Notes On Recent SOTA Inverse Rendering Works

Zichen Wang

inspired by discussions with
Gemmechu Hassena

supervised by
Xi Deng, Steve Marschner

Spring 2023

1 Introduction

In recent years, gradient-based optimization methods have led to huge improvements in the long-established inverse rendering problem. Current state-of-the-art models perform fairly well in reconstructing a single object and can render images indistinguishable from the target images by human eyes. As the field is developing rapidly, this manuscript serves to summarize recent state-of-the-art models, discuss the relation and improvements between models, and supply a general framework to lay the ground for comparison. This manuscript serves best for people who have knowledge of these models and want to gain more insights into them, as well as people who first get into this field and want to have a list of references. In Section 2, we frame the inverse problem as a process from 2D images to a 3D scene. We discuss the components necessary to accomplish the task, as well as different approaches to the inverse rendering problem. In Section 3, we give brief reviews of recent models, state which approach they take, and what components the models leverage.

2 Image to Scene

In one sentence, the ultimate goal of inverse rendering is that given a set of images, we want to reconstruct the scene model. In figure 1, this can be seen as the process to go from left (2D images) to right (3D scene). Naturally, we can start from the left and take conventional CV methods to extract some useful information from the images, or we can start from the right and decompose the scene into various components, such as camera, lighting, geometry, material, etc.

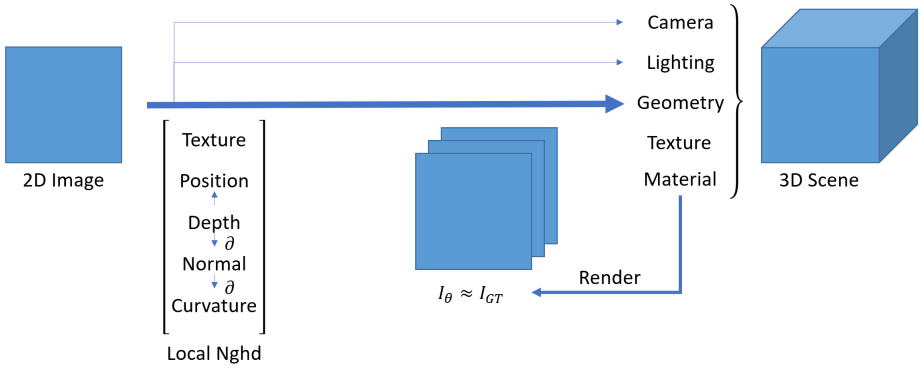

Figure 1: The Image2Scene framework. Our ultimate goal is to go from 2D images on the left to the 3D scene on the right. To achieve this, we can either start from the left and take conventional CV approaches, or we can start from the right and think about 3D representation and rendering methods.

2.1 CV: Left to Right

Apart from the direct RGB values stored in the images, there is much more information hidden in the images. Extracting this indirect information belongs to the field of *Computer Vision*.

Depth Map The *depth map* of an image tells the distance of the object to the camera. It provides important primitive information about the spatial position of a pixel. For example, *forward mapping* combines the depth maps of all input images to project the pixels onto their corresponding spatial positions.

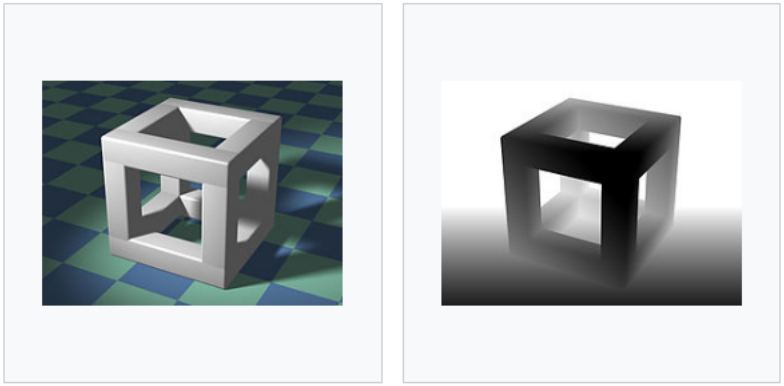

Figure 2: On the left is the image, and on the right is its depth map. Image from Wikipedia.

Normal Map If we take the gradient of the depth map, we will obtain the *normal map*, which tells us the normal of the scene objects.

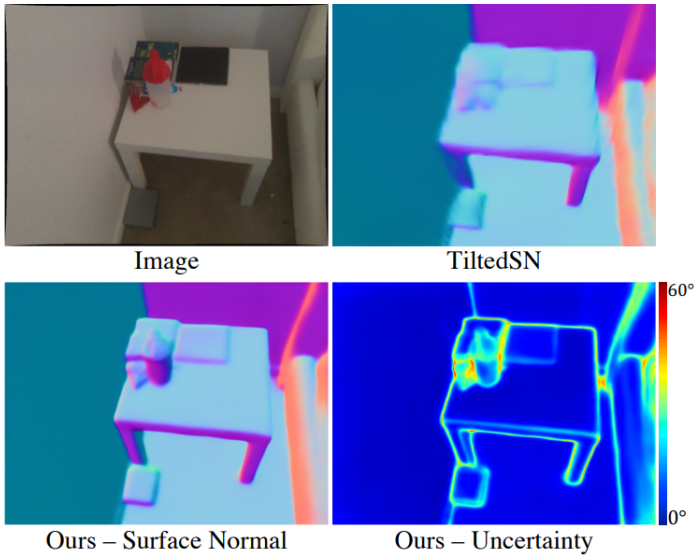

Figure 3: Normal maps. Figure from [1]

Curvature In theory, if we take the total derivative of the surface normals, we would obtain the curvature of that point on the surface. This then provides us with even more information on the scene objects.

2.2 CG: Right to Left

In the end, we want to reconstruct the scene model that is faithful to its real prototype. Answering what model should we use to represent reality and how well our model captures reality belongs to the field of *Computer Graphics*.

3D representation To bring reality into the computer, we might want to know the camera parameters for each input image, the light condition under which we take these images, the geometry of the objects in the scene, and the material/texture at each point on the object. Deciding how to model and store these components belongs to the question of *3D representation*. Although at first look one might overlook its importance, oftentimes the reconstruction method we take comes along how we envision and understand these components.

Conventional 3D representations include point clouds, mesh, and voxels, while nowadays implicit surfaces are gaining much attention in the inverse rendering problem. In 2019, Mescheder et al. proposed the first implicit representation *occupancy network* [6]. They leverage the depth map to train a neural network that discriminates positions inside the surface to positions outside the surface. Its successor, *signed distance field (SDF)*, assigns the position with the signed distance to the closest surface [9].

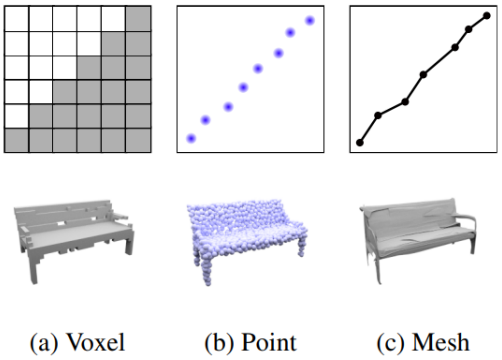

Figure 4: Conventional 3D representations. Figure from [6]

Rendering After some initial attempts, we might as well ask ourselves how we know that our reconstruction is faithful to reality. A natural answer is to *render* the scene and compare the rendered images with the ground truth input images. Now ideally we would of course want a photo-realistic renderer that also supports differentiation (so that we can leverage gradient-based methods to optimize the scene). Many works around *differentiable rendering* are devoted to this direction [4][5][12]. On the other hand, rendering methods specific to the 3D representation and to the scene are also popular in current literature, such as *volume rendering* [7][8] and *neural rendering* [10].

Inductive Bias Another approach to take a step from right to left is to assume *inductive bias* in the scene model. For example, we might know beforehand that the objects we want to reconstruct are watertight or have no sharp edges. Another common usage of inductive bias is to train the model on a huge amount of labeled 3D models. This would be helpful when the input images are missing some information to fully reconstruct the scene object, but our prior knowledge would make a reasonable guess on the missing part. To some extent, every model has to assume some inductive bias. Incorporating rendering in our reconstruction process, for example, is using our knowledge of light transport to explain shading and reflections in the images.

3 Important Works In Timeline

In this section, we seek to summarize and gain insights into recent state-of-the-art works. We start from relatively older work DeepView (2019) and Atlas (). We then move on to the first NeRF (2020) style works, as well as the SDF approach works like IDR (2020). The combination of NeRF and SDF gives volSDF (2021) and NeuS (2021). Finally, we come to the recent IRON (2022).

3.1 α-compositing

α-compositing refers to the process of combining images to mimic transparent effects. In terms of view synthesis, the α-compositing approach leverages the depth map to segment an input image into images of

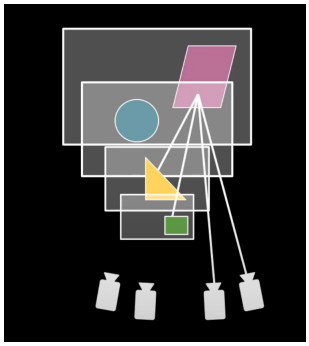

Figure 5: α-compositing. Image from [2]

different depths and then uses α-compositing to generate images of novel views. One of the last representative work in this line is DeepView [2].

If we try to fit in DeepView into the Image2Scene framework, we can see that it mainly leverages **depth** information. The main problem with DeepView, as well as many view synthesis approaches, is the **lack of a 3D representation**. Objects in DeepView are modeled as images with depths, which sacrifices many details a 3D object would have. This results in the camera can shift positions within a small range, but moving outside the range would lead to stretching and other geometry inconsistencies.

3.2 AtlasNet

A drastically different approach is to prepare a set of prototypes beforehand and adjust one of them to fit the images. Representative work in this line of approach would be the AtlasNet [3]. Such an approach mostly uses **inductive bias** by assuming all instances of a concept (for example, a plane), can be deformed from a prototype. To some extent, the nowadays prevalent generative models attest to the importance of prior knowledge in reconstruction problems. When the input images miss some information that prevents a full reconstruction, as in single-view reconstruction), prior knowledge could help to fill in the blanks.

However, the problem with AtlasNet is that it assumes that a few prototypes of a concept (for example, planes) could be representative of all instances of such a concept. In other words, AltasNet makes **too strong inductive bias** but neglects many obtainable details hidden in the images, such as **depth**, **normal**, etc.

3.3 DVR

Differentiable Volumetric Rendering (DVR) is one of the early representative works in inverse rendering. They first train an occupancy network [6] to implicitly represent the object geometry and use volume rendering to solve the implicit surface. Then, they train second a RGB network to attach RGB color to the geometry [8].

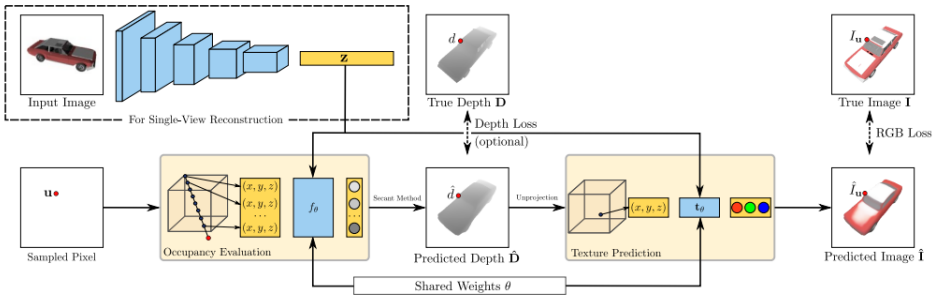

Figure 6: Differentiable Volumetric Rendering. Image from [8]

The two neural networks are essentially

$$\begin{aligned} NN_1 : \mathbb{R}^3 &\to \mathbb{R} \\ (x, y, z) &\mapsto d \\ NN_2 : \mathbb{R}^3 &\to \mathbb{R}^3 \\ (x, y, z) &\mapsto (R, G, B) \end{aligned}$$

DVR is one of the first works that introduce **rendering** and defines the loss function between the rendered images and the input images. It also clearly states the occupancy network as its **implicit 3D representation**. This greatly improves shape consistency when viewing from different perspectives. On the other hand, solving the implicit surface of an occupancy network requires almost bruteforce search along the ray, and attaching RGB colors to positions means that the position has the same color viewed from all directions. This results in the rendered images still artificial to human eyes.

3.4 NeRF

In 2020, Midenhall et al. proposed the very first Neural Radiance Fields (NeRF) that is still impactful today [7]. They combine the network for geometry and the network for RGB into a single MLP and use fully-differentiable volume rendering to render the images.

$$\begin{aligned} MLP : \mathbb{R}^5 &\to \mathbb{R}^4 \\ (x, y, z, \theta, \phi) &\mapsto (R, G, B, \sigma) \end{aligned}$$

where (x, y, z) gives the spatial position, (θ, ϕ) gives the viewing direction, and σ denotes the volume density.

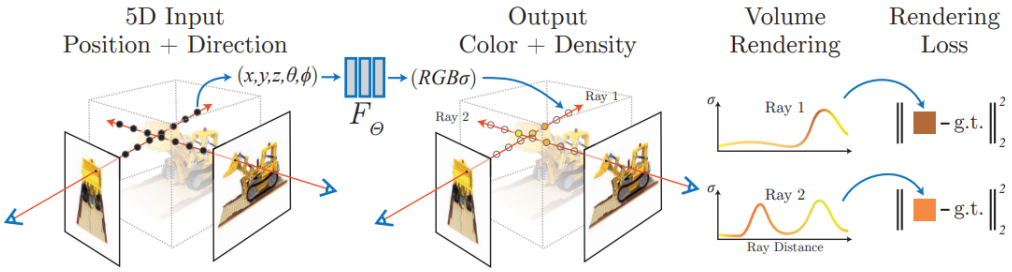

Figure 7: NeRF. Image from [7]

To some extent, one can think of NeRF as a neural version of forward mapping. At each position, NeRF trains a local neighborhood, denoted in their σ, and a function $(\theta, \phi) \mapsto (R, G, B)$ that memorizes colors of different viewing directions. At first, these local neighborhoods spread out evenly in space, but as the model trains, they converge to roughly the object surface.

The huge impact that NeRF brings essentially comes from that it can render images highly realistic to human eyes. Although they coined the term "neural radiance field" as their **3D representation**, we can see that NeRF is actually a direct upgrade of DVR as its neural network is more expressible. Their fully-differentiable **volume rendering** also generalizes the idea of α-compositing but does not limit the 3D representation to 2D images.

However, if we go back to the Image2Scene framework, we realize something is wrong. If NeRF has already accomplished the objective of optimizing the loss between the rendered images and the input images, then aren't we done with inverse rendering? In fact, the problem of NeRF lies in precisely its **3D representation** and **rendering** method that renders its images. We all know NeRF does not optimize directly the **geometry**, and a closer look at its generated depth map reveals many inaccurate places. Even though it mimics shading and reflection, these light effects are essentially memorized as RBG colors and do not generalize to other lighting conditions. In other words, the renderer does not leverage the important knowledge of **light transport**. NeRF represents a typical strategy that throws everything into the training set and lets the network find its way. However, we do see that **tackling specifically each of the components** in the scene would bring us much more piratical usage.

3.5 IDR

Following right after NeRF, Yariv et al. proposed Implicit Differentiable Renderer (IDR). Similar to DVR, they train one SDF network to implicitly represent the geometry, solve the implicit surface with sphere tracing, and train another neural renderer to attach implicit BSDF to the surface and render realistic images.

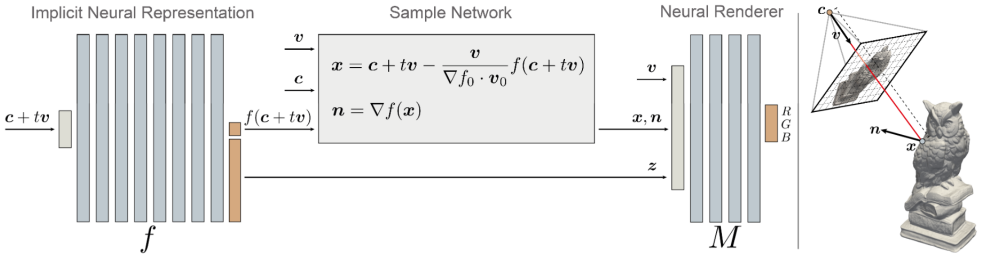

Figure 8: IDR. Image from [10]

As stated in the part, IDR is most closely related to DVR, but outperforms its predecessor in all aspects. In terms of **implicit 3D representation**, SDF contains more information than the occupancy network, and sphere tracing is more time efficient than bruteforce line search. More importantly, the surface normal is almost free in SDF because we can simply take the gradient of SDF. The Eikonal regularization also let SDF more nicely behaved. In addition, the notion of a **neural renderer** is novel in IDR. Different from the RGB network in DVR, the neural renderer in IDR is not specific to the scene. If we think of it as a learned function, then we see the neural renderer is yet more expressible than NeRF

$$NR : \mathbb{R}^7 \to \mathbb{R}^3$$
$$(\mathbf{p}, \mathbf{n}, \mathbf{v}) \mapsto (R, G, B)$$

It takes the position, **normal**, and viewing direction as input and outputs RGB colors. This also attests to IDR's **partial usage of light transport**— although IDR does not support light bounces, the incorporation of surface normal can be seen as leveraging Lambert's cosine law. We can also see that IDR clearly disentangles **geometry** from appearance. This not only solves the depth map problem in NeRF, but also allows the neural renderer to be applied across scenes.

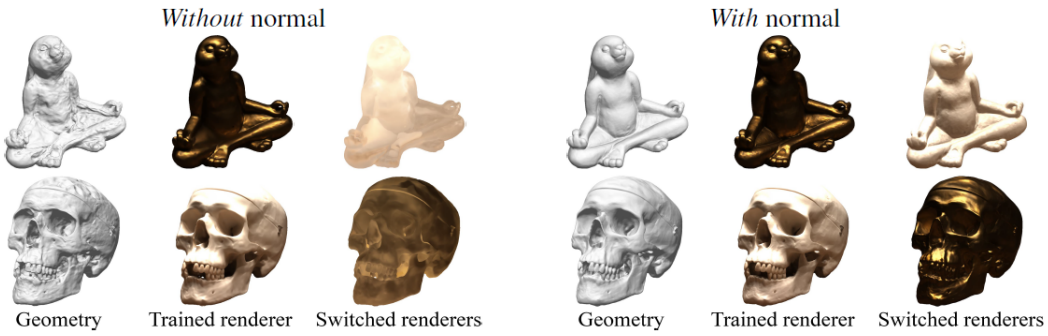

Figure 9: Switching neural renderers between scenes. Notice how disentangling geometry from appearance greatly boosts the neural renderer.

3.6 VolSDF

One year after IDR, the same group of people proposed VolSDF [11] as a combination of SDF and neural radiance field. VolSDF first trains an SDF network as usual, but then it uses the SDF to define the volume density and trains a second NeRF-style network. Compared to IDR, VolSDF changes the second network from

a neural renderer to a NeRF.

$$MLP : \mathbb{R}^7 \to \mathbb{R}^4$$
$$(\mathbf{p}, \mathbf{n}, \mathbf{v}) \mapsto (R, G, B, \sigma)$$

Contrary to the name, it might be more conducive to think of VolSDF as a NeRF-style work that leverages SDF to provide a good **initialization**. As discussed previously, one can think of NeRF as training many local neighborhoods (small balls) that reflects different colors from different viewing direction. The question naturally follows as to how to piece together these local neighborhoods. Since NeRF does not have an explicit geometry, although it can optimize the loss function well, in terms of faithfully reconstructing the scene NeRF often gets trapped in local minimums. Thus the **geometry** learned by the SDF network provides an excellent initial volume density to avoid local minimums.

It should be noted that in both IDR and VolSDF, a high-dimensional global feature vector is simultaneously trained as the output of the first SDF network and served as the input of the second appearance network. According to the authors, this feature vector helps to reason about global geometry. It can be seen as an encoded vector of the scene geometry, and its high dimensionality further increases the expressibility of the networks. It comes clear now that on one hand, we can throw everything as input and let the neural network find its way, as seen in NeRF as well as the global feature vector; or on the other hand, we can carefully **disentangle useful components** from the huge pile of messy data, and training the networks directly upon these key components would lead to more desirable results. This attests again to the importance of inductive bias from conventional Computer Graphics.

3.7 IRON

Taking a step further, Kai Zhang et al. proposed Inverse Rendering by Optimizing Neural scene components (IRON) that separates appearance in previous models into specific diffuse albedo, specular albedo, and specular roughness [13]. The training takes in two stages. In the first stage, IRON trains an SDF network for geometry and a NeRF for appearance, similar to previous works. In the second stage, IRON introduces two more networks, one for specular albedo and one for specular roughness, that separate the specular component from the NeRF network. IRON then fixes the view direction of the NeRF network to always be the normal.

$$NN : \mathbb{R}^5 \to \mathbb{R}^6$$
$$(\mathbf{p}, \mathbf{n}) \mapsto (R, G, B, \beta, \kappa, \alpha)$$

where β is the diffuse albedo, κ is the specular albedo, and α is the specular roughness.

In addition to the finer separation of appearance, IRON is also the first state-of-the-art model that incorporates differentiable rendering. This allows us to handle boundary edge discontinuities and thus obtain a more flexible **geometry**. Compared with volume rendering, differentiable rendering incorporates a standard path renderer that supports backpropagation into the inverse rendering pipeline. It tells clearly where the intersection happens so that we can better reason about the **albedo**. Although IRON has not supported full **light transport**, it partially leverages the knowledge to take a step forward in reconstructing appearance.

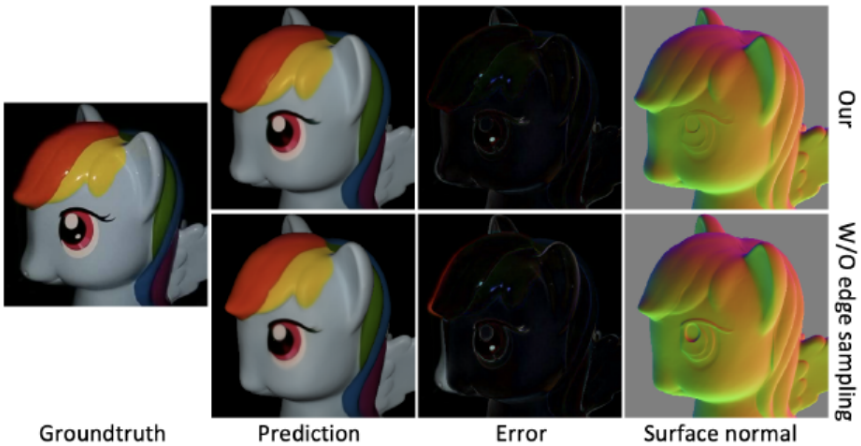

Figure 10: Comparison between with edge sampling and without edge sampling. Image from [13]

4 Summary

Over the main bulk of this manuscript, we propose a general Image2Scene framework and review the recent state-of-the-art models. We see how computer vision can be seen as starting from the left, while computer graphics is starting from the right. We also see that while "throwing everything into the training set and letting the network finds its way" first achieves wide attention, the follow-up models perform a more careful job separating key components from the messy data. Moreover, we see how these state-of-the-art models leverage more and more key components, and the expressibility of their models continuously grows. The major open question remaining in this field, as the author of this manuscript thinks, still lies in the absence of a renderer that supports full light transport, that is, bounces of light between scene objects. Toward this goal, a more time-efficient differentiable renderer that fits into the current inverse rendering pipeline is urgently needed.

References

[1] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. "Estimating and Exploiting the Aleatoric Uncertainty in Surface Normal Estimation". In: *International Conference on Computer Vision (ICCV)*. 2021.

[2] John Flynn et al. "DeepView: View Synthesis with Learned Gradient Descent". In: *CoRR* abs/1906.07316 (2019). arXiv: 1906.07316. URL: http://arxiv.org/abs/1906.07316.

[3] Thibault Groueix et al. *AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation*. 2018. DOI: 10.48550/ARXIV.1802.05384. URL: https://arxiv.org/abs/1802.05384.

[4] Tzu-Mao Li et al. "Differentiable Monte Carlo Ray Tracing through Edge Sampling". In: *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 37.6 (2018), 222:1–222:11.

[5] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. "Reparameterizing Discontinuous Integrands for Differentiable Rendering". In: *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38.6 (Dec. 2019). DOI: 10.1145/3355089.3356510.

[6] Lars Mescheder et al. "Occupancy Networks: Learning 3D Reconstruction in Function Space". In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[7] Ben Mildenhall et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *ECCV*. 2020.

[8] Michael Niemeyer et al. "Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision". In: *CoRR* abs/1912.07372 (2019). arXiv: 1912.07372. URL: http://arxiv.org/abs/1912.07372.

[9] Jeong Joon Park et al. "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[10] Lior Yariv et al. "Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance". In: *Advances in Neural Information Processing Systems* 33 (2020).

[11] Lior Yariv et al. "Volume rendering of neural implicit surfaces". In: *Thirty-Fifth Conference on Neural Information Processing Systems*. 2021.

[12] Cheng Zhang et al. "Path-Space Differentiable Rendering". In: *ACM Trans. Graph.* 39.4 (2020), 143:1–143:19.

[13] Kai Zhang et al. "IRON: Inverse Rendering by Optimizing Neural SDFs and Materials from Photometric Images". In: *IEEE Conf. Comput. Vis. Pattern Recog.* 2022.