol{S} \in \mathbb{R}^{3\times3}$ and rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$, the covariance matrix $\boldsymbol{\Sigma}$ is represented as :

$$\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{R}^\top \tag{14.4}$$

These 3D Gaussians are projected onto the image plane ($z = 1$ plane) via an ap-

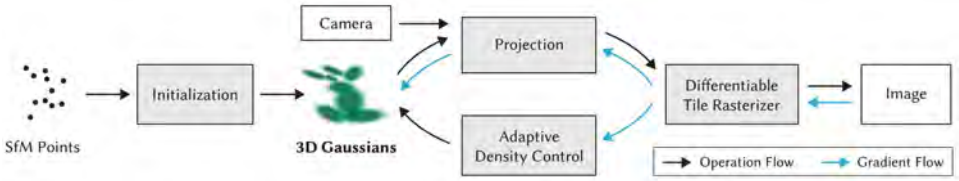


Figure 14.14 Overview diagram of the 3DGS optimization process [560].

proximated projection function for differentiable rasterization. The projected Gaussian's covariance matrix in 2D image space $\mathbf{\Sigma'}$ is formulated as:

$$\mathbf{\Sigma'} = \mathbf{JW\Sigma W}^\top \mathbf{J}^\top \tag{14.5}$$

where $\mathbf{J}$ denote Jacobian of the projective transformation and $\mathbf{W}$ is viewing transformation matrix. The upper left 2×2 block of $\mathbf{\Sigma'}$ is used as a 2D covariance matrix. We obtain the value of the 2D Gaussian function by querying a pixel $\mathbf{x}_{2\mathrm{D}} = \begin{bmatrix} u, v \end{bmatrix}^\top$.

The final color C of the pixel $\mathbf{x}_{2\mathrm{D}}$ is determined through alpha blending of the 3D Gaussians:

$$C = \sum_{i \in \mathcal{N}} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{14.6}$$

where $\mathcal{N}$ is the set of Gaussians along the pixel's ray, and α_i is the product of the opacity o_i the pixel-space Gaussian function:

$$\alpha_i = o_i G_{2\mathrm{D}} \left(\mathbf{x}_{2D} \right) \tag{14.7}$$

In alpha-blending, the 3D Gaussians are sorted in depth order relative to the rendering viewpoint. This sorting process is performed for each tile of the input image, which is divided into 16×16 tiles, using a GPU-optimized radix sort. Moreover, for efficiency, only the Gaussians that fall within the current view frustum are considered (Frustum Culling). The parameters of the 3D Gaussians are estimated by minimizing a weighted sum of the L1 loss and the SSIM loss between the rendered image and the ground truth image, with a weight λ

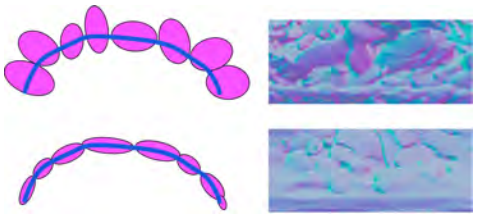
$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda \mathcal{L}_{\mathrm{SSIM}} \tag{14.8}$$

Furthermore, during the optimization process, adaptive densification and pruning are performed to flexibly adjust the number of Gaussians based on their gradients and opacities (Figure 14.14).

14.3.2 Applications of 3D Gaussian Splatting

Similar to NeRF's evolution into a generic Neural Field, 3DGS has been applied to various 3D vision tasks. Given the large volume of research, we highlight a few early papers in this area.

Surface Reconstruction. 3DGS is primarily designed for photorealistic novel view synthesis and does not inherently ensure geometric accuracy. Several works address this limitation (Figure 14.15). SuGAR [419] introduces geometric regularization to 3DGS, enabling geometrically accurate 3D mesh extraction. It defines surface normals for each 3D Gaussian and enforces local surface smoothness by encouraging neighboring Gaussians to be as flat as possible. 2D Gaussian Splatting (2DGS) [491] replaces 3D Gaussians with 2D disks, explicitly defining surface normals within the graphic primitives. By incorporating normal and depth consistency losses, 2DGS further improves surface reconstruction accuracy.



(a) SuGAR [419]. [TODO: Reuse permission]

(b) 2DGS [491].

Figure 14.15 Examples of surface-oriented Gaussian Splatting methods. By defining surface normal directions and enforcing smoothness between neighboring Gaussians, these methods achieve accurate surface reconstruction. Images are from [419] [491].

Semantic Scene Understanding. 3DGS can also be extended to semantic scene understanding. Like NeRF, the most common approach propagates 2D semantic segmentation to 3D Gaussians with semantic features via end-to-end training enabled by differentiable rendering. For instance, LangSplat [902] and FMGS [1311] integrate 2D foundation models such as CLIP and DINO into multi-view consistent 3D Gaussians during the 3DGS training process, enabling open-vocabulary semantic segmentation (Figure 14.16). Since 3D Gaussians are explicit discrete representations, editing the reconstructed scene (*e.g.,* manipulation or removal) is more straightforward than with the continuous NeRF representation.

14.3.3 3D Gaussian Splatting for SLAM

3DGS has achieved remarkable success in real-time novel view synthesis, and its potential for SLAM is currently under active exploration. Similar to NeRF, 3DGS leverages end-to-end optimization via differentiable rendering to seamlessly unify lo-

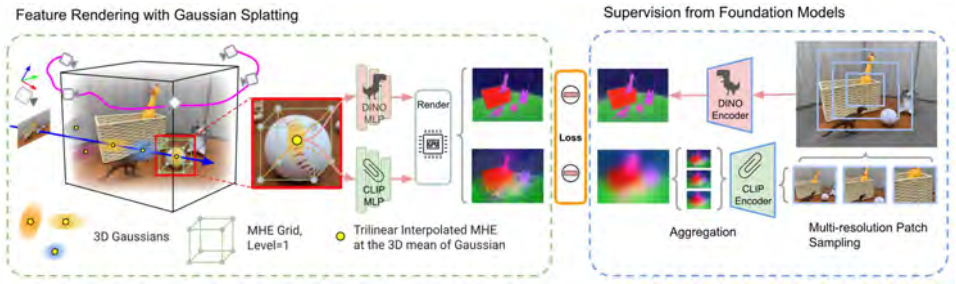


Figure 14.16 An example of Gaussian Splatting method for semantic scene reconstruction. Multi-view 2D image features are fused into 3D Gaussians via differentiable rendering. The image is from [1311]. [TODO: Reuse permission]

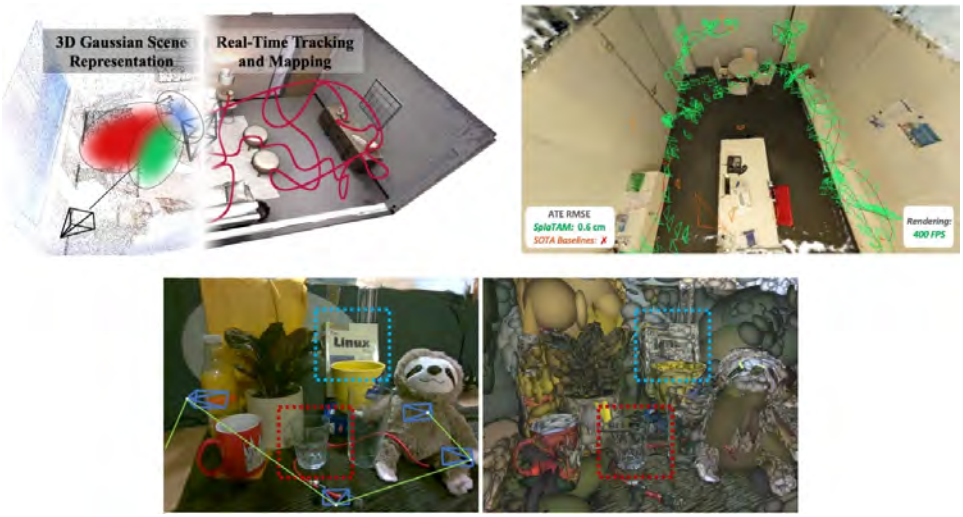


Figure 14.17 Examples of 3D Gaussian Splatting-based visual SLAM methods [1210, 556, 747]. [TODO: Reuse permission for [1210]; [556] is by Krishna]

calization and mapping. This integration eliminates the need for multi-stage, hand-designed algorithms and enables the modeling of complex light fields —including those produced by transparent objects— that are challenging to capture with classical 3D representations.

Beyond these similarities, 3DGS offers several unique advantages over NeRF. First, its efficient, rasterization-based pipeline enables significantly faster image rendering, directly benefiting downstream applications that require real-time map interaction (*e.g.,* robot navigation). Second, its representation as discrete, point-like Gaussian primitives allows for flexible allocation and updates of scene elements. This flexibility is particularly well-suited for SLAM tasks where state estimates

must dynamically adapt to runtime observations, such as scene boundary updates and loop closures.

Leveraging these properties, 3DGS can serve as a core scene representation in visual SLAM systems (Figure 14.17). Several early works have explored its inherent capabilities for simultaneous localization and mapping [1210, 556, 747]. A straightforward application is RGB-D SLAM, where an external structured sensor —such as a Time-of-Flight (ToF) depth camera— provides accurate, dense depth data that can be used as a prior or ground-truth signal for the positions of Gaussians. By incorporating this depth information, the positions of the 3D Gaussians can be accurately initialized and further regularized by minimizing a depth rendering loss, leading to accurate camera tracking as well as high-fidelity and fast map rendering.

However, these approaches require additional hardware to reconstruct 3D Gaussians, even though the original 3DGS method only needs a commodity camera to capture RGB images. Ideally, the method should operate solely in a vision-based monocular SLAM setting by fully leveraging the flexible and optimizable nature of 3D Gaussians. In this context, work such as MonoGS [747] has demonstrated that purely color image-based monocular SLAM is indeed possible (Figure 14.18). In MonoGS, even though the initial locations of the 3D Gaussians are unknown, they can be incrementally estimated by randomly initializing the Gaussians and refining their positions through multi-view optimization.

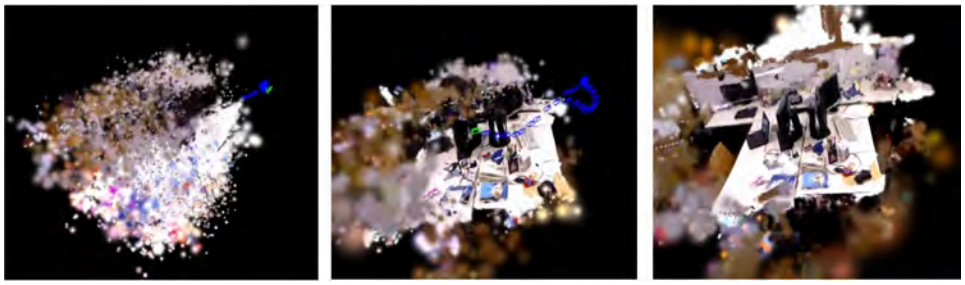


Figure 14.18 Monocular Reconstruction result of MonoGS [747]. Thanks to 3D Gaussian's flexibility of its spatial allocation, the method can perform joint estimation of camera pose and scene reconstruction only from color images without any depth supervision.

The discrete nature of 3D Gaussians provides significant advantages for global map updates, particularly in the context of loop closure. Because these representations consist of a collection of individual, well-defined primitives, it becomes easier to match and align various segments of a scene, even as the environment evolves over time. For instance, LoopSplat [1303] demonstrates a practical application of this concept. In their approach, the discrete 3D Gaussian elements serve as robust features that can be reliably registered across different time instances and viewpoints. Another notable advantage of 3DGS as a SLAM representation is its ability to cap-

ture complex light fields. By effectively capturing the intricate distribution of light within an environment, 3DGS not only enhances visual fidelity but also provides additional cues that can improve both mapping and localization performance. I^2-SLAM [49] fully exploits this property by integrating a more precise physics-based rendering process, such as motion blur and tone mapping, into the existing 3DGS framework. This integration enables simultaneous HDR map reconstruction and accurate camera pose tracking (Figure 14.19).

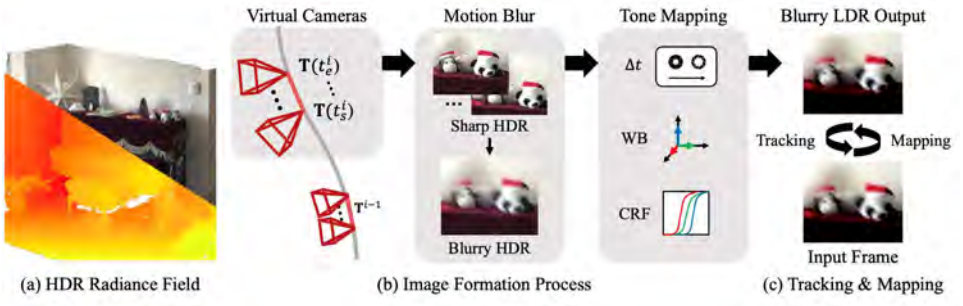


Figure 14.19 The system over view of I^2-SLAM [49]. The method integrates physical imaging process into SLAM to capture HDR radiance field. [TODO: Reuse permission]

While the full potential of 3D Gaussian Splatting (3DGS) as a unified SLAM representation is still under active investigation, another promising research direction involves integrating 3DGS into existing point-cloud processing methods. Since 3DGS can be treated as a point cloud by simply omitting its additional features, it becomes straightforward to combine with well-established point-based algorithms that are known for their stability and practical performance in real-world applications (Figure 14.20). For example, Photo-SLAM [496] combines a sparse tracking system (ORB-SLAM) with 3DGS, while both GS-ICP SLAM [431] and RTG-SLAM [875] leverage ICP for tracking. In the case of RTG-SLAM, the system further incorporates ORB keypoint-based backend optimization, resulting in fast and accurate performance with high practical applicability.

Furthermore, this inherent versatility of 3DGS extends to its fusion with LiDAR measurements. Several recent approaches have proposed hybrid LiDAR+3DGS frameworks [479, 1059, 631], demonstrating that 3DGS can serve as a unifying map representation across multiple sensor modalities. This wide range of applications, from RGB-based SLAM to LiDAR-enhanced reconstructions, demonstrates key advantages of 3DGS: its flexibility and adaptability, which make it an attractive choice for diverse SLAM scenarios.

(a) PhotoSLAM [496] (b) RTG-SLAM [875]

Figure 14.20 Example of the methods which combine 3DGS with keypoint-based SLAM. [TODO: Reuse permission]

14.4 Challenges and Future Directions

NeRF/3DGS-based SLAM offers several advantages, including end-to-end optimization, complex light-field capture, real-time rendering, and dynamic primitive allocation, making it a promising research direction. However, significant technical challenges remain. This section discusses these limitations and outlines potential future research directions.

Real-time performance. One major challenge of NeRF/3DGS-based end-to-end SLAM is its extremely slow processing speed. While classical dense SLAM methods like ElasticFusion [1181] achieved real-time performance including tracking, mapping, and rendering, on a GTX 780Ti in 2015, most NeRF/3DGS-based SLAM methods struggle to exceed 5 fps even on modern GPUs with significantly higher FLOPs and VRAM (*e.g.,* NVIDIA RTX 4090). This limitation makes current NeRF/3DGS-based SLAM impractical for real-time robotics applications, restricting its use to offline server GPUs. Since 3DGS enables real-time rendering, optimizing it efficiently remains a key challenge. One potential direction for improving efficiency is adopting a compact representation that reduces the number of Gaussians, thereby lowering computational overhead. Another approach is employing a higher-order optimizer instead of standard first-order gradient descent to accelerate convergence. Alternatively, a learned network could predict both camera poses and 3D Gaussians in a single forward pass, eliminating the need for iterative optimization. In support of this approach, PixelSplat [175] and MVSplat [198] have demonstrated that a **feed-forward network** can predict 3D Gaussian parameters from posed images without any test-time optimization. Although these early works required camera poses estimated from an external module, pose-free 3D reconstruction methods based on feed-forward networks have begun to gain traction following the success of DUSt3R [1162] (see Chapter 13 for more details), which robustly predicts pointmap without relying on pose input. Building on this progress, NoPoSplat [1237] (Figure 14.21) has extended the concept to achieve pose-free 3DGS prediction, showing promising results. Exploring these techniques for multi-view, incremental SLAM settings is an interesting future direction.

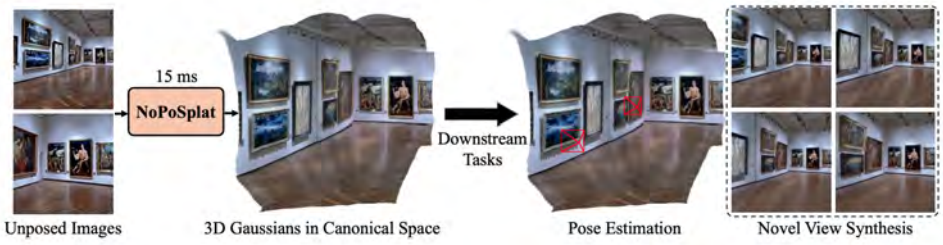


Figure 14.21 System overview of NoPoSplat [1237]. The method directly predicts 3D Gaussians from 2 input views, which can be used for pose estimation and novel view synthesis. The image is from [1237].

Camera Tracking Accuracy. Camera tracking accuracy in NeRF/3DGS-based SLAM often falls behind classical sparse SLAM methods. These approaches rely on photometric consistency across multiple frames, performing well on synthetic datasets like Replica [1045], where ground-truth depth and color images are available. However, in real-world scenarios, this assumption often breaks down due to unmodeled effects such as sensor noise and rolling shutter. Improving accuracy may require more precise modeling of the scene's light field and rendering process or exploring alternative matching paradigms beyond the color image domain. This challenge might also be addressed by incorporating a more robust feed-forward prediction model, as mentioned in the previous paragraph.

4D Scene Reconstruction. Currently, most NeRF/3DGS SLAM methods assume a static environment, while real-world scenes often exhibit significant dynamic changes. One common approach involves masking out potentially dynamic objects to focus solely on static background reconstruction for robust odometry estimation. However, considering the notable success of 3DGS in non-rigid/4D scene reconstruction and point-tracking, exploring a comprehensive 4D-SLAM framework appears to be a promising direction [144, 991, 1284, 748].

Large-scale/Outdoor Scene Reconstruction. Most NeRF/3DGS-based SLAM methods are currently limited to room-scale sequences, making large-scale and outdoor scene reconstruction a major challenge. Scaling up requires efficient map updates, such as adaptive resolution to manage computational load, and drift correction to maintain accuracy over extended areas. Outdoor environments introduce additional complexities, including illumination changes due to varying weather and time of day, which complicate light field modeling.

14.5 Conclusion

In this chapter, we provided a comprehensive overview of 3D scene representations using differentiable rendering, with a particular focus on NeRF and 3D Gaussian

Splatting. We reviewed the historical background that led to the development of NeRF, detailed the underlying principles of NeRF and its derivative techniques, and elaborated on the scene representation, rendering, and optimization methods employed by 3DGS. Finally, we discussed the application of these techniques to SLAM, current research trends, and potential future directions. Due to its high expressive capability and flexibility, 3DGS in particular is expected to be increasingly utilized across a wide range of practical applications. Moreover, 3DGS holds promise for further developments, including handling dynamic scenes, large-scale and outdoor environments, integrating semantic information, and learning from limited input data.