IMLS-Splatting: Efficient Mesh Reconstruction from Multi-view Images via Point Representation

KAIZHI YANG, University of Science and Technology of China, China
LIU DAI, University of California San Diego, USA
ISABELLA LIU, University of California San Diego, USA
XIAOSHUAI ZHANG, Hillbot Inc., USA
XIAOYAN SUN, University of Science and Technology of China, China
XUEJIN CHEN*, University of Science and Technology of China, China
ZEXIANG XU, Hillbot Inc., USA
HAO SU, University of California San Diego and Hillbot Inc., USA

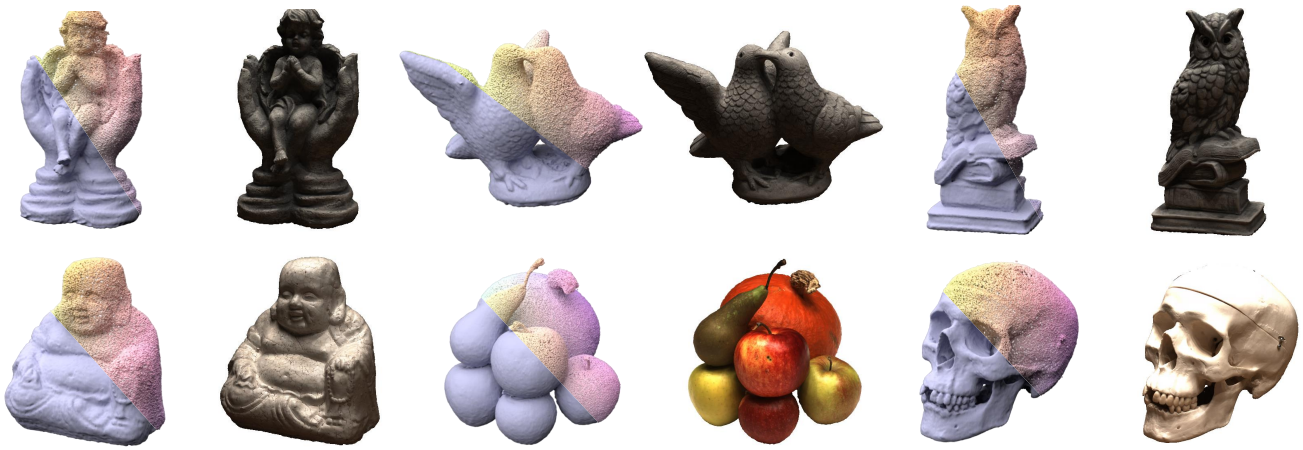


Fig. 1. Our IMLS-Splatting establishes a direct, differentiable bridge between discrete point clouds and continuous surfaces. By integrating differentiable rasterization, we achieve seamless end-to-end mesh optimization from multi-view images. Here, we present our reconstruction and rendering results from the DTU dataset, where our method produces uniformly distributed point clouds and high-detailed meshes that support high-quality renderings.

Multi-view mesh reconstruction has long been a challenging problem in graphics and computer vision. In contrast to recent volumetric rendering methods that generate meshes through post-processing, we propose an end-to-end mesh optimization approach called IMLS-Splatting. Our method leverages the sparsity and flexibility of point clouds to efficiently represent the underlying surface. To achieve this, we introduce a splatting-based differentiable Implicit Moving-Least Squares (IMLS) algorithm that enables the fast conversion of point clouds into SDFs and texture fields, optimizing both mesh reconstruction and rasterization. Additionally, the IMLS representation ensures that the reconstructed SDF and mesh maintain continuity and smoothness without the need for extra regularization. With this efficient pipeline, our method enables the reconstruction of highly detailed meshes in approximately 11 minutes, supporting high-quality rendering and achieving state-of-the-art reconstruction performance. Our code is available at https://github.com/SilenKZYoung/IMLS-Splatting.

CCS Concepts: • **Computing methodologies → Point-based models**.

Additional Key Words and Phrases: Multi-view, Point, Mesh, Rasterization

ACM Reference Format:
Kaizhi Yang, Liu Dai, Isabella Liu, Xiaoshuai Zhang, Xiaoyan Sun, Xuejin Chen*, Zexiang Xu, and Hao Su. 2025. IMLS-Splatting: Efficient Mesh Reconstruction from Multi-view Images via Point Representation. *ACM Trans. Graph.* 44, 4 (August 2025), 11 pages. https://doi.org/10.1145/3731210

Corresponding author Xuejin Chen is with MoE Key Laboratory of Brain-inspired Intelligent Perception and Cognition, University of Science and Technology of China. Authors' addresses: Kaizhi Yang, University of Science and Technology of China, Hefei, China, ykz0923@mail.ustc.edu.cn; Liu Dai, University of California San Diego, San Diego, USA, l2dai@ucsd.edu; Isabella Liu, University of California San Diego, San Diego, USA, lal005@ucsd.edu; Xiaoshuai Zhang, Hillbot Inc., San Diego, USA, x@hillbot.ai; Xiaoyan Sun, University of Science and Technology of China, Hefei, China, sunxiaoyan@ustc.edu.cn; Xuejin Chen, University of Science and Technology of China, Hefei, China, xjchen99@ustc.edu.cn; Zexiang Xu, Hillbot Inc., San Jose, USA, zexiangxu@hillbot.ai; Hao Su, University of California San Diego and Hillbot Inc., San Diego, USA, haosu@eng.ucsd.edu.

ACM 0730-0301/2025/8-ART
https://doi.org/10.1145/3731210

1 INTRODUCTION

High-quality mesh Reconstruction from multi-view images is a long-standing challenge in graphics and computer vision, which forms a

critical foundation for downstream tasks such as rendering, scene understanding, and geometric manipulation. Early approaches predominantly rely on multi-view stereo techniques [Agarwal et al. 2009; Snavely et al. 2006], which generate dense point clouds through pixel matching and construct meshes using methods like Delaunay triangulation[Su and Drysdale 1995] or Poisson reconstruction [Kazhdan et al. 2006]. However, these methods are highly sensitive to variations in texture and lighting, limiting their applications.

Recently, advancements in neural rendering technologies have introduced innovative solutions to multi-view reconstruction. Representative works like Neural Radiance Field (NeRF) [Mildenhall et al. 2020] and 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] represent 3D scenes with implicit functions and explicit Gaussian kernels, respectively, optimizing them through volumetric rendering to achieve high-quality photorealistic results. To improve geometric accuracy, subsequent methods such as NeuS [Wang et al. 2021] and 2DGS [Huang et al. 2024] refine scene representations and incorporate additional regularizations, enabling volumetric rendering to closely approximate surface rendering. However, in these approaches, the mesh is typically obtained through a separate post-processing step, which introduces additional geometric and rendering errors that are not accounted for during optimization.

In addition to these volume rendering-based methods, many efforts have been made to optimize surface meshes from multi-view images directly in an end-to-end manner. NvDiffrec [Munkberg et al. 2022] and FlexiCubes [Munkberg et al. 2022] employ a signed distance volume combined with the iso-surfacing algorithms to produce topology-optimizable meshes. While these methods can directly produce high-quality surfaces that are suitable for subsequent geometric processing and photorealistic rendering, their reliance on dense volumetric representations fails to account for the intrinsic sparsity of 3D surfaces, leading to high storage and computational costs. Furthermore, additional regularization losses are usually inevitable to maintain distance field properties and ensure surface smoothness, sacrificing fine details in complex surfaces.

In contrast to dense volumetric representations, point clouds offer spatial sparsity that aligns with the inherent sparsity of 3D surfaces. Their topological flexibility also provides a distinct advantage in representing complex and versatile surfaces. Building upon this, Point-set Surface (PSS) algorithms provide a direct bridge between the points clouds and surfaces. Implicit Moving Least-Squares (IMLS) [Kolluri 2008] is a classical PSS method that combines the signed distance fields of tangent planes at each point to create a global distance field to represent the corresponding shape. Leveraging its concise representation and theoretical guarantees, some methods [Liu et al. 2021; Wang et al. 2024b] have incorporated IMLS into neural geometric reconstruction, achieving superior performance and computational efficiency. However, these methods supervise the IMLS field at discretely sampled points with ground-truth SDF, which fail to generate explicit surface during training, limiting their integration into multi-view reconstruction frameworks.

In this paper, we propose IMLS-Splatting, which integrates IMLS into multi-view reconstruction for end-to-end mesh optimization. To overcome the limitations of previous methods, we propose a splatting-based IMLS construction method that enables differentiable generation of complete and continuous surface mesh from input points. Starting from input points, we construct a 3D grid and identify all grid corners influenced by each point. We then splat each point's information, such as position and normal, onto the corresponding grid vertices and perform IMLS computations. This approach generates a sparse SDF grid that fully covers the surface. Additionally, to support high-quality rendering, we propose an IMLS-based texture computation method that maps each point's texture features to a texture feature field for subsequent rendering.

Inspired by the efficient splatting in 3DGS [Kerbl et al. 2023], we further design a differentiable algorithm for the aforementioned IMLS construction, enabling the fast transformation of 10,000-level points onto a high-resolution SDF grid. By integrating differentiable iso-surfacing and rasterization, we achieve end-to-end mesh reconstruction from multi-view images through point cloud optimization.

Benefiting from the IMLS representation, the inherent topological flexibility of the point cloud enables the reconstructed mesh to undergo topological optimization seamlessly. Furthermore, the weighted averaging property of the IMLS formulation ensures that the reconstructed surface naturally satisfies smoothness requirements without the need for additional regularization losses.

However, due to the limited influence range of each point, which restricts its gradient receptive field, we introduce two optimization strategies—upsampling and resampling—to prevent the optimization from getting stuck in local minima. The upsampling strategy reconstructs the 3D surface progressively, refining it from coarse to fine by upsampling the point cloud. The re-sampling strategy ensures that all points contribute to the surface by resampling new points on the existing surface and discarding outdated ones. Together, these strategies dynamically adjust the point cloud distribution, enabling more robust and effective geometric optimization.

In summary, our work makes the following key contributions:

- We propose IMLS-Splatting, a novel end-to-end mesh reconstruction method from multi-view images, built on a fast differentiable splatting-based IMLS algorithm.
- By incorporating the IMLS formulation, our method enables smooth SDF and detailed mesh reconstruction, without any additional regularization.
- Our approach achieves state-of-the-art performance in geometry reconstruction and delivers superior novel-view synthesis results compared to other mesh-based methods.

2 RELATED WORK

2.1 Implicit rendering representation

Neural implicit representations have garnered significant attention in 3D vision and computer graphics, owing to their ability to provide continuous and highly expressive representations of 3D geometry and appearance. NeRF [Mildenhall et al. 2020] employs a fully connected neural network to model scenes as volumetric radiance fields, enabling photorealistic novel view synthesis. Subsequent works have extended NeRF to overcome its limitations, such as long training times [Chen et al. 2023b; Müller et al. 2022; Sun et al. 2022] and challenges in handling dynamic scenes [Li et al. 2023b; Park et al. 2021; Pumarola et al. 2021]. However, density-based representations struggle to clearly define the underlying 3D surface geometry. To address this, occupancy-based [Niemeyer et al. 2020; Oechsle et al.

2021] and SDF-based [Wang et al. 2021] fields have been developed, offering well-defined surfaces. These approaches reparameterize and integrate the representations into volume rendering for more effective optimization.

To alleviate the inefficiency of MLP-based field queries, some subsequent methods [Jin et al. 2023; Sun et al. 2023; Wu et al. 2023] store geometric and appearance features in a 3D grid. During volume rendering, features at sampling locations can be directly obtained via the interpolation of the grid vertices. To improve surface quality, HF-NeuS [Wang et al. 2022] introduced a displacement field to capture high-frequency surface details, while PET-NeuS [Wang et al. 2023] employed a tri-plane representation to improve the surface resolution. Neuralangelo [Li et al. 2023a] incorporates multi-resolution 3D hash grids to recover detailed SDFs, leveraging a coarse-to-fine training strategy and high-order numerical gradients to enhance the quality of hash-encoded surface reconstruction. However, these methods still require dense spatial sampling for volume rendering, leading to high computational costs and slow rendering efficiency. Additionally, their volumetric representations have limited compatibility with standard 3D pipelines, which rely on meshes. Although meshes can be obtained from the implicit field through an optimization-independent iso-surfacing procedure, this process often introduces additional geometric errors.

2.2 Point-based rendering representation

Leveraging the sparsity and flexibility of point clouds, several methods [Aliev et al. 2020; Kopanas et al. 2021; Lassner and Zollhofer 2021; Wiles et al. 2020; Xu et al. 2022] associate scene information with explicit points to achieve high-quality view synthesis. In particular, 3DGS [Kerbl et al. 2023] introduced anisotropic 3D Gaussians as the primitive, combining them with a fast alpha-blending splatting algorithm for rendering and optimization. This approach enables high-quality results at real-time frame rates. Subsequent methods have focused on improving rendering quality [Lee et al. 2025; Yan et al. 2024; Yu et al. 2024a] and enhancing memory efficiency [Lee et al. 2024; Niedermayr et al. 2024; Ren et al. 2024]. While these point-based representations achieve high-quality novel view synthesis, they often fall short in producing accurate surfaces.

To improve surface quality, DSS [Yifan et al. 2019] employs surface-based splatting and introduces a specialized backpropagation strategy to handle the discontinuities caused by occlusion, ensuring smooth optimization. Subsequent methods of 3DGS either combine Gaussians with SDF [Chen et al. 2023a; Guédon and Lepetit 2024; Lyu et al. 2024] or use more precise Gaussian-Ray intersection calculations [Huang et al. 2024; Zhang et al. 2024] to better align 3D Gaussians with the scene surface. Similar to implicit methods, these approaches also require additional post-processing to extract the mesh, which is not involved in the optimization and may introduce geometric errors. Moreover, such procedures, e.g. TSDF fusion, are often computationally intensive, making them difficult to integrate into an end-to-end optimization framework.

2.3 Mesh-based rendering representation

Early mesh-based differentiable rendering approaches [Luan et al. 2021; Walker et al. 2023; Worchel et al. 2022] typically deform a fixed template mesh, which does not allow for topology optimization. Recent works have introduced implicit SDFs as underlying mesh representations, enabling topology changes during optimization. For instance, Nvdiffrec [Munkberg et al. 2022] uses Deep Marching Tetrahedra (DMTet) [Shen et al. 2021] to differentiably convert the SDF grid into a triangle mesh, while jointly learning environmental lighting and material properties. Nvdiffrecmc [Hasselgren et al. 2022] further enhances Nvdiffrec's capacity to decompose shape, material, and lighting by incorporating ray tracing and Monte Carlo integration. Flexicubes [Shen et al. 2023] extend this concept by introducing additional geometric parameters in Dual Marching Cubes, enabling more flexible optimization of geometry and connectivity, which results in improved geometry quality compared to Nvdiffrec.

Although high-quality mesh that supports rendering can be directly obtained, their reliance on dense volumetric SDF representations wastes storage and computational resources. These methods also suffer from slower training speeds compared to 3DGS-based approaches. In contrast, our method takes advantage of the sparsity and flexibility of point clouds, incorporating the IMLS to convert them into sparse SDFs and meshes for efficient multi-view optimization. Moreover, our parallel splatting algorithm enables fast IMLS computation, achieving training speeds on par with 3DGS methods.

3 ALGORITHM OVERVIEW

Given a set of images of a static scene with corresponding camera poses, our goal is to optimize a 3D mesh for dense reconstruction and high-quality rendering of the target scene. We use point clouds with features as the underlying representation and design a Splatting-based Implicit Moving Least-Squares (IMLS) method to convert the sparse point clouds into an SDF and neural texture field, as shown in Fig. 2. The surface mesh is then extracted using the Marching Cubes. We project the mesh with neural textures into camera views, generating neural texture maps, normal maps, and a foreground mask through differentiable rasterization. These maps are blended into RGB images with deferred shading. The entire process, starting from the point clouds, is differentiable, enabling optimization of the underlying point attributes and guiding the mesh to reconstruct the 3D scene with supervision from the multi-view images.

3.1 Implicit Moving Least-Squares

To reconstruct a surface mesh from a point cloud, we adopt the Implicit moving least-squares (IMLS) [Kolluri 2008] algorithm. IMLS is a classic point-set surface modeling approach that describes an implicit scalar field based on a point cloud. Specifically, from a point cloud $\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^N$ with a normal vector $\mathbf{n}_i \in \mathbb{R}^3$ and influence radius $r_i \in \mathbb{R}^+$ of each point, IMLS defines a field $\mathcal{F}$ where the field value at a specific query point $\mathbf{q}$ is formulated as the weighted average of the signed projection distance of all the points:

$$\mathcal{F}(\mathbf{q}) = \frac{\sum_{\mathbf{p}_i \in \mathcal{P}} \theta(||\mathbf{q} - \mathbf{p}_i||, r_i) \cdot \langle \mathbf{q} - \mathbf{p}_i, \mathbf{n}_i \rangle}{\sum_{\mathbf{p}_i \in \mathcal{P}} \theta(||\mathbf{q} - \mathbf{p}_i||, r_i)}, \tag{1}$$

where weight $\theta(||\mathbf{q} - \mathbf{p}_i||, r_i) = e^{-||\mathbf{q}-\mathbf{p}_i||^2/r_i^2}$. The field $\mathcal{F}$ can be regarded as an SDF approximation of the sample surface of the point cloud [Kolluri 2008], and the 0-level set of $\mathcal{F}$ can be an approximation of the underlying surface

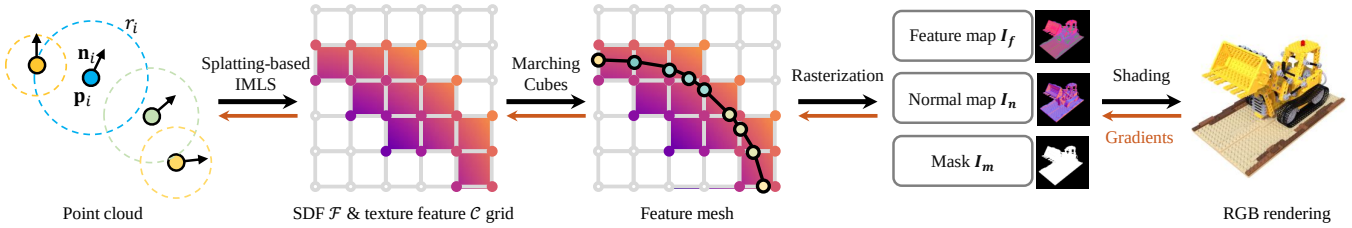


Fig. 2. The overview of our method. We represent 3D scenes using point clouds with neural texture features. Through our fast, splatting-based IMLS approach, the point clouds are converted into SDF and texture feature fields in a differentiable way. Next, 3D meshes with vertex features are extracted using the Marching Cubes algorithm. By integrating differentiable rasterization, our method can be optimized by multi-view image supervision.

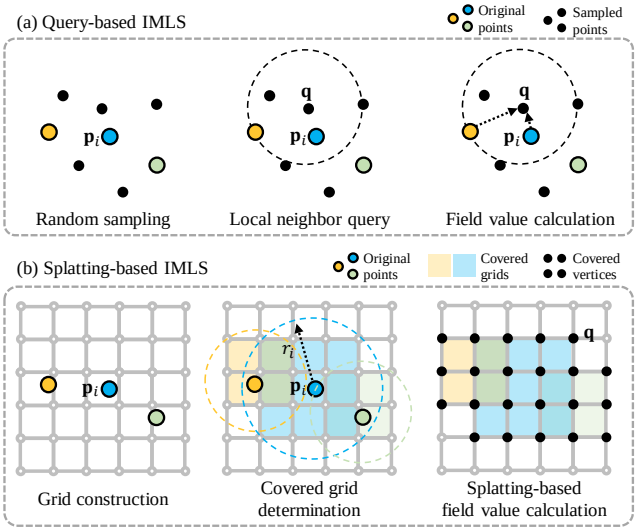


Fig. 3. Comparison between query-based IMLS and splatting-based IMLS. The query-based method starts from the query point, finds its neighboring input points within a fixed query radius r_q, and computes the corresponding field value. In contrast, our splatting-based method starts from the input points and splats the information to all the grid vertices it covers, based on the influence radius r_i.

Existing IMLS-based neural reconstruction methods primarily focus on geometric modeling [Liu et al. 2021; Wang et al. 2024b], requiring ground-truth SDFs for supervision. In contrast, our approach targets point cloud optimization based on multi-view rendering loss, without relying on ground-truth surfaces. To achieve this, we introduce an extra per-point texture feature $\mathbf{c} \in \mathbb{R}^D$ for differentiable neural rendering and multi-view optimization. Then, we build a texture feature field $C(\mathbf{q})$ based on the IMLS formula for an arbitrary query position $\mathbf{q}$:

$$C(\mathbf{q}) = \frac{\sum_{p_i \in \mathcal{P}} \theta(||\mathbf{q} - \mathbf{p}_i||, r_i) \cdot \mathbf{c}_i}{\sum_{p_i \in \mathcal{P}} \theta(||\mathbf{q} - \mathbf{p}_i||, r_i)}. \tag{2}$$

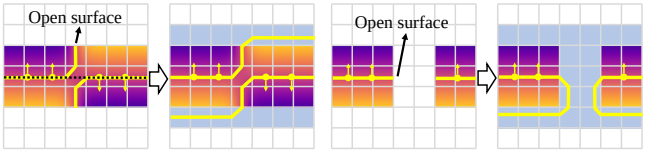
In subsequent steps, the texture feature field will be interpolated onto the extracted surface mesh to model texture. Therefore, each point in $\mathcal{P}$ has $7 + D$ parameters, $(\mathbf{p}_i, \mathbf{n}_i, r_i, \mathbf{c}_i)$, to be optimized during multi-view reconstruction.

3.2 Fast Splatting-based IMLS

Existing IMLS-based neural reconstruction methods [Liu et al. 2021; Wang et al. 2024b] primarily focus on geometric reconstruction by fitting the IMLS field of sampled points to the ground-truth SDF. These methods rely on a query-based approach: for each sampled point $\mathbf{q}$, they query the neighboring region $\Omega(\mathbf{q}, \mathcal{P})$ within a fixed query radius r_q and compute the IMLS field value using the neighboring points $\mathcal{P}$, as shown in Fig. 3 (a) In contrast, we focus on multi-view optimization, which drives point optimization by rendering images rather than GT SDFs. In this context, a complete surface covering all input points should be constructed for subsequent differentiable rendering and optimization. So, a natural solution is to compute the field values on a grid and then extract the surface using an iso-surfacing algorithm.

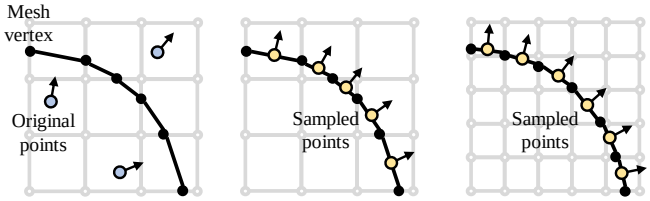
Although the aforementioned query-based approach works for computing the SDF at grid vertices, the varying influence radius r_i for each point means that the choice of query radius r_q greatly impacts the field. A large query radius r_q will introduce irrelevant points and increase the computational load, while a small r_q may overlook key points, leading to discontinuities in the field and hindering optimization. To address these issues, we propose a splatting-based IMLS algorithm that directly splats input points' information into their covered grid vertices. Moreover, we explicitly account for each point's influence radius r_i during splatting. This allows information from points with larger r_i to be splatted over broader spatial regions until the corresponding weights θ decay to a small threshold ϵ. This operation also prevents discontinuities in the generated field. The whole process is visualized in Fig. 3 (b).

To accelerate the splatting process during training and inference, inspired by 3DGS [Kerbl et al. 2023], we design a parallel algorithm to compute SDF and texture fields. Due to the spatial decay property of the weighting function $\theta(||\mathbf{q} - \mathbf{p}_i||, r_i)$, we only consider each point's influence on the grid vertices within its neighborhood. We set the neighborhood radius as $2r_i$. During the forward pass, we first identify all the grid vertexes $V_i = \{\mathbf{v}_j\}$ that lie within the neighborhood of each point $\mathbf{p}_i$, i.e., for any $||\mathbf{v}_j - \mathbf{p}_i|| < 2r_i$. To compute the SDF $\mathcal{F}$ and texture feature fields C at each grid vertex $\mathbf{v}_j$, we need to gather all the input points that exert an influence on $\mathbf{v}_j$. To efficiently perform this gather operation, we instantiate the pair of each input point on each touched grid vertices as a set of $\{(\mathbf{v}_j, \mathbf{p}_i)\}$ pairs and sort all the pairs based on the grid vertex indices j using a fast GPU Radix sort operation[Merrill and Grimshaw 2010]. After sorting, we can obtain an influence point list $\mathcal{N}(\mathbf{v}_j)$ by



(a) Inconsistent normals (b) With background SDF (c) Sparsely covered grid boundary (d) With background SDF

Fig. 4. Illustration of our proposed background SDF for closed surface generation. Given a set of 3D points (yellow) with their normals (yellow arrows), the SDF of their covered grid vertexes can be calculated using IMLS. The purple regions have positive SDF while red represents negative. Compared to the GT surface (black dashed lines), the extracted 0-isosurface (yellow lines) is sensitive to the point normals and grid coverage. Inconsistent point normals (a) and limited point influence radius (c) will lead to open surfaces, which impede stable training. By introducing background SDF (b, d), we can consistently construct outward-facing closed surfaces, facilitating stable surface optimization.



(a) Before Sampling (b) Re-sampling (c) Up-sampling

Fig. 5. Our re-sampling and up-sampling strategy generates more evenly distributed points on the surface and progressively increases the resolution of point clouds and meshes for subsequent optimization.

identifying the first and last pair of $\mathbf{v}_j$. Subsequently, we compute the SDF value and texture field value at each grid vertex with the corresponding point list in parallel based on Eq. (1) and Eq. (2).

In the backward pass, we construct the gradient flow from each grid vertex's field value to the attributes of the input points. To improve efficiency, we store intermediate variables computed during the forward pass for direct use in the backward pass. In each thread, we accumulate all the gradients from SDF $\frac{\partial \mathcal{L}}{\partial \mathcal{F}(\mathbf{q}_j)}$ and texture feature field $\frac{\partial \mathcal{L}}{\partial \mathcal{C}(\mathbf{q}_j)}$ within the covered vertices of one point to get its attribute gradient. Both the forward and backward passes are parallelized and accelerated using CUDA.

3.3 Iso-surfacing with Background SDF

After obtaining the SDF and texture features on the 3D grid, we use the Marching Cubes algorithm [Lorensen and Cline 1987] to extract the surface and interpolate the texture features on the surface. However, with nonideal IMLS points and noisy normals, particularly at the early stages of the optimization process, constructing a mesh on such a sparse grid may lead to the following issues:

- Since the normals of the input points are estimated during the iterative optimization process and do not guarantee consistent directions, artifacts can arise when merging the signed projection distance of points with varying orientations on a sparsely covered grid. As shown in Fig. 4 (a), inconsistent point cloud normals can create open surfaces that are perpendicular to the actual surface, and are difficult to eliminate through optimization.
- Limited coverage of the IMLS points may result in truncated open surfaces at the boundaries of the covered grids, as Fig. 4 (c) shows.

The normals of open surfaces are ambiguous, making it challenging to maintain consistency during optimization. On the other hand, the mesh vertices at the boundary of open surfaces lack adequate neighborhood information, which can result in unstable gradients. To address this, we propose to dilate a background positive distance field around the original sparsely covered grid, ensuring that watertight surfaces can be generated. Since our algorithm extracts only the 0 level-set for subsequent rendering, adding the background positive SDF will not affect regions with positive values at the boundary of the original distance field. However, if the boundary has negative values, a new outward-facing surface will be formed between the background and the original field. As shown in Fig. 4 (b,d), by incorporating the background SDF, the original open surfaces are converted into outward-facing watertight surfaces, facilitating stable surface optimization.

3.4 Neural Shading Model

We use Nvdiffrast[Laine et al. 2020] to rasterize the mesh with neural textures into the camera view and produce a foreground mask $\mathbf{I}_m \in \mathbb{R}^{H\times W}$, normal map $\mathbf{I}_n \in \mathbb{R}^{H\times W\times 3}$, and 2D feature map $\mathbf{I}_f \in \mathbb{R}^{H\times W\times D}$. For the normal map, previous methods [Huang et al. 2024; Verbin et al. 2022] derive normal maps based on volume rendering, by integrating either the gradient directions of the SDF or MLP-predicted normal directions. In contrast, we directly compute the vertex normals by averaging the normals of its surrounding triangles and rasterize them into a normal map. This operation explicitly associates the mesh normals with the rendered images, facilitating our surface optimization.

To better disentangle the geometry and view-dependent appearances, we follow Ref-NeRF [Verbin et al. 2022] to decode the feature map $\mathbf{I}_f$ into a collection of spatially-varying scene properties for each pixel, including a diffuse color $\mathbf{c}_d$, and a specular tint $\mathbf{s}$, and a specular feature $\mathbf{f}_s$ by an MLP Ψ. Previous works [Han and Xiang 2023; Verbin et al. 2022] use complex directional functions to construct the reflected radiance from specular feature $\mathbf{f}_s$, which is computationally expensive. To improve the training efficiency, we directly train a tiny MLP Φ to generate the specular color according to the reflection direction ω_r. In order to model complex interreflections and occlusions, we also input the view directions ω of each pixel to the MLP Φ to model more complex view-independent effects. Consequently, the final color value is obtained as:

$$\mathbf{c} = \mathbf{c}_d + \mathbf{s} \odot \Phi(\mathbf{f}_s, \omega, \omega_r), \tag{3}$$

where ω is obtained by calculating the direction vector from the camera to the 3D point located on the surface mesh, and the reflect direction $\omega_r = 2(-\omega \cdot \mathbf{n})\mathbf{n} + \omega$. The normal $\mathbf{n}$ is from the normal map $\mathbf{I}_n$. A rasterized color image $\mathbf{I}$ can be finally obtained by multiplying the foreground mask $\mathbf{I}_m$ to the color $\mathbf{c}$ of each pixel.

3.5 Loss Function

Starting from the point cloud, we have developed a complete mesh reconstruction and differentiable rendering pipeline, which enables

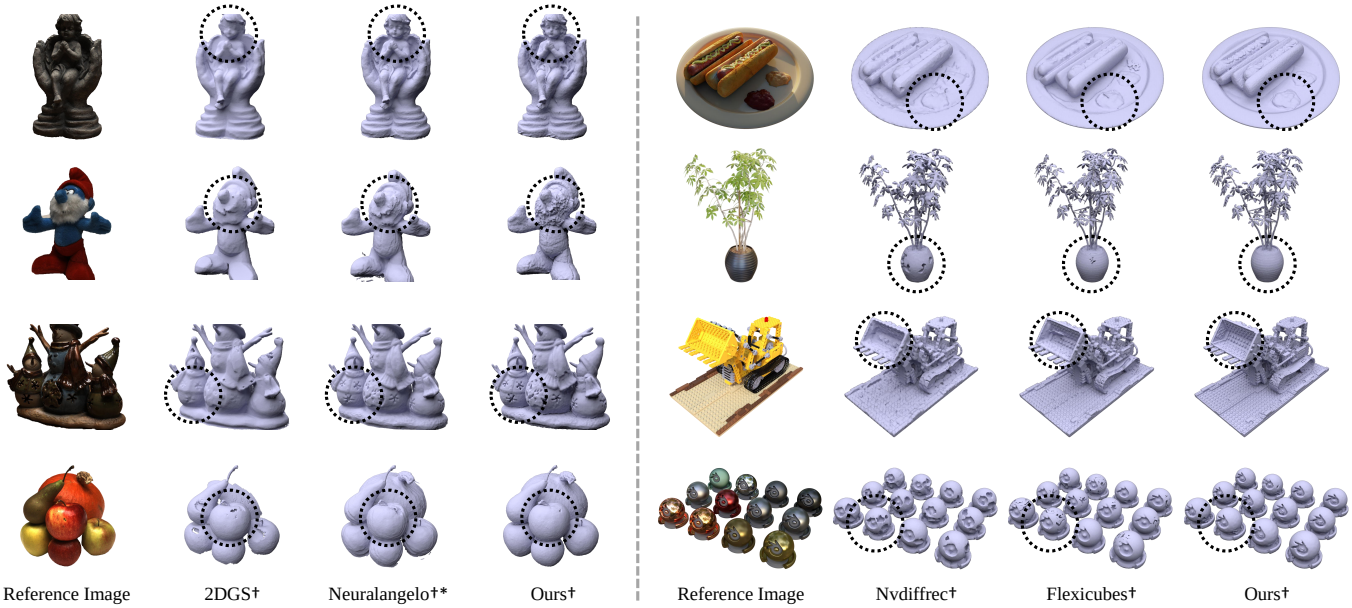


Fig. 6. Qualitative comparison of reconstruction results on the DTU and NeRF datasets.

us to optimize the point cloud $\mathcal{P}$ with $7+D$ parameters $(\mathbf{p}_i, \mathbf{n}_i, r_i, \mathbf{c}_i)$ for each point according to the difference between the rendered image $\mathbf{I}$ and the ground-truth image $\mathbf{I}_{gt}$. The loss function contains a pixel-level L1 term and a D-SSIM term:

$$\mathcal{L} = (1-\lambda)\mathcal{L}_1(\mathbf{I}, \mathbf{I}_{gt}) + \lambda\mathcal{L}_{ssim}(\mathbf{I}, \mathbf{I}_{gt}). \tag{4}$$

λ is set as 0.2 in all our experiments. Additionally, we focus on the foreground reconstruction for the image data where the GT foreground masks are available. In our experiments, we remove the background pixels and fill the background with random colors in each iteration to reduce ambiguity between the foreground and the background.

Unlike previous implicit SDF-based methods (like NeuS [Wang et al. 2021]) or point-based methods (like 2DGS [Huang et al. 2024]) that require additional regularization losses, such as Eikonal loss or depth distortion loss, the IMLS representation theoretically guarantees the smoothness and stability of the constructed field. This allows us to obtain highly detailed SDF and surface mesh without extra regularization.

3.6 Coarse-to-fine Optimization

Since the IMLS surface is a weighted average of the tangent plane of local points, thus these points do not necessarily lie on the generated surface strictly during training. Point noises will also lead to isolated small surface meshes. To solve these problems, we propose a point re-sampling strategy. Specifically, we first calculate the visibility of each triangle in the reconstructed mesh from the current point set. Triangles that can not be observed from any training view are discarded. Next, we re-sample a new point set on the remaining triangles by picking the centroids of all mesh triangles. The texture features $\mathbf{c}_i$ of the new points are set to the average of the features of the triangle's vertices, while the point normal $\mathbf{n}_i$ is directly set as the triangle normal. Since these triangles are generated through the Marching Cubes, the new points are uniformly distributed across the reconstructed surface. The new influence radius r_i is initially set as the voxel size v in the 3D grid. To encourage locality, we restrict the influence radius $r_i < 1.5v$ using a sigmoid operation during optimization. After resampling, we discard all previous points and use the newly sampled points for subsequent optimization. The resampling process is illustrated in Fig.5 (b). The resampled point set are accurately located and more evenly distributed on the surface.

On the other hand, each point has a limited influence range, which restricts its gradient receptive field and may result in geometric local minima. To progressively recover more geometric details and avoid local minima, we design an upsampling strategy that gradually increases the grid resolution for coarse-to-fine optimization. As Fig.5 (c) shows, based on a higher-resolution grid, the number of triangles on the extracted mesh also grows, leading to more resampled points for finer reconstruction.

4 EXPERIMENTS

We evaluate both the surface reconstruction and novel view synthesis capabilities of our method. In the following sections, we first introduce the implementation details. Then, we show quantitative and qualitative comparisons with other state-of-the-art methods. Finally, ablation studies are reported to validate the effectiveness of our specific designs.

4.1 Implementation

We initialize with a coarse point cloud obtained by 7,000 iterations of 3DGS, which costs about one minute. The shortest axis of each Gaussian is used as its normal, facing toward the camera which produces the highest alpha-blending weight. The texture features of each point $\mathbf{c}$ and the parameters of MLPs Φ and Ψ in the shading

Table 1. Comparison of Chamfer Distance on DTU dataset for mesh quality evaluation. We compare with implicit methods, Gaussian-based methods, and mesh-based methods. †indicates that the object masks were utilized during the training, and * indicates the reproduced result using the official code. All 3DGS-based methods and ours start from COLMAP points, as we use early-stage coarse 3DGS points as initialization. We color each cell as best, second best, and thrid best.

	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	**Avg.**	**Time**
VolSDF [Yariv et al. 2021]	1.14	1.26	0.81	0.49	1.25	0.70	0.72	1.29	1.18	0.70	0.66	1.08	0.42	0.61	0.55	0.86	>12h
NeuS [Wang et al. 2021]	1.00	1.37	0.93	0.43	1.10	0.65	0.57	1.48	1.09	0.83	0.52	1.20	0.35	0.49	0.54	0.84	>12h
Neuralangelo [Li et al. 2023a]	0.37	0.72	0.35	0.35	0.87	0.54	0.53	1.29	0.97	0.73	0.47	0.74	0.32	0.41	0.43	0.61	>12h
Neuralangelo* [Li et al. 2023a]	0.41	0.85	0.40	0.33	0.89	0.66	1.88	1.42	1.83	0.80	0.45	1.40	0.34	0.94	0.86	0.90	>12h
SuGaR [Guédon and Lepetit 2024]	1.47	1.33	1.13	0.61	2.25	1.71	1.15	1.63	1.62	1.07	0.79	2.45	0.98	0.88	0.79	1.33	52m
2DGS [Huang et al. 2024]	0.48	0.91	0.39	0.39	1.01	0.83	0.81	1.36	1.27	0.76	0.70	1.40	0.40	0.76	0.52	0.80	9m
GoF [Yu et al. 2024b]	0.50	0.82	0.37	0.37	1.12	0.74	0.73	1.18	1.29	0.68	0.77	0.90	0.42	0.66	0.49	0.74	18m
NeuS†[Wang et al. 2021]	0.83	0.98	0.56	0.37	1.13	0.59	0.60	1.45	0.95	0.78	0.52	1.43	0.36	0.45	0.45	0.77	>12h
Neuralangelo†* [Li et al. 2023a]	0.45	0.74	0.33	0.34	1.05	0.54	0.53	1.33	1.05	0.72	0.43	0.69	0.34	0.38	0.42	0.62	>12h
2DGS†[Huang et al. 2024]	0.46	0.84	0.31	0.45	0.92	1.01	0.83	1.23	1.30	0.66	0.61	1.07	0.45	0.71	0.54	0.76	9m
Nvdiffrec†[Munkberg et al. 2022]	3.04	3.02	2.10	0.78	2.18	1.60	1.46	1.67	2.85	1.26	1.10	3.26	1.13	1.31	1.19	1.86	>1h
Flexicubes†[Shen et al. 2023]	1.66	1.60	0.91	0.50	2.60	1.32	0.87	1.45	1.60	1.38	0.73	1.85	1.01	0.64	0.75	1.26	>1h
Ours†	0.52	1.02	0.37	0.32	0.86	0.50	0.48	1.15	0.76	0.59	0.37	0.67	0.33	0.33	0.34	0.57	11 m

module are randomly initialized. For the differentiable Marching Cubes, we employ the CUDA-based implementation developed by NeuManifold [Wei et al. 2023]. The original implementation only supports surface mesh construction. We additionally add a feature interpolation operation based on the texture field and the corresponding gradient computation for optimization.

In all experiments, we perform iterative optimization for 20,000 steps per scene. The up-sampling operation is performed every 4,000 iterations until 12,000 iterations and the grid resolution starts at 64^3 and increases by 64 at each step until reaching 256^3. Re-sampling begins at 3,000 iterations and performs every 1,000 iterations until 15,000 iterations. All experiments are conducted on a single NVIDIA RTX 3090 GPU. Apart from the CUDA-based components, the remaining parts are implemented using the PyTorch framework.

Two common datasets, the NeRF-synthetic and DTU datasets, are used for evaluation. The NeRF-synthetic dataset [Mildenhall et al. 2020] contains a total of 8 object-level scenes. Each scene includes 100 training images and 200 test images, with a resolution of 800 × 800. The DTU dataset [Jensen et al. 2014] contains 15 real-world objects, each with 49 or 69 images at a resolution of 1600×1200. Following previous works, we downsample these images to a resolution of 800 × 600 for training and evaluation.

4.2 Surface reconstruction

We evaluate the surface reconstruction quality of the proposed method by comparing with several state-of-the-art approaches, including implicit methods (VolSDF [Yariv et al. 2021], NeuS [Wang et al. 2021], and Neuralangelo [Li et al. 2023a]), Gaussian-based methods (SuGaR [Guédon and Lepetit 2024], 2DGS [Huang et al. 2024], and GoF [Yu et al. 2024b]), and mesh-based methods (Nvdiffrec [Munkberg et al. 2022] and Flexicubes [Shen et al. 2023]).

In particular, mesh-based reconstruction approaches rely on differentiable rasterization for optimization and require object masks to specify the foreground. While Nvdiffrec and Flexicubes incorporate an additional mask loss, our method achieves the same goal by randomly assigning background colors during training. In contrast, the other two categories of methods rely on volumetric rendering for optimization and generally do not require masks. To ensure a fair comparison, we not only compare their results with standard settings, but also select some representative works from each category (NeuS, Neuralangelo, and 2DGS) and add the mask loss to generate results for comparison. For Nvdiffrec and Flexicubes, we compare their results under the manifold setting by discarding their vertex refinement, same as ours.

Table 1 compares the surface reconstruction quality of the proposed method with other approaches on the DTU real-world dataset. Our method achieves the lowest average Chamfer Distance (CD). For Neuralangelo, we reproduced its results using the official code (denoted as Neuralangelo*), obtaining an average Chamfer Distance (CD) of 0.90, which is similar to the reproduced results reported in prior works [Li et al. 2024; Wang et al. 2024a]. Building upon this, we further incorporate an additional mask loss (Neuralangelo†*), improving the average CD to 0.62. Under the same setting, our method achieves a lower average CD of 0.57 while requiring only 11 minutes of training on average, which is significantly faster than Neuralangelo (over 12 hours). When compared to NeuS and 2DGS under the foreground mask setting, our method achieves a notably lower CD (0.57) than those of NeuS (0.77) and 2DGS (0.76).

Table 2. Comparison of Chamfer distance on NeRF synthetic dataset.

	Mic	Chair	Ship	Materials	Lego	Drums	Ficus	Hotdog	**Avg.**
NeuS†	0.53	0.37	1.25	0.35	0.83	0.86	0.40	0.96	0.69
Neuralangelo†*	0.52	0.29	0.34	0.85	0.61	0.77	0.54	0.87	0.60
2DGS†	1.39	0.92	1.92	1.11	0.95	1.63	2.65	2.00	1.57
Nvdiffrec†	0.52	0.45	2.48	0.73	0.71	1.33	0.61	0.69	0.94
Flexicubes†	0.46	0.44	5.23	0.55	0.70	1.05	0.26	0.68	1.17
Ours†	0.44	0.29	0.99	0.39	0.54	0.66	0.36	0.54	0.53

Compared to the DTU dataset, the NeRF synthetic dataset is more challenging due to its intricate appearance and diverse reflective properties. Despite these challenges, our method still achieves consistently high-quality reconstruction across eight scenes and also obtains the best average results, as shown in Table 2.

Fig. 6 presents a visual comparison of surface reconstructions on the DTU dataset and NeRF dataset. 2DGS tends to produce overly

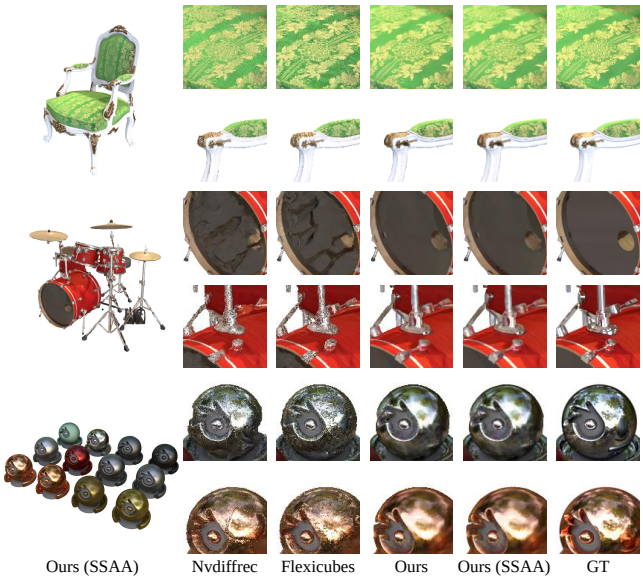


Fig. 7. Qualitative rendering comparison with mesh-based methods. Our method effectively captures details of high-frequency appearance, achieving superior rendering quality. By incorporating SSAA, edge aliasing can be further reduced, leading to an additional improvement.

smoothed surfaces and exhibits noticeable holes in reflective areas. Compared to Neuralangelo, our method better preserves fine surface details, even in challenging reflective regions. Compared to Nvdiffrec and Flexicubes, which also utilize surface mesh rasterization for optimization, our method not only significantly reduces training time but also achieves marked improvements in geometric quality.

4.3 Novel-view synthesis

Due to the significant differences in the ability of volumetric rendering and mesh rendering methods to capture high-frequency appearance details, we compare our method with two mesh-driven approaches: Nvdiffrec and Flexicubes. We conduct experiments on eight scenes from the NeRF synthetic dataset. The average results in Table 3 demonstrate that our method outperforms the other mesh-based method, achieving higher PSNR and SSIM scores, and lower LPIPS across multiple scenes. This highlights its superior ability to capture appearance details and model complex textures.

We also report the rendering metrics of 2DGS and two 3DGS models in Table 3 for reference. The "3DGS-7k" indicates the 3DGS model trained for 7,000 iterations, whose points are used to initialize our method, and "3DGS" indicates the full model trained for 30,000 iterations. Although our approach outperforms other mesh-based methods, it still lags behind 3DGS in PSNR. We analyze the reasons as follows:

- Volume rendering methods like 3DGS do not strictly constrain surface geometry, allowing more flexible appearance modeling but sacrificing surface quality. In contrast, our method enforces surface constraints and achieves improved geometric quality.
- We only use two lightweight MLPs for efficiency in light field modeling. Employing a more advanced shading model could further enhance our quality.

It is worth noting that the limited resolution of the surface mesh during rasterization can cause aliasing, potentially skewing evaluation metrics. To mitigate this effect, we also provide a comparison using additional supersample anti-aliasing (SSAA) during evaluation, which significantly reduces aliasing artifacts. Fig.7 shows a visual comparison of the novel-view rendering results. Our method not only better captures high-frequency details, such as the texture patterns on the chair, but also delivers superior results in high-reflection areas, like the ball example. Furthermore, all these mesh-based rendering methods exhibit aliasing artifacts along the object boundary. The use of SSAA effectively mitigates these issues, enhancing overall rendering quality.

Table 3. Rendering quality comparison on the NeRF synthetic dataset.

	PSNR ↑	SSIM ↑	LPIPS ↓
3DGS-7k	31.29	0.96	0.04
3DGS	33.33	0.97	0.03
2DGS	33.06	0.97	0.03
Nvdiffrec	26.87	0.93	0.09
Flexicubes	27.50	0.93	0.08
Ours	28.38	0.95	0.06
Ours (SSAA)	29.16	0.95	0.05

4.4 Ablation Study

Why IMLS representation? Compared to other mesh methods like Nvdiffrec and Flexicubes, which directly optimize SDF values on a 3D grid, our approach leverages IMLS to bridge points with the SDF, resulting in improved mesh reconstruction. To validate the advantage of our method using IMLS for SDF, we conducted an ablation study by removing IMLS during optimization, thus the SDF and texture field values are directly optimized on the 3D grid. For a fair comparison, we first optimize the SDF and texture field using the full pipeline with IMLS for 500 iterations, which is the same as our main experiment, to obtain a coarse SDF field as initialization. Subsequently, the IMLS computation was disabled, and the SDF values of the 3D grid vertexes are directly optimized.

As shown in Fig.8 (a) and (d), without the inherent regularization provided by IMLS, the optimized surface exhibits similar results to Nvdiffrec and Flexicubes, with numerous holes and fractures. In contrast, our complete pipeline could produce accurate and high-detail surfaces. Table 4 presents a CD comparison between the two setups on the NeRF synthetic dataset. The IMLS-based optimization consistently outperforms the grid-based optimization in most scenarios, achieving an average Chamfer Distance of 0.53 compared to 0.75 for the volume-based approach. On the DTU dataset, the IMLS-based approach achieves an average Chamfer Distance of 0.57, outperforming the volume-based approach, which records 0.69.

Notably, integrating our fast IMLS method into the training process barely adds extra time. On the DTU dataset, the average optimization time for the IMLS-based setup is 11 minutes, compared to 10 minutes for the grid-based approach. This efficiency will be

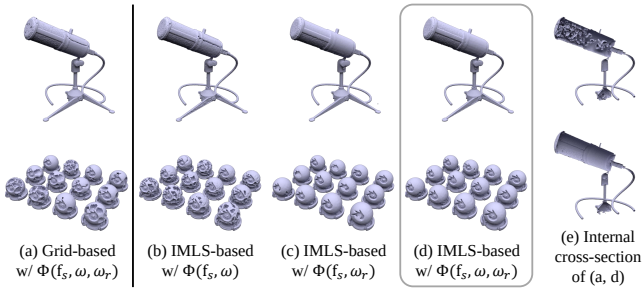


Fig. 8. Results visualization of ablations on grid/IMLS and shading models.

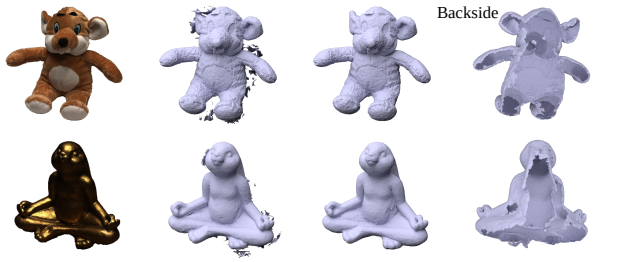


Fig. 9. The impact of background SDF on reconstruction results. With the background SDF, discontinuities and geometric artifacts can be avoided during optimization. After training, an open surface can be obtained by simply removing the background SDF.

further discussed in Section 4.4. Furthermore, since our generated mesh is tightly coupled with the point cloud, irrelevant surfaces can be directly removed by deleting invisible points, as described in the resampling operation in Sec. 3.6. This results in a clean internal structure, as shown in Fig. 8 (e), which is difficult to achieve with grid-based optimization.

Table 4. Comparison of CD between grid-based reconstruction and IMLS-based reconstruction.

	Mic	Chair	Ship	Materials	Lego	Drums	Ficus	Hotdog	**Avg.**
Grid-based	0.45	0.28	2.03	0.85	0.56	0.70	0.59	0.56	0.75
IMLS-based	0.44	0.29	0.99	0.39	0.54	0.66	0.36	0.54	0.53

Impact of shading models. Our method is compatible with a variety of shading approaches. However, the shading model more or less impacts the quality of the surface reconstruction and rendering. In this work, we decompose the pixel color into view-independent diffuse and view-dependent specular components. We model the specular component using an MLP Φ that incorporates both the reflection direction ω_r and the viewing direction ω to capture complex near-field lighting effects. To validate its effectiveness on geometric optimization, we compare view-based modeling $\Phi(\mathbf{f}_s, \omega)$, reflection-based modeling $\Phi(\mathbf{f}_s, \omega_r)$, and a fusion model $\Phi(\mathbf{f}_s, \omega_r, \omega)$. As shown in Fig. 8 and Table 5, relying solely on view-based modeling requires more complex surfaces to approximate specular highlights (b), leading to poorer reconstruction quality. In contrast, reflection-based modeling yields smoother surfaces but struggles to capture high-frequency appearance details (c). By combining both approaches, we achieve high-quality surface meshes and rendered images (d).

Table 5. The impact of shading models Φ on reconstruction quality. The model $\Phi(\mathbf{f}_s, \omega, \omega_r)$ which considers both the reflection direction ω_r and view direction ω achieves the best reconstruction and rendering quality.

Shading models	CD ↓	PSNR ↑	SSIM ↑	LPIPS ↓
$\Phi(\mathbf{f}_s, \omega)$	0.69	28.18	0.949	0.063
$\Phi(\mathbf{f}_s, \omega_r)$	0.59	27.87	0.945	0.064
$\Phi(\mathbf{f}_s, \omega, \omega_r)$	0.53	28.38	0.954	0.060

Background SDF and open surface reconstruction. By introducing a background SDF around the point cloud, the generated geometric surface is guaranteed to be closed and watertight, significantly improving the optimization stability. This eliminates issues such as ambiguous normal directions and gradient discontinuities caused by many open surfaces, as shown in Fig. 9 (b). In our case, the background SDF facilitates the optimization and can be removed after training, enabling direct reconstruction of open surfaces. This flexible design allows our method to reconstruct both closed and open surfaces, as shown in Fig. 9.

Time complexity and memory analysis. Our efficient splatting algorithm enables the fast conversion of point clouds into the corresponding SDF and texture fields. In Table 6, we report the time and memory consumption per iteration during training on the Lego scene across different grid resolutions. As the resolution increases from 64^3 to 384^3, both the number of points and faces grow according to our resampling and upsampling strategy. Despite the large point count, our method remains highly efficient — processing around 1 million points at 384^3 resolution, generating 2.27M faces, and consuming 19.1GB GPU memory, with IMLS and Marching Cubes each taking approximately 12 ms per iteration.

Table 6. Time and GPU memory cost of each module in a single iteration. Our IMLS-Splatting algorithm demonstrates consistent efficiency in converting point clouds into the corresponding fields in various grid resolutions.

Grid resolution	64^3	128^3	192^3	256^3	320^3	384^3
GPU memory	6.4G	7.4G	7.8G	10.0G	13.3G	19.1G
Point number	27k	113k	260k	471k	729k	1.03M
Face number	36k	173k	427k	849k	1.45M	2.27M
IMLS splatting	0.50 ms	1.04 ms	2.53 ms	4.80 ms	7.72 ms	11.53 ms
Marching cubes	0.90 ms	1.15 ms	1.85 ms	3.89 ms	7.22 ms	12.19 ms
Rasterization	0.84 ms	1.21 ms	1.87 ms	3.01 ms	4.20 ms	5.73 ms
Shading	5.85 ms	5.97 ms	6.33 ms	6.92 ms	8.15 ms	10.38 ms

Point cloud and SDF visualization. Besides the final surface mesh, we also visualize the point cloud and SDF generated by IMLS after optimization in Fig. 10. It can be observed that the point cloud closely follows the object surface and is evenly distributed. Moreover, benefiting from the theoretical guarantees of IMLS, our method does not require additional constraints, such as the commonly used Eikonal loss, to construct the SDF field. This eliminates potential issues like slower convergence or loss of detail that can arise from introducing extra regularization during training. The resulting SDF field also accurately captures the geometric and topological features of the target object while maintaining smoothness and continuity.

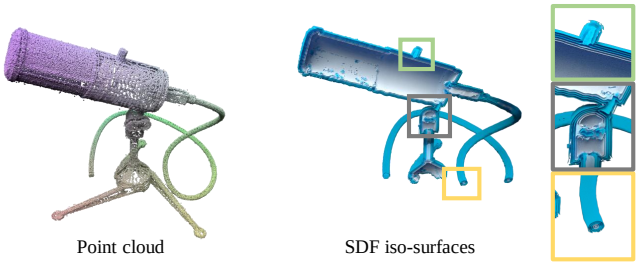


Fig. 10. Visualization of point cloud and SDF iso-surfaces after training. Without regularization, our method can produce a uniform point cloud on the surface with correct normals and an accurate SDF near the surface.

Limitations. Although the proposed method is a general-purpose multi-view reconstruction approach, its scalability is somewhat limited by the grid resolution and GPU memory, as shown in Table 6, making it challenging for large-scale scenes. To further illustrate this, we conduct experiments on the Mip-NeRF 360 dataset [Barron et al. 2022]. To handle the unbounded nature of the scenes, we applied the contraction function (Eq. (5)) proposed in Mip-NeRF 360 to compress the point cloud into a unit sphere, which was then fed into our splatting-based IMLS and optimization pipeline. Based on the contraction, the background space is compressed more aggressively into fewer voxels. Performing resampling on such a grid can reduce the background points number, thereby lowering the overall computational cost and enabling training at 448^3 resolution.

$$contract(\mathbf{x}) = \begin{cases} \mathbf{x}, & \|\mathbf{x}\| \le 1 \\ \left(2 - \frac{1}{\|\mathbf{x}\|}\right)\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right), & \|\mathbf{x}\| > 1 \end{cases} \tag{5}$$

We present several results in Fig. 11, showcasing visually plausible renderings and foreground meshes. However, due to limitations in grid resolution, the reconstructed foreground meshes remain relatively coarse. Additionally, the use of the contraction function introduces distortions in the background mesh, which could be potentially alleviated by incorporating a dedicated background modeling module. For larger-scale scenarios, like autonomous driving, this resolution is far from sufficient. Advanced sparse data structures (e.g., fVDB [Williams et al. 2024]) could be a potential solution, and we leave it for future work.

5 CONCLUSIONS

In this paper, we propose IMLS-Splatting, a novel multi-view mesh reconstruction method. Considering that geometric surfaces are typically sparsely distributed in 3D space, we leverage the sparsity and flexibility of point clouds to reconstruct surfaces. To further enhance efficiency, we propose a splatting-based Implicit Moving Least-squares algorithm to convert the point cloud to the implicit fields in real time, which can be integrated with subsequent differentiable rendering modules for multi-view reconstruction. By incorporating the IMLS formulation, our method reconstructs smooth and detailed meshes without additional regularization, supporting both open and closed surfaces. Our approach achieves SOTA performance in geometry reconstruction and delivers superior novel-view synthesis results compared to other mesh-based methods.

(a) Reference Image (b) Rendering Image (c) Reconstruction mesh

Fig. 11. Results visualization on MipNeRF-360 dataset.

ACKNOWLEDGMENTS

This work was in part supported by the National Natural Science Foundation of China under grants 62032006, 62021001 and 62472399.

REFERENCES

Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. 2009. Building Rome in a day. In *The IEEE/CVF International Conference on Computer Vision*.

Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. 2020. Neural Point-Based Graphics. In *The European Conference on Computer Vision*. 696–712.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2023b. TensoRF: Tensorial Radiance Fields. In *The European Conference on Computer Vision*. 333–350.

Hanlin Chen, Chen Li, and Gim Hee Lee. 2023a. NeuSG: Neural Implicit Surface Reconstruction with 3D Gaussian Splatting Guidance. In *arXiv preprint arXiv:2312.00846*.

Antoine Guédon and Vincent Lepetit. 2024. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5354–5363.

Kang Han and Wei Xiang. 2023. Multiscale Tensor Decomposition and Rendering Equation Encoding for View Synthesis. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4232–4241.

Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. 2022. Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. In *Advances in Neural Information Processing Systems*, Vol. 35. 22856–22869.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH Conference Papers*. Article 32, 11 pages.

Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. 2014. Large scale multi-view stereopsis evaluation. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 406–413.

Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. 2023. TensoIR: Tensorial Inverse Rendering. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 165–174.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *The Eurographics Symposium on Geometry Processing*.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4, Article 139 (July 2023), 14 pages.

Ravikrishna Kolluri. 2008. Provably good moving least squares. *ACM Transactions on Algorithms* 4, 2 (2008).

Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. 2021. Point-Based Neural Rendering with Per-View Optimization. *Computer Graphics Forum* 40, 4 (2021), 29–43.

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics* 39, 6, Article 194 (Nov. 2020), 14 pages.

Christoph Lassner and Michael Zollhofer. 2021. Pulsar: Efficient Sphere-Based Neural Rendering. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1440–1449.

Byeonghyeon Lee, Howoong Lee, Xiangyu Sun, Usman Ali, and Eunbyung Park. 2025. Deblurring 3D Gaussian Splatting. In *The European Conference on Computer Vision*. 127–143.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. 2024. Compact 3D Gaussian Representation for Radiance Field. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21719–21728.

Kunyi Li, Michael Niemeyer, Zeyu Chen, Nassir Navab, and Federico Tombari. 2024. MonoGSDF: Exploring Monocular Geometric Cues for Gaussian Splatting-Guided Implicit Surface Reconstruction. In *arXiv preprint arXiv:2411.16898*.

Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H. Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. 2023a. Neuralangelo: High-Fidelity Neural Surface Reconstruction. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8456–8465.

Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. 2023b. DynIBaR: Neural Dynamic Image-Based Rendering. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4273–4284.

Shi-Lin Liu, Hao-Xiang Guo, Hao Pan, Pengshuai Wang, Xin Tong, and Yang Liu. 2021. Deep Implicit Moving Least-Squares Functions for 3D Reconstruction. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

William E. Lorensen and Harvey E. Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH Conference Papers*.

Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. 2021. Unified Shape and SVBRDF Recovery using Differentiable Monte Carlo Rendering. *Computer Graphics Forum* 40, 4 (2021), 101–113.

Xiaoyang Lyu, Yang-Tian Sun, Yi-Hua Huang, Xiuzhe Wu, Ziyi Yang, Yilun Chen, Jiangmiao Pang, and Xiaojuan Qi. 2024. 3DGSR: Implicit Surface Reconstruction with 3D Gaussian Splatting. In *arXiv preprint arXiv:2404.00409*.

Duane G. Merrill and Andrew S. Grimshaw. 2010. Revisiting sorting for GPGPU stream architectures. In *Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques*. 545–546.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *The European Conference on Computer Vision*. 405–421.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics* 41, 4, Article 102 (July 2022), 15 pages.

Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. 2022. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8280–8290.

Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. 2024. Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10349–10358.

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. 2020. Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Michael Oechsle, Songyou Peng, and Andreas Geiger. 2021. UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction. In *The IEEE/CVF International Conference on Computer Vision*. 5589–5599.

Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. 2021. HyperNeRF: a higher-dimensional representation for topologically varying neural radiance fields. *ACM Transactions on Graphics* 40, 6, Article 238 (Dec. 2021), 12 pages.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10318–10327.

Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. 2024. Octree-GS: Towards Consistent Real-time Rendering with LOD-Structured 3D Gaussians. In *arXiv preprint arXiv:2403.17898*.

Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. 2021. Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis. In *Advances in Neural Information Processing Systems*, Vol. 34. 6087–6101.

Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. 2023. Flexible Isosurface Extraction for Gradient-Based Mesh Optimization. *ACM Transactions on Graphics* 42, 4, Article 37 (July 2023), 16 pages.

Noah Snavely, Steven M. Seitz, and Richard Szeliski. 2006. Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics* 25, 3 (2006).

Peter Su and Robert L. Scot Drysdale. 1995. A comparison of sequential Delaunay triangulation algorithms. In *The Annual Symposium on Computational Geometry*.

Cheng Sun, Guangyan Cai, Zhengqin Li, Kai Yan, Cheng Zhang, Carl Marshall, Jia-Bin Huang, Shuang Zhao, and Zhao Dong. 2023. Neural-PBIR Reconstruction of Shape, Material, and Illumination. In *The IEEE/CVF International Conference on Computer Vision*. 18046–18056.

Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct Voxel Grid Optimization: Super-Fast Convergence for Radiance Fields Reconstruction. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5459–5469.

Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. 2022. Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5491–5500.

Thomas Walker, Octave Mariotti, Amir Vaxman, and Hakan Bilen. 2023. Explicit Neural Surfaces: Learning Continuous Geometry with Deformation Fields. In *Advances in Neural Information Processing Systems Workshop*.

Fangjinhua Wang, Marie-Julie Rakotosaona, Michael Niemeyer, Richard Szeliski, Marc Pollefeys, and Federico Tombari. 2024a. UniSDF: Unifying Neural Representations for High-Fidelity 3D Reconstruction of Complex Scenes with Reflections. In *Advances in Neural Information Processing Systems*.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In *Advances in Neural Information Processing Systems*, Vol. 34. 27171–27183.

Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. 2022. HF-NeuS: Improved Surface Reconstruction Using High-Frequency Details. In *Advances in Neural Information Processing Systems*, Vol. 35. 1966–1978.

Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. 2023. PET-NeuS: Positional Encoding Tri-Planes for Neural Surfaces. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12598–12607.

Zixiong Wang, Pengfei Wang, Peng-Shuai Wang, Qiujie Dong, Junjie Gao, Shuangmin Chen, Shiqing Xin, Changhe Tu, and Wenping Wang. 2024b. Neural-IMLS: Self-Supervised Implicit Moving Least-Squares Network for Surface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics* 30, 8 (2024), 5018–5033.

Xinyue Wei, Fanbo Xiang, Sai Bi, Anpei Chen, Kalyan Sunkavalli, Zexiang Xu, and Hao Su. 2023. NeuManifold: Neural Watertight Manifold Reconstruction with Efficient and High-Quality Rendering Support. In *arXiv preprint arXiv:2305.17134*.

Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. 2020. SynSin: End-to-end View Synthesis from a Single Image. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Francis Williams, Jiahui Huang, Jonathan Swartz, Gergely Klar, Vijay Thakkar, Matthew Cong, Xuanchi Ren, Ruilong Li, Clement Fuji-Tsang, Sanja Fidler, Eftychios Sifakis, and Ken Museth. 2024. fVDB: A Deep-Learning Framework for Sparse, Large Scale, and High Performance Spatial Intelligence. *ACM Transactions on Graphics* 43, 4, Article 133 (Nov. 2024), 15 pages.

Markus Worchel, Rodrigo Diaz, Weiwen Hu, Oliver Schreer, Ingo Feldmann, and Peter Eisert. 2022. Multi-View Mesh Reconstruction With Neural Deferred Shading. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6187–6197.

Tong Wu, Jiaqi Wang, Xingang Pan, Xudong Xu, Christian Theobalt, Ziwei Liu, and Dahua Lin. 2023. Voxurf: Voxel-based Efficient and Accurate Neural Surface Reconstruction. In *The International Conference on Learning Representations*.

Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. 2022. Point-NeRF: Point-Based Neural Radiance Fields. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5438–5448.

Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. 2024. Multi-Scale 3D Gaussian Splatting for Anti-Aliased Rendering. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20923–20931.

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. 2021. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems*.

Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. 2019. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics* 38, 6, Article 230 (Nov. 2019), 14 pages.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. 2024a. Mip-Splatting: Alias-free 3D Gaussian Splatting. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19447–19456.

Zehao Yu, Torsten Sattler, and Andreas Geiger. 2024b. Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes. *ACM Transactions on Graphics* (2024).

Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. 2024. RaDe-GS: Rasterizing Depth in Gaussian Splatting. In *arXiv preprint arXiv:2406.01467*.