SDFDiff: Differentiable Rendering of Signed Distance Fields for 3D Shape Optimization

Yue Jiang, Dantong Ji, Zhizhong Han, Matthias Zwicker
University of Maryland, College Park
{yuejiang, dji, h312h, zwicker}@cs.umd.edu

Abstract

We propose SDFDiff, a novel approach for image-based shape optimization using differentiable rendering of 3D shapes represented by signed distance functions (SDFs). Compared to other representations, SDFs have the advantage that they can represent shapes with arbitrary topology, and that they guarantee watertight surfaces. We apply our approach to the problem of multi-view 3D reconstruction, where we achieve high reconstruction quality and can capture complex topology of 3D objects. In addition, we employ a multi-resolution strategy to obtain a robust optimization algorithm. We further demonstrate that our SDF-based differentiable renderer can be integrated with deep learning models, which opens up options for learning approaches on 3D objects without 3D supervision. In particular, we apply our method to single-view 3D reconstruction and achieve state-of-the-art results.

1. Introduction

The "vision as inverse graphics" or "inverse rendering" strategy has long been attractive as a conceptual framework to solve inverse problems such as recovering shape or appearance models from images. In this analysis-by-synthesis approach, the goal is to reproduce given input images by synthesizing them using an image formation model, possibly including shape, appearance, illumination, and camera parameters. Solving this problem implies finding suitable model parameters (shape, appearance, illumination, camera) that describe the underlying scene. While conceptually simple, this approach can be challenging to use in practice, because it requires a suitable parameterization of a powerful image formation model, and effective numerical techniques to solve the resulting optimization problem. Recently, automatic differentiation has attracted renewed attention to implement differentiable renderers or image formation models that can be used in gradient-based optimization techniques. In particular, it is attractive to combine differentiable rendering with neural networks to solve highly ill-posed inverse problems, such as single-view 3D reconstruction.

In this paper, we advocate using signed distance fields (SDFs) in a differentiable image formation model because they have several advantages over other geometry representations. In contrast to triangle meshes, the surface topology is not fixed in SDFs and can adapt to the actual scene topology during optimization. Point clouds can also represent arbitrary topologies but they do not provide continuous surface reconstructions. Instead, SDFs inherently represent continuous, watertight surfaces, which are required for downstream applications such as 3D printing and physics-based simulation. In addition, SDFs can easily be used in a multi-resolution framework, which is important to avoid undesired local minima during optimization.

The main contribution of our paper is SDFDiff, a differentiable renderer based on ray-casting SDFs. Our renderer is integrated with a deep learning framework such that it can be combined with neural networks to learn how to solve highly ill-posed inverse problems. Finally, we provide an effective multi-resolution strategy to improve the robustness of gradient-based optimization. We demonstrate the usefulness of our approach using several application studies, including multi-view 3D reconstruction and learning-based single-view 3D reconstruction. Our results demonstrate the advantages of SDFs over other surface representations.

In summary, we make the following contributions:

- We introduce a differentiable renderer based on ray-casting SDFs, and we describe an implementation that is integrated into a standard deep learning framework. Advantages of using SDFs for differentiable rendering and shape optimization include that we can adapt the topology freely, and that the resulting shapes are guaranteed to consist of watertight surfaces.

- We present results of a multi-view 3D reconstruction approach using shape optimization via differentiable rendering. Using a multi-resolution approach, our gradient-descent optimization reliably converges to high quality solutions. Our approach is able to reconstruct geometry with high level of detail and complex topology, even with few input views.

- We leverage the SDF-based differentiable renderer to train deep neural networks to perform single-view 3D shape reconstruction without 3D supervision. We demonstrate the advantages of our approach by recovering accurate 3D shapes with arbitrary topology.

2. Related Work

Signed Distance Functions. A distance function is a level set representation [39, 42] that, at each point in 3D, stores the distance to the closest point on the 3D surface. Signed distance fields (SDFs) [8] store a signed distance to distinguish between the inside and outside of objects. SDFs are often discretized using uniform voxel grids [35, 38]. Compared to meshes or parametric surfaces, the implicit surface representation [32, 6] of SDFs has the advantage that it can represent arbitrary topologies. In contrast to point clouds, SDFs always represent watertight surfaces. SDFs recently started attracting interest for shape analysis via deep learning. DeepSDF [41] was proposed to learn continuous SDF representation of a class of shapes. Similarly, deep level sets [33] were introduced as an end-to-end trainable model that directly predicts implicit surfaces of arbitrary topology. However, these methods require 3D supervision during training, such as pairs of 3D coordinates and their corresponding SDF values [41], or voxel occupancy maps [33].

Learning-based 3D Reconstruction. Reconstructing 3D models from 2D images is a classic problem in computer vision. Compared to traditional multi-view stereo and shading-based approaches [46], learning-based 3D reconstruction can achieve impressive performance even with very few input views. Deep learning models for 3D shape understanding have been proposed for different kinds of 3D representations, including multiple views [13, 14], point clouds [52, 18], triangle meshes [27, 28], voxel grids [54, 50], and signed distance functions (SDFs) [41]. However, most learning-based methods [51, 49, 10, 1, 7, 19, 23] require 3D supervision. Although some methods [4, 16] do not require supervised learning, they are often limited by specific settings, such as restricted lighting conditions or annotation of object orientation. In contrast, predicting 3D shapes with differentiable renderers [25, 18, 19, 34, 29, 37] has recently attracted increasing attention as it enables 3D reconstruction without 3D supervision, that is, by optimizing neural networks only using images as training data.

Differentiable Rendering Voxel-based differentiable renderers [50, 11, 17, 20, 54, 12, 55, 53, 36] first drew attention performing volumetric ray marching, however, they are limited to low-resolution voxel grids. To render SDFs we use sphere tracing, also used in scene representation networks (SRNs) [48]. SRNs learn how to sphere trace novel views, but cannot produce full 3D shapes in a single step. They do not reconstruct a view independent shape representation and focus more on novel view synthesis rather than 3D watertight surface reconstruction, which is our goal. Starting with Loper and Black's OpenDR [30], much recent work focused on mesh-based differentiable renderers [16, 40]. Kato et al. [21] proposed a neural 3D mesh renderer with hand-designed gradients. Paparazzi [27] employed analytically computed gradients to adjust the location of vertices. Similarly, SoftRas [28] assigned each pixel to all faces of a mesh in a probabilistic rasterization framework. Although these methods enable to learn 3D mesh reconstruction without 3D supervision, they are restricted to a fixed, usually spherical mesh topology. Many of these mesh-based rendering approaches are differentiable with respect to geometry [4, 21, 30, 26, 9, 45, 28], lighting models [5], textures [26, 44, 2], or materials [44, 31, 2]. Mesh-based differentiable rendering [43, 25] has also been applied to real images, although current results are rather limited and applying differentiable rendering for real photographs remains an open challenge as it requires a comprehensive global illumination model.

Differentiable rendering has also been applied to Monte Carlo ray tracing [25] and point cloud rendering [19, 34]. Insafutdinov et al. [18] proposed a point cloud-based differentiable renderer with visibility modeling by conducting orthogonal projection on voxelized 3D space holding the point cloud. Surface splatting [52] was employed to model the visibility in point cloud rendering. Although point clouds can be easily acquired using range sensing technology, such as Microsoft Kinect and LIDAR, they do not explicitly represent topology and require post-processing to produce watertight surfaces. Concurrent works [29, 37] proposed differentiable rendering based on SDFs and on occupancy networks, further improving the quality of 3D reconstruction.

3. Overview

We propose a novel approach for image-based 3D shape optimization by leveraging SDFs as the geometric representation to perform differentiable rendering. Given a set of parameters Θ representing the geometry description, lighting model, camera position, etc, a renderer R can be written as a forward operator that produces an image I by computing $I = R(\Theta)$. In contrast, optimizing geometry and other scene parameters from images is a backward process. Given a desired target image I, our goal is to get the set of parameters $\Theta = R^{-1}(I)$ that produces the target image. The rendering process itself is not invertible. Hence, instead of solving the inverse rendering problem directly, we can formulate it as an energy minimization problem,

$$\Theta^* = \arg\min_{\Theta} \mathcal{L}_{\text{img}}(R(\Theta), I) \tag{1}$$

where $\mathcal{L}_{\text{img}}$ is a loss function measuring the distance between the target image and the rendered image from the 3D object. In practice, the loss is typically accumulated over multiple target images. Getting the desired parameters Θ^* is equivalent to minimizing the loss $\mathcal{L}$. While all rendering parameters including geometry, illumination, camera pose, and surface appearance could in theory be recovered from images this way, we focus on shape optimization in this paper and assume the other parameters are known. To enable gradient-based optimization, a key issue is to obtain the gradient of $\mathcal{L}_{\text{img}}(R(\Theta), I)$ with respect to the parameters Θ. A differentiable renderer achieves this by producing not only images from a description of the scene, but also the derivatives of pixel values with respect to scene parameters.

In this paper, we propose a novel differentiable renderer which uses signed distance functions (SDFs) and camera pose as inputs and renders an image. Our SDF-based differentiable renderer leverages the ray casting algorithm and uses automatic differentiation to compute the derivatives.

4. Differentiable SDF Rendering

We represent discrete SDFs by sampling SDF values on regular grids, and apply a standard ray casting algorithm based on sphere tracing [15] to find the intersection points between rays and the object surface. For this purpose we employ trilinear interpolation to reconstruct continuous SDFs that can be evaluated at any desired location. This allows us to continuously represent the object surface, which is given by the zero level set of the interpolated SDF.

A key observation is that the derivatives of a given pixel with respect to rendering parameters only depend on a local neighborhood of eight SDF samples that define the value of the trilinearly interpolated SDF at the surface intersection point. In other words, the sphere tracing process itself does not need to be differentiable. Instead, only the local computations involving the local set of eight SDF samples around the surface intersection need to be differentiable. Therefore, our approach proceeds in two stages: first, we apply sphere tracing to identify the eight samples nearest to the surface intersection. This step is not differentiable. Second, we locally compute the pixel color based on the local set of SDF samples. This step is implemented using an automatic differentiation framework to obtain the derivatives.

While differentiable ray marching of voxel grids has been used before [54, 55, 36], these approaches are based on voxel opacities, given by binary variables or continuous occupancy probabilities. In these cases ray marching through the entire volume needs to be differentiated because all voxels along a ray may influence the corresponding pixel.

Sphere Tracing. We perform ray casting via sphere tracing [15] in the first stage by starting from the ray origin, and evaluating the SDF using trilinear interpolation to find the minimum distance from the current point on the ray to the object. Then we move along the ray by that distance. Moving along the ray by the minimum distance to the object guarantees that we will never move across the boundary of the object, while allowing us to make a possibly large step towards the surface. We repeat this process until we reach the surface of the object, that is, until the SDF value at our current position on the ray is small enough, or until we leave the bounding box of the object. While the efficiency of sphere tracing can be improved by increasing the step size [29], we implemented sphere tracing directly in CUDA without support for automatic differentiation. Hence, the computation cost of this step is negligible in our approach.

Differentiable Shading. In the second stage, we compute the pixel color as a function of the local SDF samples that define the SDF at the intersection point, as determined by the first stage. These computations are implemented in a framework that supports automatic differentiation, allowing us to easily obtain the derivatives of the pixel. For each pixel, the input consists of the light and camera parameters, and the eight SDF samples closest to the ray-surface intersection point. The computations include: getting the intersection point and the surface normal at the intersection point as a function of the trilinear basis coefficients (i.e., the eight SDF samples), and evaluating a shading model.

To take into account the dependence of the pixel value on the ray-surface intersection point, we express the intersection point as a function of the eight local SDF samples. Let us denote the local SDF values by $d_0, \ldots, d_7$, the current position on the ray (obtained from the ray casting stage) by $s \in \mathbb{R}^3$, and the unit ray direction by $v \in \mathbb{R}^3$. To express the intersection point as a function of $d_0, \ldots, d_7$, we use the same approximation as in the ray casting stage, that is, the approximate intersection is $p(d_0, \ldots, d_7) = s + \text{trilinear}(d_0, \ldots, d_7; s)v$, where $\text{trilinear}(d_0, \ldots, d_7; s)$ is the trilinear interpolation of the SDF at location s and considered as a function of $d_0, \ldots, d_7$. This approximation is conservative in the sense that it is accurate only if the SDF represents a plane that is perpendicular to the ray direction v. Otherwise, $p(d_0, \ldots, d_7)$ is guaranteed not to cross the true intersection along the ray.

As an alternative to our conservative approximation, one could express the intersection point exactly as the solution of the intersection of the ray $s + tv$ and the local trilinear interpolation of the SDF. That is, we could express the solution of $\text{trilinear}(d_0, \ldots, d_7; s + tv) = 0$ with respect to $t \in \mathbb{R}$ as a function of $d_0, \ldots d_7$. However, this involves finding roots of a cubic polynomial, and we found that our much simpler approach works more robustly in practice.

To evaluate a shading model, we need the surface normal at the intersection point $p(d_0, \ldots, d_7)$. Considering that the surface normal corresponds to the gradient of the SDF, we first compute gradients at the grid vertices using central finite differencing, and then trilinearly interpolate them at the intersection point $p(d_0, \ldots, d_7; s)$. In summary, this leads to an expression of the normal at the intersection point as a function of SDF coefficients within an $4 \times 4 \times 4$ neighborhood around the intersection (because of central finite differencing). Surface normals are normalized after trilinear interpolation. Finally, in our current implementation we evaluate a simple diffuse shading model.

Implementation. We implemented SDF ray casting using CUDA to leverage the computational power of GPUs. Differentiable shading is implemented with the Pytorch library, which supports automatic differentiation and allows seamless integration of the renderer with neural network training. Pytorch also leverages the GPU, and our implementation directly accesses the output of the ray casting stage that is stored in GPU memory, avoiding any unnecessary memory transfers. Our code is available at https://github.com/YueJiang-nj/CVPR2020-SDFDiff.

5. Multi-view 3D Reconstruction

In this section we describe how to perform multi-view 3D reconstruction using our differentiable renderer. This is a proof of concept, where we assume known camera poses, illumination, and surface appearance, and we only optimize over the 3D shape represented by the SDF. Our inputs are synthetically rendered images from a fixed set of camera poses. We set the camera poses to point from the center of each face and edge, and from each vertex of the object bounding box towards its center, where the bounding box is a cube. Since the cube has 6 faces, 8 vertices, and 12 edges, we obtain 26 camera poses in total. Figure 1 shows the input images we used to reconstruct the bunny in our experiments. In addition, we initialize the SDF to a sphere.

5.1. Energy Function

For simplicity we choose the L_2 distance between the rendered and the target images as our image-based loss, that is $\mathcal{L}_{\text{img}}(R(\Theta), I) = ||R(\Theta) - I||^2$. The loss is summed over all target views. In this proof of concept scenario, the optimization parameters Θ include only the SDF values, as we assume the other rendering parameters are known. Minimizing the image-based loss by optimizing SDF values requires differentiable rendering, where we compute the gradient of the image loss w.r.t. the SDF values as in Section 4.

In addition, we impose a regularization loss that ensures that the SDF values Θ represent a valid signed distance function, that is, its gradient should have unit magnitude.

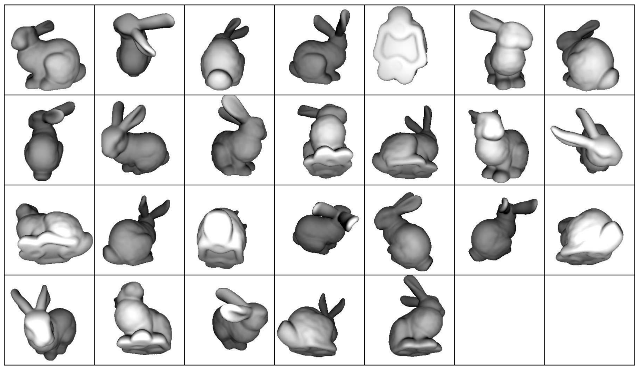
Figure 1. We use 26 input views in our multi-view reconstruction experiments as shown here for the bunny.

Writing the SDF represented by Θ as a function $f(x; \Theta)$, where x is a point in 3D, the regularization loss is

$$\mathcal{L}_{\text{reg}} = \int ||1 - ||\nabla f(x; \Theta)||^2||^2 dx \qquad (2)$$

In practice, we obtain the gradients via finite differencing and we compute a discrete sum over the SDF grid vertices.

5.2. Iterative Optimization

We apply gradient descent optimization using ADAM [22] to iteratively optimize our SDF to match the target images. Compared to straightforward gradient descent, ADAM is more robust and faster in convergence. In addition, we accelerate convergence by greedily selecting a single view in each gradient descent step to compute the gradient, similar to active mini batch sampling. The intuition is that some parts of the 3D model may have more complex structures so it is more difficult to optimize SDF values using some views than others. Different views may incur image losses of varying magnitude, and we should focus on the views with large losses to make sure all parts of the object can be well-reconstructed. Our approach first calculates the average loss for all the camera views from the result of the previous iteration. If a loss for a view is greater than the average loss, then during the current iteration, we update the SDF until the loss for this view is less than the average (with max. 20 updates). For the other views, we update the SDF five times. If one update increases the loss, then we switch to the next view directly. We stop our optimization process when the loss is smaller than a given tolerance or the step length is too small.

Reconstructing high-resolution 3D objects is challenging because gradient descent takes many iterations to eliminate low frequency errors. Therefore, we apply a coarse-to-fine multi-resolution approach. We start by initializing the SDF grid at a resolution of 8^3 to the SDF of a sphere. We then iterate between performing gradient descent optimization as described above, and increasing the grid resolution. We in-

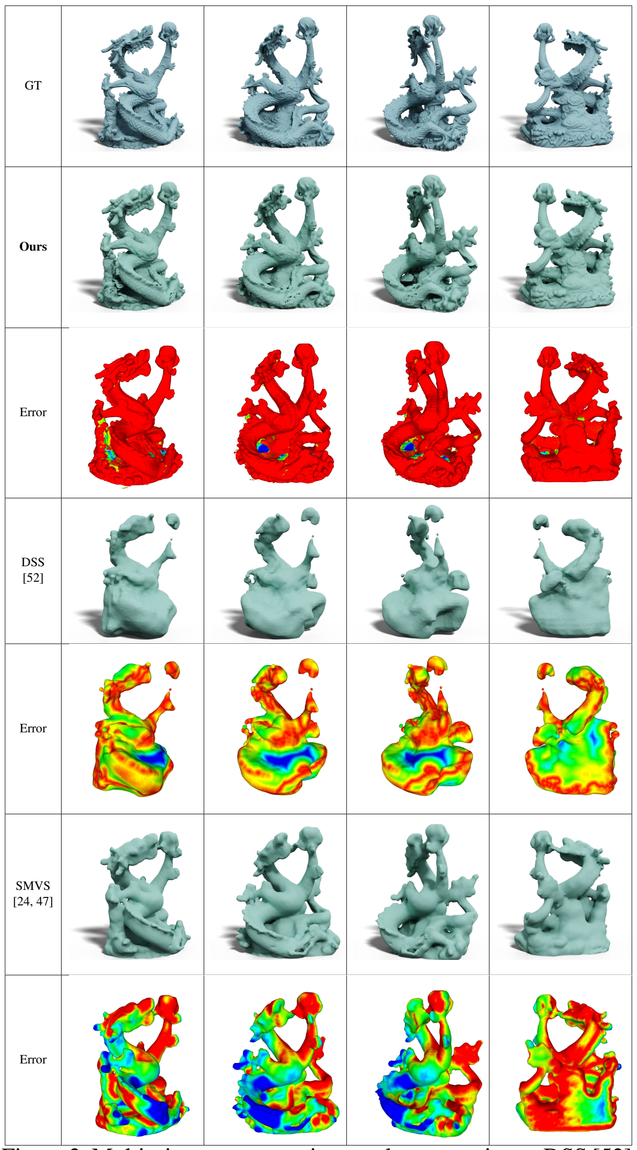

Figure 2. Multi-view reconstruction results comparing to DSS [52] and SMVS [24, 47] with the corresponding error visualizations based on Hausdorff distance (red means zero and blue high error).

crease the resolution simply by performing trilinear interpolation and stop at a resolution of 64^3.

To further improve the efficiency of the multiresolution scheme, we choose an appropriate image resolution for rendering corresponding to the SDF resolution at each resolution level. We determine the appropriate resolution by ensuring that a sphere with a radius equivalent to the grid spacing, and placed at the corner of the bounding box of the SDF furthest from the camera, has a projected footprint of at most the size of a 2×2 pixel block.

5.3. Experimental Results

Qualitative Results. We compare our results with DSS [52], which is a differentiable renderer for point clouds based on surface splatting [56]. We let both systems deform a sphere to fit the target object given as input. When running DSS, we adopt the same settings used in their original experiments: the system randomly selects 12 from a set of 26 views of the target in each optimization cycle, and optimizes for up to 16 cycles. We experimented with different numbers of 3D points and report the best result. For SDFDiff we use our optimization technique from Section 5.2 using the same set of 26 views. Figure 2 shows the comparison between SDFDiff and DSS. DSS cannot recover geometric details as accurately as SDFDiff.

Object	**Ours**	DSS [52]	SMVS [24, 47]
Torus	**0.015637**	0.035398	N/A
Bunny	**0.026654**	0.109432	N/A
Dragon	**0.074723**	0.179456	0.097816

Table 1. Comparison of the symmetric Hausdorff distance between ground truth and reconstructed meshes for torus, bunny and dragon. SMVS could not reconstruct torus and bunny because camera pose estimation failed.

We also compare our result with SMVS [24, 47], which is a state-of-the-art shading-aware multi-view 3D reconstruction approach. We use the default settings, and provide 1000 randomly sampled views of the dragon rendered by our SDF renderer as input. Note that SMVS automatically recovers camera parameters from the input images and estimates surface albedo and illumination, hence the comparison is not entirely fair. As shown in Figure 2, however, even with a large number of input views the SMVS output can be overly smooth and lack details. SMVS may also fail with fewer views due to inaccurate camera pose estimation. In contrast, SDFDiff can obtain better results using only 26 views (with known camera poses, albedo, and illumination).

Quantitative Results. Table 1 compares the symmetric Hausdorff distance between ground truth and reconstructed meshes for torus, bunny and dragon. The visual results of torus and bunny are in supplementary materials. For a fair comparison, we report errors relative to the size of the bounding boxes. We observe that SDFDiff leads to smaller symmetric Hausdorff distances, which means our reconstruction results are closer to the ground truth than the other two approaches.

5.4. Parameter Study

Initial Resolution. Figure 3 shows the impact of the initial resolution in our multi-resolution scheme. We fix the number of multi-resolution steps and our target resolution being 64, and then set the initial resolution to be 8, 16, 32, and 48 respectively. We find that a lower initial resolution can reconstruct qualitatively better 3D shapes because it more robustly captures large scale structures.

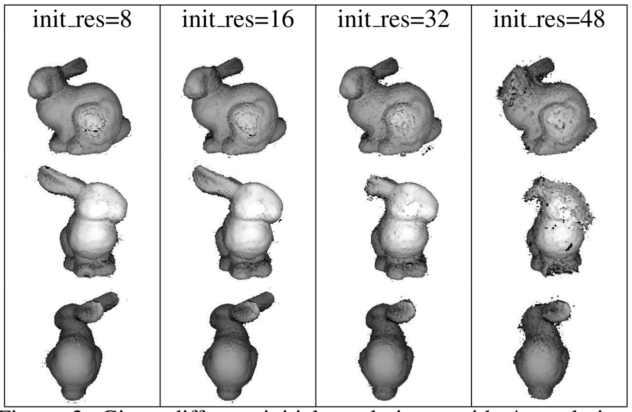

Figure 3. Given different initial resolutions, with 4 resolution stages, we can find that our 3D reconstruction results are better if the initial resolution is lower.

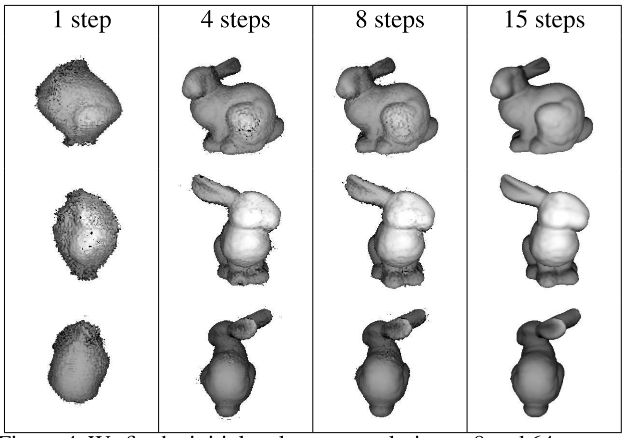

Figure 4. We fix the initial and target resolution to 8 and 64 respectively, but use different numbers of intermediate resolution stages. We find that more resolution stages can give us better results.

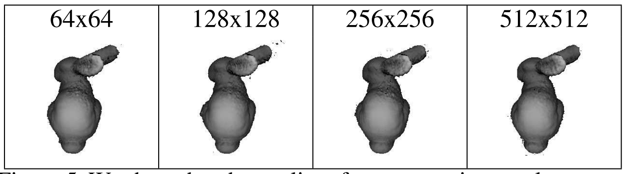

Figure 5. We show that the quality of reconstruction results are not affected much by the image resolution.

Number of Multi-Resolution Steps. Figure 4 shows that given fixed initial (init_res=8) and target (target_res=64) resolutions, adding more multi-resolution steps can give us better results. In particular, single-resolution optimization (1 step) cannot reconstruct the object successfully, further justifying our multi-resolution setup.

Image Resolution. Figure 5 shows that image resolution does not significantly affect the quality of the results, where we use images with various resolutions for optimization.

Noisy Data. As shown in Figure 6, when some noise is added to target images or camera poses, our approach can still maintain its robustness.

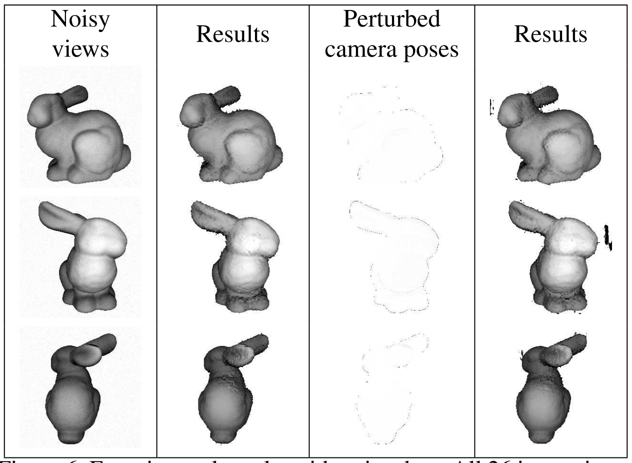

Figure 6. Experimental results with noisy data. All 26 input views or camera poses are perturbed with Gaussian noise (variance=0.03 for views and variance=0.01 for camera poses). The third column shows the view differences caused by perturbed camera poses.

6. Learning-based Single-view Reconstruction

In the following experiments, we leverage SDFDiff to train a neural network to perform single-view 3D reconstruction without 3D supervision. We use the same dataset as [21, 28], which includes 13 categories of objects from ShapeNet [3]. Each object has 24 rendered images from different views at 64×64 resolution. We use the same train/validate/test sets on the same dataset as in [21, 28, 54], and the standard reconstruction metric, *i.e.,* 3D intersection over union (IoU) [28] for quantitative comparisons.

Network. Our network contains two parts as shown in Figure 7. The first part is an Encoder-Decoder network which takes images as input and outputs coarse SDF results. The second part is a refiner network to further improve the quality of the 3D reconstruction results. The network is trained on all the 3D shapes in the dataset simultaneously.

Loss Function. In addition to the energy function as shown in Section 5.1 containing the L_2 image-based loss $\mathcal{L}_{\text{img}}$ and the SDF loss $\mathcal{L}_{\text{reg}}$ ensuring the SDF values represent a valid signed distance function, we also add a geometry loss $\mathcal{L}_{\text{geo}}$ that regularizes the finite difference Laplacian of the predicted SDFs to obtain smooth outputs. Furthermore, we use a narrow band technique to control the effects of the SDF and Laplacian losses since we care more about these losses locally around the surfaces. Also, the SDF-loss cannot be enforced everywhere on the discrete grid due to singularities (e.g., the medial axis of the shape forms a sharp crease) in the continuous SDF. The narrow-band considers the SDF-loss only close to the surface, avoiding SDF discretization issues elsewhere in the volume. We use a distance-based binary mask $\mathcal{M}$ to zero them out further away from the zero level-set. The mask is defined as

$$\mathcal{M} = ||SDF|| \leq \mu \times voxelSize, \tag{3}$$

Category	Airplane	Bench	Cabinet	Car	Chair	Display	Lamp	Speaker	Rifle	Sofa	Table	Phone	Vessel	Mean
NMR [21]	0.6172	0.4998	0.7143	0.7095	0.4990	0.5831	0.4126	0.6536	0.6322	0.6735	0.4829	0.7777	0.5645	0.6015
SoftRas (sil.) [28]	0.6419	0.5080	0.7116	0.7697	0.5270	0.6156	0.4628	0.6654	**0.6811**	0.6878	0.4487	0.7895	0.5953	0.6234
SoftRas (full) [28]	0.6670	0.5429	0.7382	0.7876	0.5470	0.6298	0.4580	0.6807	0.6702	0.7220	0.5325	0.8127	0.6145	0.6464
Ours	**0.6874**	**0.6860**	**0.7735**	**0.8002**	**0.6436**	**0.6584**	**0.5150**	0.6534	0.5553	**0.7654**	**0.6288**	**0.8278**	**0.6244**	**0.6674**

Table 2. Comparison of IoU with the state-of-the-art approaches [21, 28] on 13 ShapeNet categories.

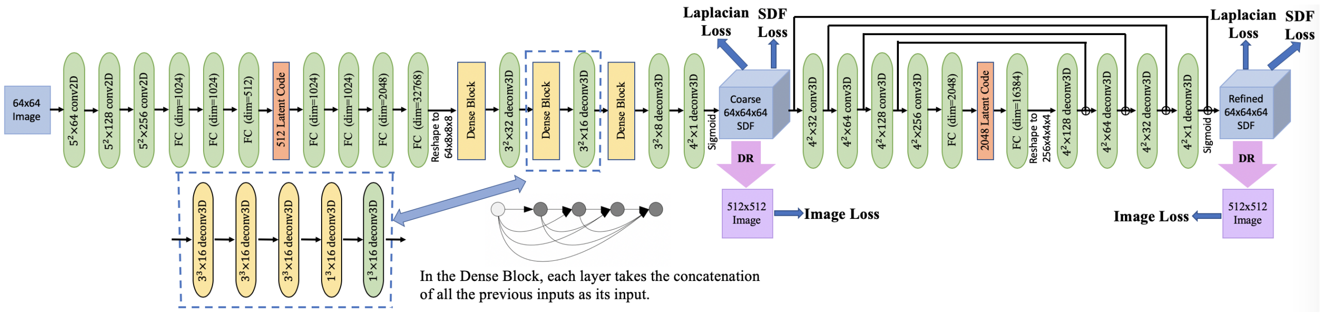

Figure 7. Network structure for single-view SDF reconstruction.

where μ is a hyperparameter to define the width of the narrow band. We currently set it to be 1.6, which is determined experimentally. The final loss function is a weighted sum of the three losses with weights $\lambda_1 = \lambda_2 = 0.02$,

$$\mathcal{L} = \mathcal{L}_{\text{img}} + \mathcal{M} \otimes (\lambda_1 \mathcal{L}_{\text{reg}} + \lambda_2 \mathcal{L}_{\text{geo}}). \tag{4}$$

Training Process. We first train the Encoder-Decoder part of the network alone based on the three loss terms. Then we fix the encoder and decoder and train the refiner network on the same three loss terms to get refined SDF shapes. In the end, we train all the three parts, *i.e., encoder, decoder, and refiner* together to further improve the results. We do not use the multi-resolution approach.

Qualitative Evaluation. Figure 8 shows that our method can reconstruct detailed objects and accurately recover complicated topologies. In contrast, SoftRasterizer [28] relies on a template mesh with spherical topology and it cannot capture the complex topology of the chairs.

Quantitative Evaluation. We compare our method with the state-of-the-art [21, 28] in terms of 3D IoU scores in Table 2. Our method can reconstruct shapes with finer details in the 13 categories. In addition, the IoU numbers show that our results achieve higher accuracy, where our scores surpass other approaches in most of the categories. A comparison to Chen et al. [5] is omitted because they use different data preprocessing than the other methods [21, 28].

7. Discussion and Limitations

As a main advantage, SDFs can represent arbitrary topologies, in contrast to triangle meshes that are restricted to the topology of a template. In contrast to point clouds, SDFs inherently represent continuous watertight surfaces. We demonstrated applications of our approach in multi-view shape reconstruction, and single view 3D reconstruction using deep learning. Our experimental results showed that we can more robustly perform multi-view reconstruction than a state-of-the-art point-based differentiable renderer. In addition, we achieve state-of-the-art results on single view 3D reconstruction with deep learning models.

In our multi-view 3D reconstruction approach, our current shading model is not sufficient to perform inverse rendering from real images taken with a camera. For example, we currently do not include effects such as shadows, interreflections, texture, non-diffuse surfaces, or complex illumination. In contrast to rasterization-based differentiable renderers, our ray tracing-based renderer could be extended to include all such effects. A disadvantage of our deep learning approach is that we output a discrete SDF on a 3D grid. Instead, we could learn a continuous signed distance function represented by a deep network like in DeepSDF [41]. This would be more memory efficient, but it might be computationally too expensive for unsupervised 3D reconstruction with differentiable rendering, since it would require to evaluate the network for each ray marching step.

8. Conclusion

We proposed a novel approach to differentiable rendering using signed distance functions to represent watertight 3D geometry. Our rendering algorithm is based on sphere tracing, but we observe that only the local shading computation needs to be differentiable in our framework, which makes the approach computationally more efficient and allows for straightforward integration into deep learning frameworks. We demonstrate applications in multi-view 3D reconstruction and unsupervised single-view 3D reconstruction using deep neural networks. Our experimental results illustrate the advantages over geometry representations such as point clouds and meshes. In particular, we report the state-of-the-art results in shape reconstruction.

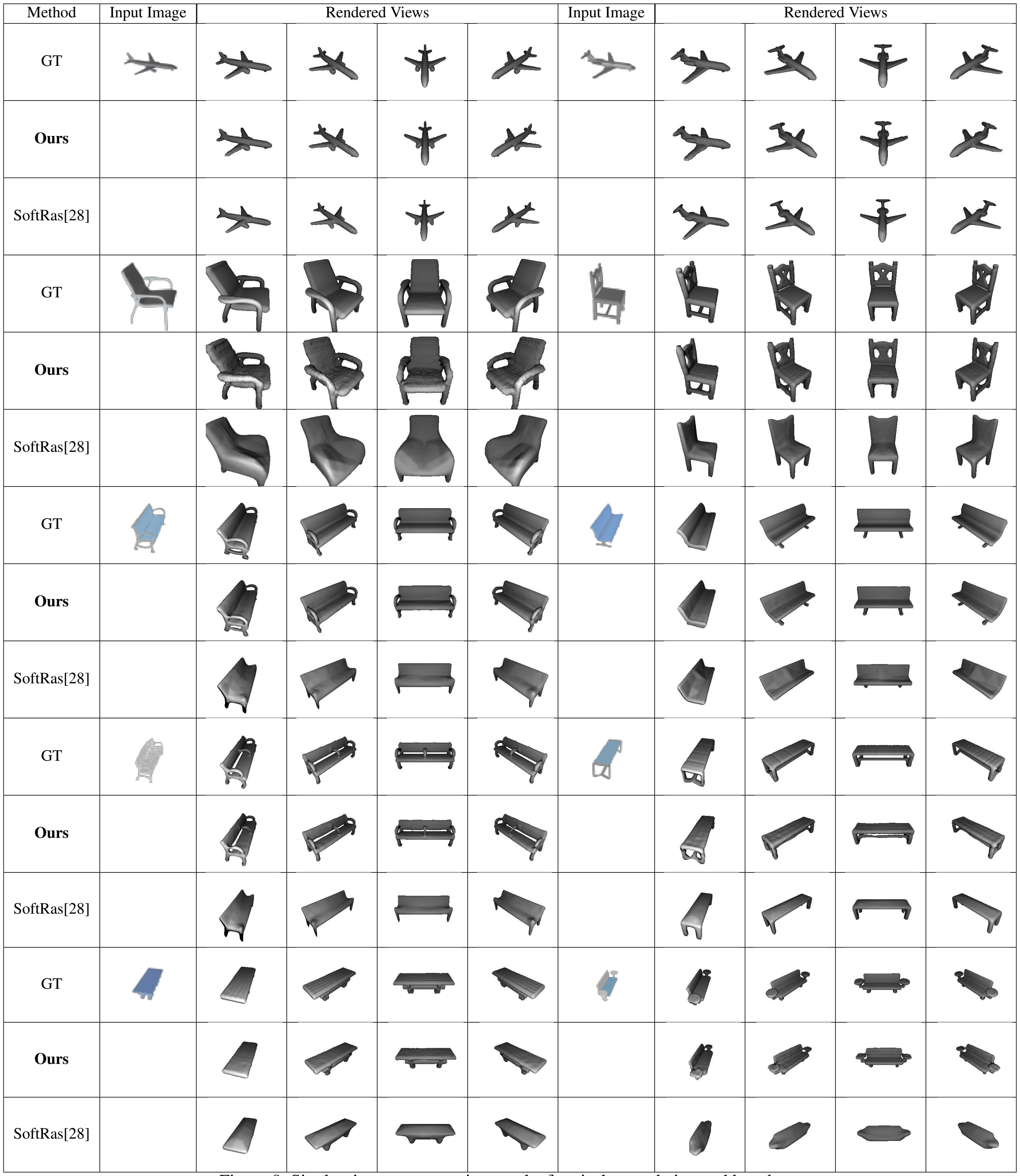

Figure 8. Single-view reconstruction results for airplanes, chairs, and benches.

9. Acknowledgements

This project was supported by NSF IIS grant nr. #1813583. We also appreciate active discussion with Qian Zheng, Hui Huang, and Daniel Cohen-Or at Shenzhen University.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Representation learning and adversarial generation of 3D point clouds. *CoRR*, abs/1707.02392, 2017.

[2] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning*, pages 284–293, 2018.

[3] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository. *ArXiv*, abs/1512.03012, 2015.

[4] Chengqian Che, Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. Inverse Transport Networks. *arXiv e-prints*, page arXiv:1809.10820, Sept. 2018.

[5] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*.

[6] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[7] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[8] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, 1996.

[9] Amaël Delaunoy and Emmanuel Prados. Gradient flows for optimizing triangular mesh-based surfaces: Applications to 3D reconstruction problems dealing with visibility. *International Journal of Computer Vision*, 95(2):100–123, Nov 2011.

[10] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3D object reconstruction from a single image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[11] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3D shape induction from 2D views of multiple objects. *2017 International Conference on 3D Vision (3DV)*, pages 402–411, 2016.

[12] J. Gwak, C. B. Choy, M. Chandraker, A. Garg, and S. Savarese. Weakly supervised 3D reconstruction with adversarial constraint. In *2017 International Conference on 3D Vision (3DV)*, pages 263–272, Oct 2017.

[13] Zhizhong Han, Mingyang Shang, Yu-Shen Liu, and Matthias Zwicker. View Inter-Prediction GAN: Unsupervised representation learning for 3D shapes by learning global shape memories to support local view predictions. In *AAAI*, pages 8376–8384, 2019.

[14] Zhizhong Han, Mingyang Shang, Xiyang Wang, Yu-Shen Liu, and Matthias Zwicker. Y2Seq2Seq: Cross-modal representation learning for 3D shape and text by joint reconstruction and prediction of view and word sequences. In *AAAI*, pages 126–133, 2019.

[15] John C. Hart. Sphere Tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, Dec 1996.

[16] Paul Henderson and Vittorio Ferrari. Learning to generate and reconstruct 3D meshes with only 2D supervision. In *Proceedings of the 29th British Machine Vision Conference (BMVC 2018)*, 2018.

[17] P Henzler, N Mitra, and T Ritschel. Escaping Plato's Cave using adversarial training: 3d shape from unstructured 2d image collections. In *Proceedings of the International Conference on Computer Vision 2019 (ICCV 2019)*, volume 2019. IEEE, 2019.

[18] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *Advances in Neural Information Processing Systems (NeurIPS 2018)*, pages 2807–2817, 2018.

[19] Li Jiang, Shaoshuai Shi, Xiaojuan Qi, and Jiaya Jia. GAL: Geometric adversarial loss for single-view 3D-object reconstruction. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[20] Danilo Jimenez Rezende, S. M. Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3D structure from images. In *Advances in Neural Information Processing Systems 29 (NeurIPS 2016)*.

[21] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[23] A. Kurenkov, J. Ji, A. Garg, V. Mehta, J. Gwak, C. Choy, and S. Savarese. DeformNet: Free-form deformation network for 3D shape reconstruction from a single image. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 858–866, March 2018.

[24] F. Langguth, K. Sunkavalli, S. Hadap, and M. Goesele. Shading-aware multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[25] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018.

[26] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Adversarial geometry and lighting using a differentiable renderer. *CoRR*, abs/1808.02651, 2018.

[27] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. Paparazzi: Surface editing by way of multi-view image processing. *ACM Transactions on Graphics*, 2018.

[28] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft Rasterizer: A differentiable renderer for image-based 3D reasoning. *The IEEE International Conference on Computer Vision*, 2019.

[29] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. DIST: Rendering deep implicit signed distance function with differentiable sphere tracing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[30] Matthew M. Loper and Michael J. Black. OpenDR: An approximate differentiable renderer. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014.

[31] Abhimitra Meka, Maxim Maximov, Michael Zollhoefer, Avishek Chatterjee, Hans-Peter Seidel, Christian Richardt, and Christian Theobalt. LIME: Live intrinsic material estimation. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[32] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[33] Mateusz Michalkiewicz, Jhony K. Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders P. Eriksson. Deep Level Sets: Implicit surface representations for 3D shape inference. *CoRR*, abs/1901.06802, 2019.

[34] KL Navaneet, Priyanka Mandikal, Mayank Agarwal, and R Venkatesh Babu. CAPNet: Continuous approximation projection for 3D point cloud reconstruction using 2d supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8819–8826, 2019.

[35] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2011*, 2011.

[36] Thu Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yong-Liang Yang. RenderNet: A deep convolutional network for differentiable rendering from 3D shapes. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, 2018.

[37] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[38] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics*, 2013.

[39] S Osher, R Fedkiw, and K Piechor. Level Set Methods and Dynamic Implicit Surfaces. *Applied Mechanics Reviews*, 2004.

[40] Andrea Palazzi, Luca Bergamini, Simone Calderara, and Rita Cucchiara. End-to-end 6-DoF object pose estimation through differentiable rasterization. In *The European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[41] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *arXiv e-prints*, page arXiv:1901.05103, Jan 2019.

[42] Danping Peng, Barry Merriman, Stanley Osher, Hongkai Zhao, and Myungjoo Kang. A PDE-Based Fast Local Level Set Method. *Journal of Computational Physics*, 1999.

[43] Felix Petersen, Amit H. Bermano, Oliver Deussen, and Daniel Cohen-Or. Pix2Vex: Image-to-geometry reconstruction using a smooth differentiable renderer. *CoRR*, abs/1903.11149, 2019.

[44] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 117–128, New York, NY, USA, 2001. ACM.

[45] Elad Richardson, Matan Sela, Roy Or-El, and Ron Kimmel. Learning detailed face reconstruction from a single image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5553–5562, 07 2017.

[46] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 519–528, June 2006.

[47] Ben Semerjian. A new variational framework for multiview surface reconstruction. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014.

[48] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene Representation Networks: Continuous 3D-structure-aware neural scene representations. *CoRR*, abs/1906.01618, 2019.

[49] S. Tulsiani, A. Kar, J. Carreira, and J. Malik. Learning category-specific deformable 3D models for object reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):719–731, April 2017.

[50] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 209–217, 2017.

[51] Qingxiang Wang, Cezary Kaliszyk, and Josef Urban. First experiments with neural translation of informal to formal mathematics. *CoRR*, abs/1805.06502, 2018.

[52] WangYifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)*, 38(6), 2019.

[53] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, William T Freeman, and Joshua B Tenenbaum. MarrNet: 3D Shape Reconstruction via 2.5D Sketches. In *Advances In Neural Information Processing Systems (NeurIPS 2017)*, 2017.

[54] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective Transformer Nets: Learning single-view 3D object reconstruction without 3D supervision. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29 (NeurIPS 2016)*, pages 1696–1704. Curran Associates, Inc., 2016.

[55] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Joshua B. Tenenbaum, and William T. Freeman. Visual Object Networks: Image generation with disentangled 3D representations. In *Advances in Neural Information Processing Systems (NeurIPS 2018)*, 2018.

[56] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 371–378, New York, NY, USA, 2001. ACM.