

**ILLUMINATION-GUIDED INVERSE RENDERING
BENCHMARK:
LEARNING REAL OBJECTS WITH FEW CAMERAS**

A Thesis

by

Doğ̃a Yılmaz

Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Artificial Intelligence

Özyeğ̃in University
October 2023

Copyright © 2023 by Doğ̃a Yılmaz

**ILLUMINATION-GUIDED INVERSE RENDERING
BENCHMARK:
LEARNING REAL OBJECTS WITH FEW CAMERAS**

Approved by:

Assist. Prof. Dr. Furkan Kır  
Department of Computer Science
  zyeđ n University

Assist. Prof. Dr. Reyhan Aydođ n
Department of Computer Science
  zyeđ n University

Assist. Prof. Dr. Berk G  kber  
Department of Computer Engineering
Bođazi i University

Date Approved: 11th August 2023



To those who climb the stairs to their dreams.

ABSTRACT

The field of 3D computer vision and graphics has seen a rapid expansion of late, making the development of realistic virtual environments and digital representations of real-world objects possible. Fundamental to this development are 3D reconstruction techniques that enable the transition of physical objects’ form, color, and surface particulars to the virtual domain. Current approaches mainly rely on neural scene representations, which, despite being effective, face challenges such as the need for a large quantity of captured images and the complexity associated with converting these representations into explicit geometric forms.

An alternative strategy that has gained traction is the deployment of methods such as physically-based differentiable rendering (PBDR) and inverse rendering. These approaches require fewer viewpoints, yield explicit format results, and ensure a smoother transition to other representation methods. However, in order to effectively assess the performance of the available methods in 3D reconstruction, it is imperative to utilize standard benchmark scenes for comparison.

Although there are standard objects and scenes available in existing research, there is a noticeable deficiency of real-world benchmark data that simultaneously captures camera, lighting, and scene parameters — all of which are essential for high-quality 3D reconstructions using methods based on PBDR and inverse rendering. In this study, we present a method for capturing real-world scenes as virtual environments, incorporating lighting parameters along with camera and scene parameters to enhance the veracity of virtual representations. In addition, we provide a set of ten real-world scenes, each with its corresponding virtual counterparts, purposely designed as benchmarks. These benchmarks cover a basic assortment of geometric structures, such as convex, concave, flat, and composite surfaces. Furthermore, we showcase the 3D reconstruction results of cutting-edge 3D reconstruction techniques using PBDR in real-world scenes, using both established methodologies and our proposed one.

ÖZETÇE

Son dönemde 3B bilgisayar görüsü ve grafik alanında hızlı bir genişleme yaşanmış olup, gerçekçi sanal ortamların ve gerçek dünya nesnelerinin dijital temsillerinin geliştirilmesini mümkün kılmıştır. Bu gelişmenin temelinde, fiziksel nesnelerin şekil, renk ve yüzey özelliklerinin sanal ortama aktarılmasını mümkün kılan 3B rekonstrüksiyon teknikleri yer almaktadır. Mevcut yaklaşımların çoğunluğu, genellikle etkili olsalar da, büyük miktarda görüntüye ihtiyaç duymaları ve ürettikleri 3D rekonstrüksiyonun standart formatlara çevrilmesine ilişkin zorluklarla karşı karşıyadır.

Son dönemde dikkat çeken alternatif bir strateji, fiziksel temelli türevlenebilir renderleme (FTTR) ve ters renderleme gibi yöntemlerin kullanılmasıdır. Bu yaklaşımlar, daha az bakış açısı gerektirir ve standart formatlarda sonuçlar verir. Bununla birlikte, 3B rekonstrüksiyondaki mevcut yöntemlerin performansını etkili bir şekilde değerlendirebilmek için standart ölçüt sahneleri kullanmak son derece önemlidir.

Mevcut araştırmalarda standart nesne ve sahneler olsa da, FTTR ve ters renderleme tabanlı yöntemlerle yüksek kaliteli 3B rekonstrüksiyonlar için gereken kamera, aydınlatma ve sahne parametrelerini aynı anda yakalayan gerçek dünya ölçüt verilerinin belirgin bir eksikliği bulunmaktadır. Bu çalışmada, gerçek dünya sahnelerinin ışık parametrelerini de içeren sanal ortamlar olarak yakalanması için bir yöntem sunmaktayız. Ayrıca, temel geometrik yapıların, yani konveks, konkav, düz ve bileşik yüzeylerin yer aldığı bir dizi gerçek dünya sahnesi ve bunların karşılık gelen sanal versiyonlarını içeren bir ölçüt seti sunmaktayız. Ayrıca, FTTR kullanarak son teknoloji 3B rekonstrüksiyon tekniklerinin gerçek dünya sahnelerindeki sonuçlarını, mevcut yöntemlerle birlikte önerdiğimiz metodolojileri kullanarak sergilemekteyiz.

ACKNOWLEDGEMENTS

Firstly, I wish to express my profound gratitude to my advisor, Assist. Prof. Dr. Furkan Kır    . His guidance and mentorship during my years at the Vision and Graphics Laboratory (VGL) have been invaluable. Furthermore, his outstanding courses have provided me the opportunity to obtain a distinctive skill set that is valuable in both academic and industry settings.

Next, I extend my thanks to the rest of my thesis committee: Assist. Prof. Dr. Reyhan Aydo    an and Assist. Prof. Dr. Berk G          . Their valuable insights and contributions have significantly enriched this research.

I am grateful to my fellow colleagues at VGL for fostering a supportive and vibrant lab culture. Their collective efforts have amplified the impact of our research and personal development beyond individual capabilities.

I want to convey my special thanks to Bar           for his unwavering support during my undergraduate and postgraduate years. His contributions have been instrumental in shaping my academic journey from the early stages.

In addition, I wish to acknowledge Furkan K      , an exceptional colleague and friend, for his continual support and companionship. I also extend my thanks to Berk Buzcu, my constant lab companion, who has made this journey much more pleasant.

Furthermore, I am grateful to my friends and family. Their support, genuine belief in my potential, and unending patience throughout my research process have been the pillar of my perseverance.

Lastly, I express heartfelt thanks to Berrak Alara Baransel. Her patience in listening to countless hours of brainstorming has been greatly appreciated and has contributed to shaping my research ideas.

TABLE OF CONTENTS

DEDICATION	iii
ABSTRACT	iv
ÖZETÇE	v
ACKNOWLEDGEMENTS	vi
LIST OF TABLES	x
LIST OF FIGURES	xi
I INTRODUCTION	1
II BACKGROUND INFORMATION	4
2.1 Representing 3D Shapes and Scenes	5
2.1.1 Triangular Mesh Representation	5
2.1.2 Point Cloud Representation	6
2.1.3 Volumetric Representation	7
2.1.4 Signed Distance Function Representation	8
2.1.5 Neural Scene Representation	9
2.2 Rendering Pipeline	10
2.3 Rendering Algorithms	11
2.3.1 Rasterization	11
2.3.2 Ray-Casting	12
2.3.3 Ray-Tracing	13
2.4 Physically-based rendering	14
2.5 Differentiable Learning	14
2.5.1 Loss Functions	14
2.5.2 Optimization Algorithms	16
2.5.3 Neural Networks	17
2.6 Inverse Rendering	18

2.7	Differentiable Rendering	19
2.8	Discontinuities in Differentiable Rendering	19
2.9	Novel View Synthesis	20
2.10	3D Scene Reconstruction	21
2.11	Ambiguities in Interpreting 3D Shapes from a 2D Perspective	22
III	LITERATURE REVIEW	24
3.1	Representing 3D Geometry and Scene with NeRFs	24
3.2	Representing 3D Geometry and Scene with SDFs	25
3.3	Scene Optimization via Differentiable Rendering	25
3.4	3D Reconstruction	27
3.4.1	Multi-View 3D Reconstruction	28
3.4.2	Single-View 3D Reconstruction	28
3.4.3	Differentiable Rendering Based 3D Reconstruction	29
3.5	3D Reconstruction Using SDFs	30
3.6	Datasets	31
3.7	Applications of 3D Reconstruction	32
IV	METHODOLOGY	33
4.1	Creating Virtual Scenes	33
4.1.1	Collecting Scene Parameters via Video	34
4.1.2	Collecting Scene Parameters via Multiple Cameras	35
4.2	Benchmark Scenes	36
4.3	Baseline 3D Reconstruction Methods for Real-World Objects	38
V	RESULTS AND DISCUSSION	41
5.1	Experiments and Results on Single Turntable Video	41
5.2	Experiment Results on Benchmark Scenes	43
5.3	Effects of Different Loss Functions	45
5.4	Effects of Number of View	46
5.5	Effects of Correctly Capturing Illumination of The Real-World Scene	47

5.6 Current Limitations	48
VI CONCLUSION	50
APPENDIX A — ADDITIONAL RESULTS	52
REFERENCES	57
VITA	62



LIST OF TABLES

1	Overview of available datasets.	31
2	Details of the benchmark scenes. For each scene, we used four synchronized cameras and eight LED light sources with their respective intrinsic and extrinsic parameters. The original images were captured in the real-world scene, while the reference images were obtained by transferring the object segment onto a render of the empty virtual-scene from the corresponding virtual-camera.	37
3	Experiment Parameters	42
4	Quantitative results for the toucan and the rubber duck scenes. . . .	43
5	Quantitative results of the experiments conducted using our benchmark scenes. For each method and metric, we report the average and standard deviation across all benchmark scenes. The experiment names follow the format: "loss function used in optimization - reparameterization method". In each column, the best results are denoted in bold typeface and highlighted in green , while the second-best results are highlighted in yellow	45
6	Raw quantitative results of the experiments conducted using our benchmark scenes. The experiment name is presented in the following format: Scene Name - Experiment Preset. Experiment presets are denoted as following: A: L1 - Vicini <i>et al.</i> [1] B: L2 - Vicini <i>et al.</i> [1] C: Multi-Scale L1 - Vicini <i>et al.</i> [1] D: Multi-Scale L1 + SSIM - Vicini <i>et al.</i> [1] E: Multi-Scale L1 + Silhouette - Vicini <i>et al.</i> [1] F: L1 - Bangaru <i>et al.</i> [2] G: L2 - Bangaru <i>et al.</i> [2] H: Multi-Scale L1 - Bangaru <i>et al.</i> [2] I: Multi-Scale L1 + SSIM - Bangaru <i>et al.</i> [2] K: Multi-Scale L1 + Silhouette - Bangaru <i>et al.</i> [2]	56

LIST OF FIGURES

1	Representation of an Example Synthetic 3D Scene: The first image illustrates the scene parameters, namely meshes, textures, lights, and cameras. The second image, on the other hand, shows the rendering output generated using the scene parameters displayed in the first image.	4
2	Triangular Mesh Representation of the Stanford Bunny [3]: The red circle highlights a magnified portion of the marked region. The entire mesh is composed of a set of vertices, edges, and faces that define the shape of the object. A single face is marked in red within the magnified area.	5
3	Point Cloud Representation of the Stanford Bunny [3]: The red circle highlights a magnified portion of the marked region. The structure is composed of a set of points that define the shape of the object. A single point is marked in red within the magnified area.	6
4	Volumetric Representation of the Stanford Bunny [3]: The red circle highlights a magnified portion of the marked region. The structure is composed of a set of voxels that define the shape of the object. A single voxel is marked in red within the magnified area.	7
5	A circle represented using SDF. The value of the signed distance function is 0 when the point is on the circle, positive when the point is outside the circle, and negative when the point is inside the circle. The magnitude of the positive or negative value indicates the distance to the boundary of the circle.	8
6	Neural Representation of the Stanford Bunny [3]: The area inside the dashed-lined box is represented by a fully connected neural network. When queried with the corresponding camera parameters, as illustrated in blue, the neural network outputs the observed image from that viewpoint. Figure taken from [4].	9
7	Steps of a Typical Rendering Pipeline: The illustration depicts the sequence of processes involved in a standard rendering pipeline. . . .	11
8	Visualization of the Rasterization Operation: As depicted, each point in the scene is directly projected onto the image plane.	11
9	Visualization of the Ray-Casting Operation: Starting from the image plane, rays are projected from each pixel on the plane until they intersect with an object in the scene. The color of each pixel on the image plane is computed using the color of the surface that the corresponding ray, projected from that pixel, intersects with.	12

10	Visualization of the Ray-Tracing Operation: Starting from the image plane, rays are projected from each pixel on the plane towards the scene. Upon hitting a triangle, they reflect off it and proceed in a different direction. The number of reflections permitted corresponds to the depth of the ray tracing. The color of each pixel on the image plane is computed using the combined results from these reflections.	13
11	Visualization of a Multi-Layer Perceptron (MLP): The figure illustrates a typical MLP architecture with an input layer, multiple hidden layers, and an output layer. Data flows from left to right, starting from the input layer and moving through the hidden layers via weighted connections, before reaching the final output layer. Each node in the hidden layers applies a non-linear activation function to transform its inputs. Figure taken from [5].	17
12	The scene description is illustrated on the left-hand side. On the right-hand side, the output of the forward rendering process is displayed. The inverse of this operation, known as inverse rendering, aims to generate the scene description from a given rendered image.	18
13	Visualization of a Differentiable Rendering Loop: The figure presents the continuous loop of the differentiable rendering process. Starting with a set of parameters, the scene is rendered, and the result is compared with a target image. The difference is then used to adjust the initial parameters through back-propagation, closing the loop and initiating a new rendering process. This iterative process continues until the rendered image closely matches the target.	19
14	Visualization of the Novel View Synthesis Task: The figure illustrates the process of generating a target view from a given target pose, using the provided source views and source poses. Initially, several source views and their corresponding poses exist. Given a new target pose, the process synthesizes a novel view, referred to as the target view, representing the scene as it would appear from the perspective of the target pose. Figure taken from [6].	20
15	Visualization of the 3D Scene Reconstruction Task: The figure demonstrates the process of reconstructing a 3D scene from multiple input images. On the left, several input images are displayed, which serve as the basis for the reconstruction. On the right, the resulting image showcases the successfully reconstructed 3D scene, providing a comprehensive view of the spatial arrangement and depth of the scene's elements.	21

16	Interpreting 3D Shapes from a 2D Perspective: This figure illustrates the inherent ambiguity in interpreting 3D geometries from a 2D projection. Given a single 2D perspective, an infinite array of 3D geometries could potentially align with the observed projection. Figure sourced from [7].	22
17	Illustration of the data collection process using the methodology proposed by Reizenstein <i>et al.</i> [8]. Images of the target object from several viewpoints are shown on the left side. On the right side, an illustration of the data collection process is shown, where the blue object represents the target object, and the camera models represent the trajectory of the real camera while capturing a video of the target object. . . .	34
18	Our multi-camera setup which is used to collect data for our benchmark scenes. The left side shows the setup in the real world, while the right side displays the virtual counterpart of the test setup. In the virtual scene, the camera models represent virtual cameras, while the yellow boxes represent diffused LED bars, and the blue object represents the target object for 3D reconstruction.	35
19	Overview of our pipeline for real-world 3D reconstruction. Boxes with dashed lines represent updated elements throughout the optimization loop. We capture target object images using our real-world environment, and use them to generate reference images. Virtual initial scenes and cameras are constructed to match their real-world counterparts. After initializing the virtual cameras and scene, scene parameters are updated based on the difference between the rendered and reference images.	39
20	Single video results obtained using a turntable video of the target object. The first row presents the results generated from the toucan scene, while the second row displays the results obtained from the rubber duck scene. The optimization is conducted for 512 iterations with the multi-scale L1 loss. The illumination in the scene is assumed to be constant at every point.	42
21	Multi-camera setup results obtained using four synchronized images of the target object. The rows marked with A presents the results obtained using proposed method by Vicini <i>et al.</i> [1]. The rows marked with B presents the results obtained using proposed method by Bangaru <i>et al.</i> [2].	44
22	The first row shows the 3D reconstruction results using L1, Multi-scaled L1, Multi-scaled L1 + SSIM, and Multi-scaled L1 + Silhouette loss functions, respectively. The second row indicates the magnification of the section inside the red box in the same column.	46

23	Qualitative results of the number of view experiment: The first row is rendered from an exact viewpoint of one of the cameras. The second row represents a viewpoint different from the camera viewpoints used in the optimization. Rows A, B, C, and D correspond to 4, 3, 2, and 1 number of cameras, respectively, which were used in the optimization.	47
24	Qualitative results of the illumination ablation study: Columns A and C are rendered after 512 iterations in a correctly matched illumination setting with the real-world scene. Columns B and D are rendered after 512 iterations in a non-matching illumination setting with the real-world scene. The second row shows the magnification of the section inside the red box in the first row, in the same column.	48
25	The first image is the reference image generated using real-world data. The second image is the state of the scene at the end of the optimization from the viewpoint of the virtual camera. The third image is the state of the optimized scene from a novel viewpoint.	49
26	Single video setup results obtained using six non-synchronized images of the target object. Obtained using proposed method by Vicini <i>et al.</i> [1].	52
27	Multi-camera setup results obtained using four synchronized images of the target object. The rows marked with A presents the results obtained using proposed method by Vicini <i>et al.</i> [1]. The rows marked with B presents the results obtained using proposed method by Bangaru <i>et al.</i> [2].	53
28	Multi-camera setup results obtained using four synchronized images of the target object. The rows marked with A presents the results obtained using proposed method by Vicini <i>et al.</i> [1]. The rows marked with B presents the results obtained using proposed method by Bangaru <i>et al.</i> [2].	54
29	Multi-camera setup results obtained using four synchronized images of the target object. The rows marked with A presents the results obtained using proposed method by Vicini <i>et al.</i> [1]. The rows marked with B presents the results obtained using proposed method by Bangaru <i>et al.</i> [2].	55

CHAPTER I

INTRODUCTION

Advancements in 3D computer vision and graphics have enabled the creation of immersive virtual environments, digital replicas of real-world objects, and accurate simulations of physical phenomena. Methods for 3D reconstruction are of increasing interest due to their ability to accurately transfer shape, color, and surface roughness information from the real world to the virtual environment.

Various approaches exist to achieve this kind of conversion between the virtual and real world. A widely employed technique is to optimize the parameters of a scene representation directly to minimize the error of rendering a set of captured images. In the literature, these methods mostly utilize neural radiance fields (NeRF) [9] to represent the target scene. These methods are able to reconstruct the target scene with high fidelity; however, there are several limitations due to their inherent design. Firstly, these approaches necessitate a continuous stream of densely captured images, which may be unavailable for certain target scenes. Additionally, as a result of their use of neural implicit scene representation, converting the scene and its constituent objects to explicit formats for compatibility with conventional geometry representation methods poses a challenge.

Another widely used method for scene optimization is physically-based differentiable rendering (PBDR). One of the key challenges faced by PBDR-based methods is the handling of discontinuities that arise at the boundaries of occluding objects. These disruptions, if not properly accounted for, can introduce significant biases in the derivatives used in the optimization of scene geometry. However, with recent

enhancements in the computer graphics literature, through sophisticated reparameterization techniques, the issues associated with discontinuous integrals introduced by the renderer have been minimized. As a result, the remaining calculation can be handled using standard automatic differentiation techniques.

These methods are capable of reconstructing the target scene using a limited number of viewpoints while offering a comparable level of fidelity with respect to NeRF-based techniques. Moreover, since these methods often use a base shape and material and iteratively optimize their shape and surface parameters to reconstruct the target scene, the output type of these approaches is bounded by the selection of explicit geometry representation method. Consequently, the resulting format of the optimized scene is explicit, and it can be converted to any other desired representation method with ease. However, examples of these methods [1, 2] are limited to virtual scenes, as their usage in real-world scenes requires the construction of a real-world scene and its subsequent digital replica.

The applications of PBDR-based 3D reconstruction methods yield significant advantages across a spectrum of domains where the number of views of the target object is limited. These domains include, but are not limited to, the generation of digital assets for computer graphics applications and scene understanding in robotics.

Our work primarily targets the application of cutting-edge 3D reconstruction techniques to real-world scenes that are constrained by limited viewpoint availability. Current methods, as referenced in [1, 2], are typically demonstrated using virtual scenes due to a lack of datasets containing both camera and light source locations. Presently, existing datasets provide images captured from various viewpoints in controlled environments. To our knowledge, no available dataset combines images from multiple known viewpoints with light source location information within such an environment.

To remedy this, we’ve developed a real-world dataset that includes both camera and light source positions in a controlled setting. Additionally, we identify and

address challenges emerging when transitioning from virtual to real-world scenes, offering possible solutions. One key issue involves the loss function utilized by current state-of-the-art methods, which tends to be effective exclusively within virtual scenes. Consequently, we also aim to refine this loss function within our work’s scope, enabling these advanced 3D reconstruction methods to better adapt to real-world situations with restricted viewpoint availability.

In summary, our contributions include:

- Implementing cutting-edge 3D Reconstruction techniques [1, 2] designed for virtual scenes in real-world scenarios, addressing any emerging challenges.
- Introducing a method to capture real-world scenes as virtual ones, incorporating light source and camera data to better mirror reality virtually.
- Offering a selection of real-world scenes alongside their virtual equivalents, designed to serve as benchmarks.
- Proposing a new method that attempts to improve the loss computation of existing state-of-the-art methods by incorporating a structural component into the image-based loss function, with the goal of addressing issues specific to real-world scenes.
- Presenting the 3D reconstruction outcomes from applying cutting-edge methods to real-world scenes, using both existing techniques and our proposed method. Furthermore, we demonstrate the impact of including light information in the scene optimization process.

CHAPTER II

BACKGROUND INFORMATION

In the upcoming section, we offer a brief overview of the key principles underpinning 3D shape and scene representation, rendering algorithms, differentiable learning, inverse rendering, differentiable rendering, novel view synthesis, and 3D reconstruction.

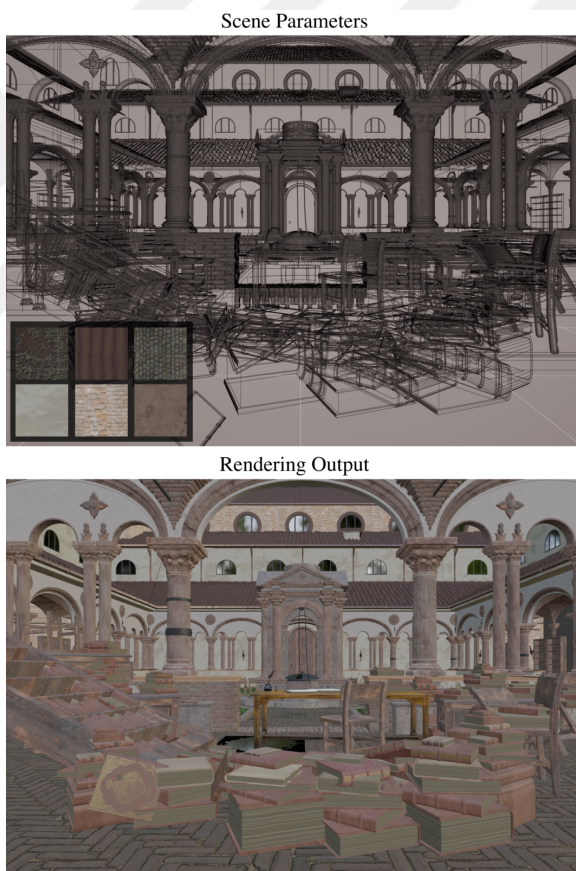


Figure 1: Representation of an Example Synthetic 3D Scene: The first image illustrates the scene parameters, namely meshes, textures, lights, and cameras. The second image, on the other hand, shows the rendering output generated using the scene parameters displayed in the first image.

2.1 Representing 3D Shapes and Scenes

In the realm of computer graphics, scenes and inherent shapes of the scenes are characterized by the geometry, texture, and material parameters of the objects it encompasses. Geometry representation is achieved by modeling an object’s or scene’s shape using a suitable representation technique, while texture representation relies on the employed rendering technique.

Various methods are employed in computer graphics and computer vision to represent, compute, and process surfaces. In the literature, the most commonly used explicit surface representation methods are triangular meshes, point clouds and volumetric representations. As an alternative to these explicit representation methods, there are also implicit representation methods. Most representative of these implicit methods are signed distance function representation and neural scene representation. An illustration of representing 3D Shapes and Scenes is shown in Figure 1.

2.1.1 Triangular Mesh Representation

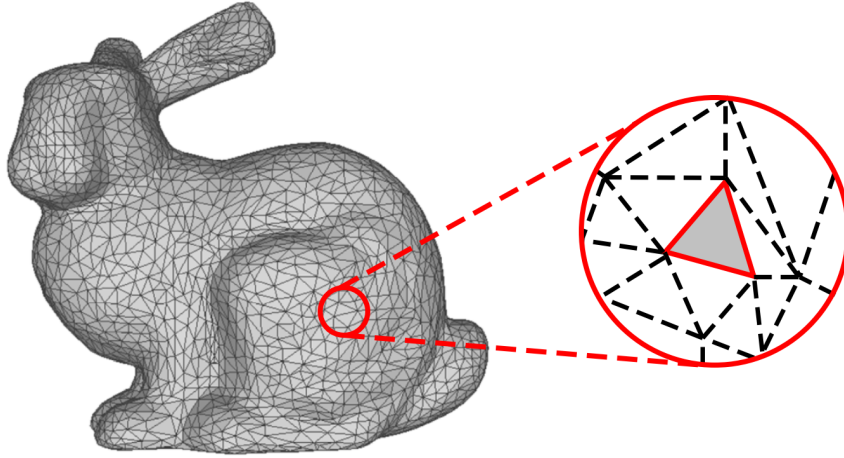


Figure 2: Triangular Mesh Representation of the Stanford Bunny [3]: The red circle highlights a magnified portion of the marked region. The entire mesh is composed of a set of vertices, edges, and faces that define the shape of the object. A single face is marked in red within the magnified area.

In a triangular mesh, a 3D model is represented as a collection of interconnected triangles. Each triangle is defined by its vertices. The vertices are 3D points in space, usually represented by their (x, y, z) coordinates. The collection of all vertices is called the "vertex set". The triangles are then defined by groups of three vertices, and the collection of all triangles is called as "face set". The manner in which the triangles are connected to each other defines the surface of the 3D model. Figure 2 visualises an example surface represented using mesh representation.

An additional piece of information that is often stored with each vertex is its "normal", a vector that is perpendicular to the surface of the model at the vertex location. Normals are used for lighting calculations, as they can determine how light interacts with the surface, producing effects such as shading and specular highlights.

2.1.2 Point Cloud Representation

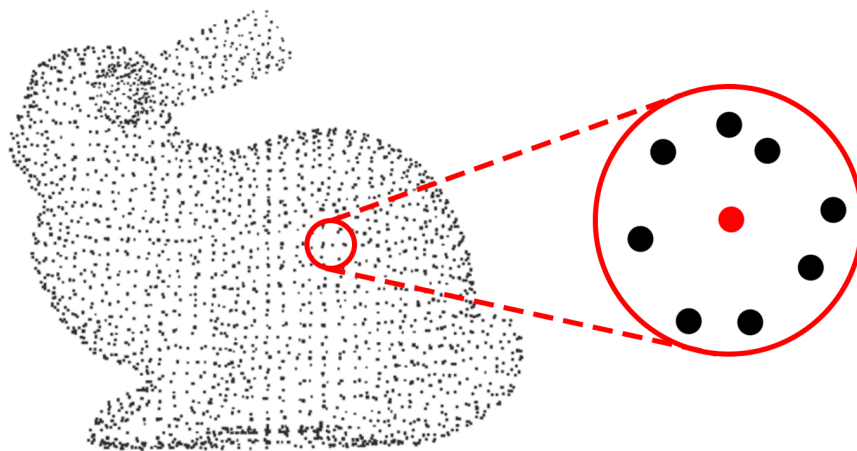


Figure 3: Point Cloud Representation of the Stanford Bunny [3]: The red circle highlights a magnified portion of the marked region. The structure is composed of a set of points that define the shape of the object. A single point is marked in red within the magnified area.

Each point in the point cloud represents an actual point in the object or the scene, and the overall collection of points gives the 3D digital representation of the

scene. However, unlike the triangular meshes, point clouds do not inherently have connectivity information between the points – they are a ‘cloud’ of points without explicit information about which points are next to each other to form a surface. Figure 3 visualises an example surface represented using point cloud representation.

2.1.3 Volumetric Representation

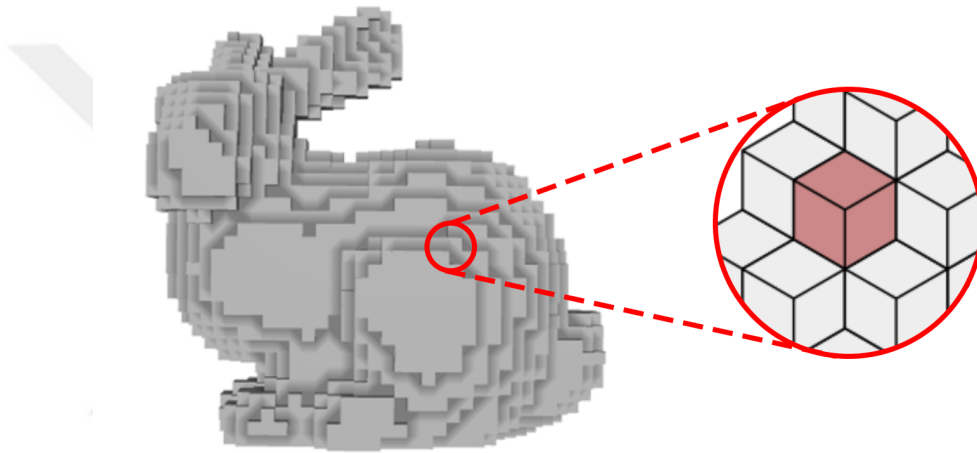


Figure 4: Volumetric Representation of the Stanford Bunny [3]: The red circle highlights a magnified portion of the marked region. The structure is composed of a set of voxels that define the shape of the object. A single voxel is marked in red within the magnified area.

While triangular meshes and point clouds focus on representing the surface of objects, volumetric representations consider the entire volume of the object. They divide the object’s volume into discrete volumetric elements (often called “voxels” in analogy with pixels in a 2D image), much like a 3D grid. Each voxel in the grid is assigned a value representing a property of the object at that location in the grid. This property could be, for example, the material density, or a binary value indicating whether the voxel is inside or outside the object. More complex volumetric representations might include additional properties such as color or texture information. Figure 4 visualises an example surface represented using volumetric

of Ω . In Equation 1, $d(x, \Omega)$ denotes the Euclidean distance function from the point x to the surface Ω . A visualization of the SDF representation of a circle can be seen in Figure 5.

$$f(x) = \begin{cases} d(x, \Omega) & \text{if } x \text{ lies outside of } \Omega \\ 0 & \text{if } x \text{ lies on } \Omega \\ -d(x, \Omega) & \text{if } x \text{ lies inside of } \Omega \end{cases} \quad (1)$$

2.1.5 Neural Scene Representation

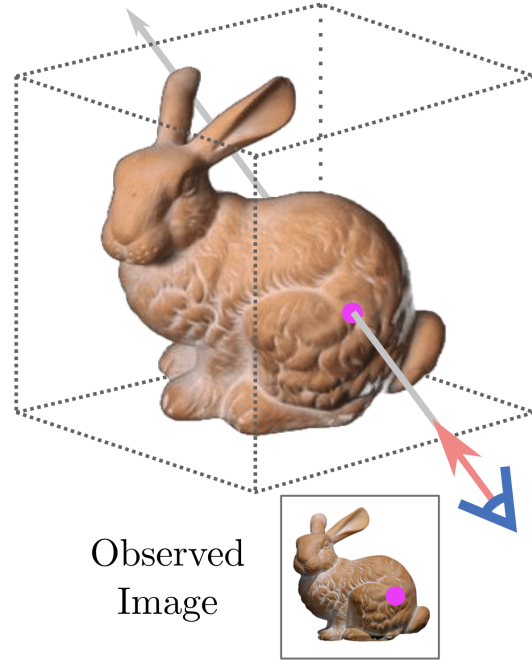


Figure 6: Neural Representation of the Stanford Bunny [3]: The area inside the dashed-lined box is represented by a fully connected neural network. When queried with the corresponding camera parameters, as illustrated in blue, the neural network outputs the observed image from that viewpoint. Figure taken from [4].

Representing a scene or an object with a neural representation is to encode the geometry, appearance, and semantic understanding of it within the parameters of a neural network. The most significant benefit of using neural networks is that they

can capture complex structures and patterns that may be challenging to model with traditional 3D representation techniques.

Learning the representation of the target scene is accomplished by training a neural network to predict the properties of a scene given a viewing direction and a location in space. Thus, unlike the other methods, neural scene representation methods require a training process for each individual scene or object to capture the geometry, appearance, and semantics of them.

Recently, with the advent of neural scene representation methods [9, 10, 11, 12, 13], Neural Radiance Fields (NeRF) have emerged. A Neural Radiance Field is a fully-connected neural network that acts as a continuous volumetric scene function. It takes as input a 3D coordinate and a viewing direction, and outputs the volume density and emitted radiance at that location. These methods characterize the scene as a continuous 5D function and can represent the underlying scene with high precision. However, their representation remains implicit. Consequently, the feasibility of manipulating and retrieving objects within the scene in an explicit form using this approach is highly limited. Figure 6 visualises an example scene represented using neural scene representation.

2.2 Rendering Pipeline

Rendering is a multi-stage process that transforms 3D geometric models into 2D images or animations that represent a virtual environment from a specific viewpoint. This process emulates how the human eye perceives the real world, simulating the behavior of light and its interaction with objects. The quality and realism of rendered images largely depend on how well this process mimics real-world physical phenomena. Figure 7 illustrates the steps of the rendering pipeline.

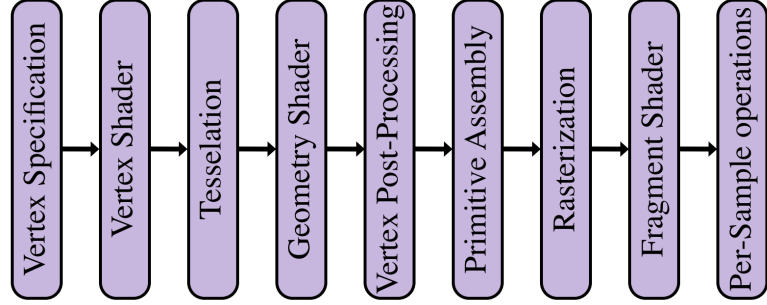


Figure 7: Steps of a Typical Rendering Pipeline: The illustration depicts the sequence of processes involved in a standard rendering pipeline.

2.3 *Rendering Algorithms*

Rendering algorithms are methods that generate a digital image from a 3D model or scene description by simulating the physical behavior of light and its interaction with objects in the virtual environment. The main techniques of rendering can be listed as rasterization [14], ray-casting [15] and ray-tracing [16].

2.3.1 Rasterization

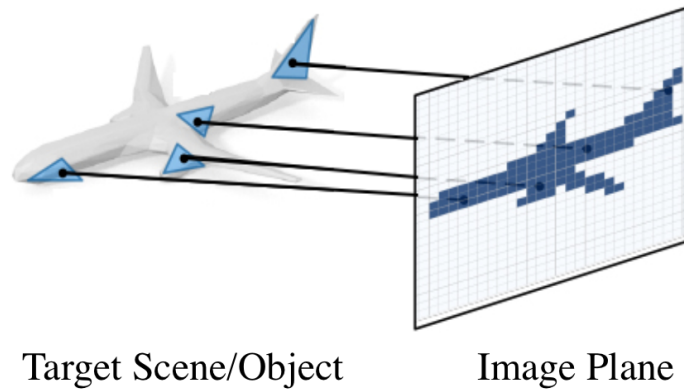


Figure 8: Visualization of the Rasterization Operation: As depicted, each point in the scene is directly projected onto the image plane.

Rasterization, also referred to as scanline rendering, is a technique in real-time graphics. This method transforms vector graphics, represented in a geometric data structure, into a raster image in the form of pixels.

The main advantage of rasterization is its efficiency and speed, facilitated by dedicated hardware like the Graphics Processing Unit (GPU). However, it lacks in producing physically accurate lighting effects, as it does not consider the paths of light rays in its process which is a crucial aspect of mimicking real-world physical phenomena. Thus, the fidelity of rendered images using this method is not on-par with methods which use light models for the rendering process. Figure 8 illustrates the rasterization process.

2.3.2 Ray-Casting

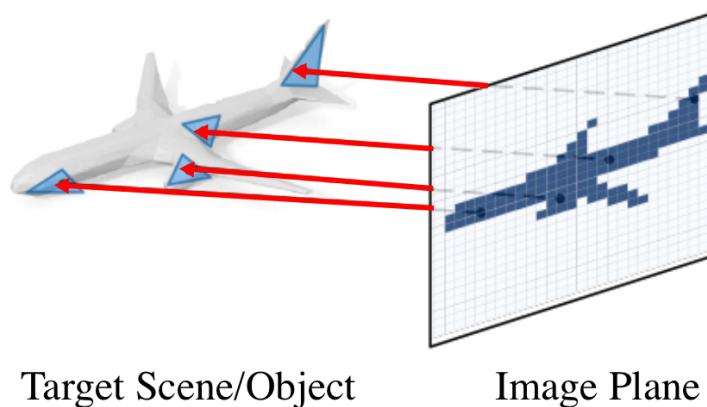


Figure 9: Visualization of the Ray-Casting Operation: Starting from the image plane, rays are projected from each pixel on the plane until they intersect with an object in the scene. The color of each pixel on the image plane is computed using the color of the surface that the corresponding ray, projected from that pixel, intersects with.

Ray-casting is a rendering method that, unlike rasterization, considers the path of light to create an image. Ray-casting involves sending rays from a virtual camera into a scene and computing the color of the pixel based on the first object the ray

intersects. Although ray-casting provides more realistic visuals than rasterization, it is less efficient, making it less suitable for real-time rendering. Figure 9 illustrates the ray-casting process.

2.3.3 Ray-Tracing

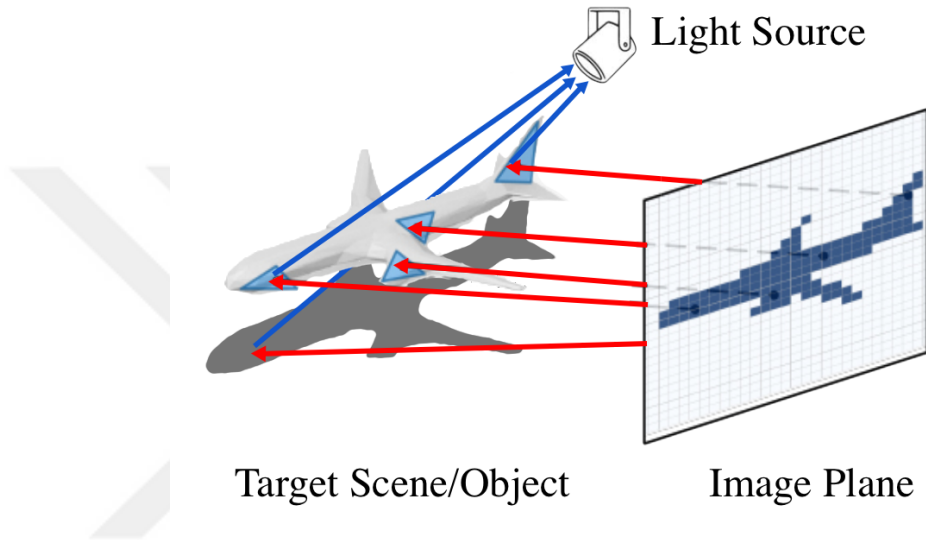


Figure 10: Visualization of the Ray-Tracing Operation: Starting from the image plane, rays are projected from each pixel on the plane towards the scene. Upon hitting a triangle, they reflect off it and proceed in a different direction. The number of reflections permitted corresponds to the depth of the ray tracing. The color of each pixel on the image plane is computed using the combined results from these reflections.

Ray-tracing is a more sophisticated and computationally intense technique compared to ray-casting. In ray-tracing, rays of light are traced back from the camera into the scene, allowing for the simulation of complex light paths and effects such as reflection, refraction, and shadows. This results in more photorealistic images, albeit at a higher computational cost. Although being computationally expensive for real-time rendering, advances in dedicated hardware and algorithms over the past years have brought real-time ray-tracing within reach. Figure 10 illustrates the ray-tracing process.

2.4 *Physically-based rendering*

Physically-based rendering involves the computation of pixel intensity through an integration over the space of light paths $\mathcal{P}$ [17]. In Equation 2, x represents a light path that includes pertinent information about the scene, such as shape parameters and texture values. The image contribution function, denoted by f is utilized to evaluate the level of contribution of a given light path to a particular pixel.

$$I_j(\pi) = \int_{\mathcal{P}} f_j(x, \pi) dx \quad (2)$$

2.5 *Differentiable Learning*

Differentiable learning, is a subset of artificial intelligence that enables a system to improve its performance on tasks over time by leveraging statistical methods to learn from data. It is built upon the premise that algorithms can be trained to make predictions or decisions without being explicitly programmed to do so. One core characteristic of differentiable learning is the concept of "differentiability," or the ability to find the derivative of a function, which is fundamental to most machine learning algorithms' optimization techniques. This plays a significant role in functions where the derivative provides a measure of how much a function changes in response to a small change in its inputs.

2.5.1 **Loss Functions**

In the context of differentiable learning, a loss function is used to quantify the alignment between a machine learning model's predictions and the actual values. It measures the model's performance by expressing the difference between predicted and ground truth values quantitatively. The objective of differentiable learning is to optimize the model's weights based on the loss function. The choice of a loss function depends on the specific problem, data characteristics, and machine learning algorithm

employed. Commonly used loss functions include L1 and L2 losses, which are distance metrics penalizing deviations between model predictions and ground truth data at the pixel level.

The L1 loss, also known as Mean Absolute Error (MAE), computes the average absolute differences of data points. It can be defined formally as shown in Equation 3:

$$L1(I_{ref}, I_{ren}) = \frac{1}{N} \sum |I_{ref} - I_{ren}| \quad (3)$$

Here, N represents the number of data points, I_{ref} denotes the ground truth value, and I_{ren} represents the model prediction. The absolute value operator is used to account for negative differences, ensuring that cases where the prediction is greater than the ground truth value do not distort the average.

On the other hand, the L2 loss, also known as Mean Squared Error (MSE), addresses negative differences differently. Like MAE, it computes the average deviations, but instead squares the differences. The formula for MSE is shown in Equation 4:

$$L2(I_{ref}, I_{ren}) = \frac{1}{N} \sum (I_{ref} - I_{ren})^2 \quad (4)$$

Again, N represents the number of data points, I_{ref} represents the ground truth value, and I_{ren} denotes the model prediction. By squaring the differences and averaging them, the L2 loss not only handles negative differences but also becomes differentiable. However, the squaring operation introduces a quadratic relationship, where larger differences result in significantly higher errors. Consequently, the L2 loss heavily penalizes larger errors, potentially distorting the metric if there are outliers in the dataset. Conversely, small errors at the decimal scale tend to be neglected.

Beside from these fairly simple yet effective loss functions, there are many more alternatives used in the literature. Peak Signal-to-Noise Ratio (PSNR) is a traditional method for image quality assessment based on the error sensitivity of human vision.

It is most commonly used to measure the quality of the reconstructed image. The PSNR is defined as in Equation 5:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \quad (5)$$

While PSNR is simple and easy to compute, it's often criticized for not aligning well with the human perception of image quality, especially for cases where structural information is of primary importance.

To overcome the limitations of the PSNR and better mimic the human visual system, the Structural Similarity Index Measure (SSIM) was introduced. It considers changes in structural information, luminance, and contrast of the images, which are all important components of perceived image quality. The SSIM is defined as in Equation 6:

$$SSIM(I, K) = \frac{(2\mu_I\mu_K + C_1)(2\sigma_{IK} + C_2)}{(\mu_I^2 + \mu_K^2 + C_1)(\sigma_I^2 + \sigma_K^2 + C_2)} \quad (6)$$

In addition to these more complex loss functions, Silhouette loss, is a commonly used metric in 3D vision and graphics, particularly in the context of differentiable rendering. In these settings, it serves as a measure of quality for the rendering of 3D models, comparing the rendered silhouette of the model against a ground truth silhouette. The silhouette loss for a single sample is defined as in Equation 7:

$$SilhouetteLoss = \sum_{i=1}^M \sum_{j=1}^N |S_{\text{rendered}}(i, j) - S_{\text{truth}}(i, j)| \quad (7)$$

2.5.2 Optimization Algorithms

An optimization algorithm in differentiable learning seeks to find the best possible parameters for a model that will minimize the loss function. Gradient descent is one of the most popular optimization algorithms used in differentiable learning. It iteratively adjusts the parameters of the model to find the minimum of the loss function. The

algorithm leverages the gradient of the loss function with respect to the parameters to update the parameters. If the gradient is positive, the parameters are adjusted in the negative direction to decrease the loss function, and vice versa if the gradient is negative.

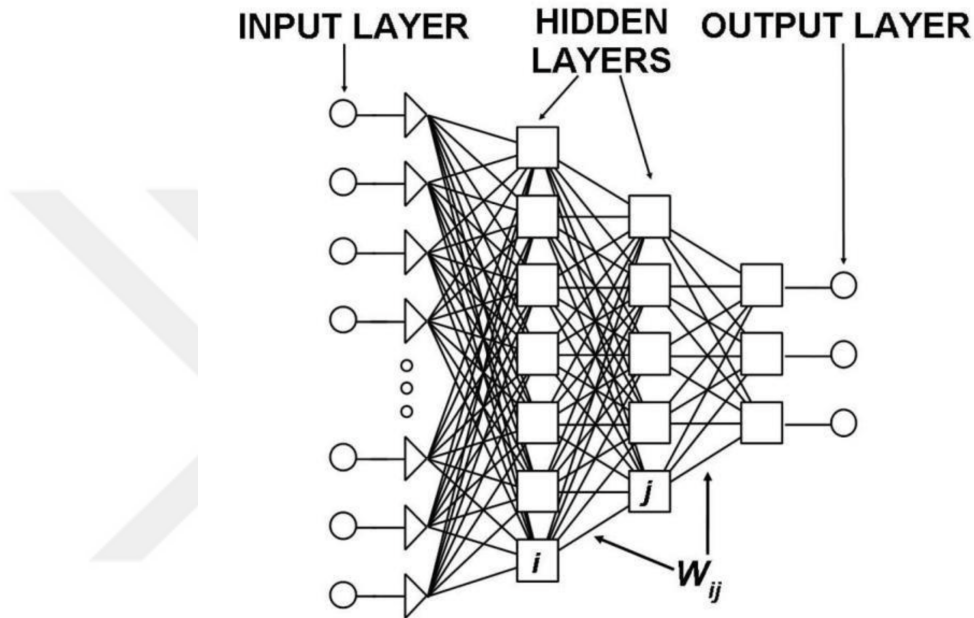


Figure 11: Visualization of a Multi-Layer Perceptron (MLP): The figure illustrates a typical MLP architecture with an input layer, multiple hidden layers, and an output layer. Data flows from left to right, starting from the input layer and moving through the hidden layers via weighted connections, before reaching the final output layer. Each node in the hidden layers applies a non-linear activation function to transform its inputs. Figure taken from [5].

2.5.3 Neural Networks

Neural networks form the backbone of differentiable learning. They are inspired by the biological neural networks in the human brain and consist of interconnected layers of nodes, or "neurons," that transmit and process data. Each connection between neurons is associated with a weight, and these weights are what the network learns during the training process. Neural networks can be simple, with only one or two layers (as in a perceptron), or they can be complex, with many layers (as in

deep learning). They can model complex, non-linear relationships and are especially powerful when dealing with high-dimensional data, such as images or text. Training a neural network involves using an optimization algorithm, such as gradient descent, to minimize a loss function and adjust the network’s weights to improve its predictions.

2.6 *Inverse Rendering*

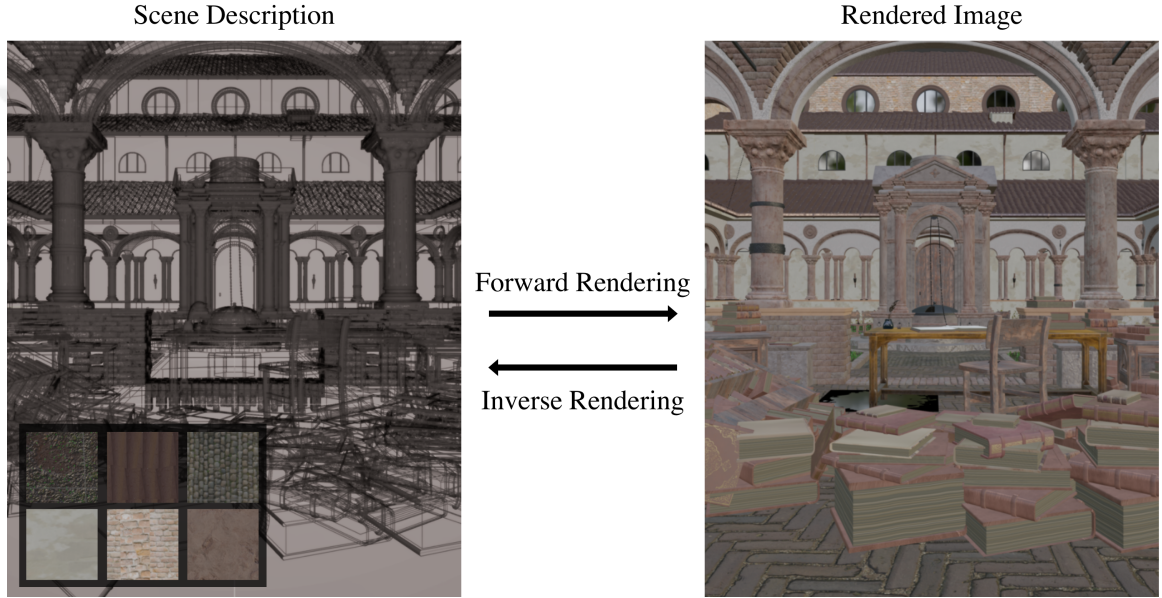


Figure 12: The scene description is illustrated on the left-hand side. On the right-hand side, the output of the forward rendering process is displayed. The inverse of this operation, known as inverse rendering, aims to generate the scene description from a given rendered image.

The inverse rendering is the reverse process of the rendering that derives unknown surface reflectance, lighting, and scene geometry from images of a 3D scene. Inverse rendering is a fundamental problem in computer graphics and computer vision and covers almost all research topics that derive the physical properties of a 3D scene from its images. Specifically, an image of a 3D scene can be determined by the geometry and layout of 3D objects in the scene, reflectance properties of the objects, as well as the lighting conditions of the scene [18].

2.7 Differentiable Rendering

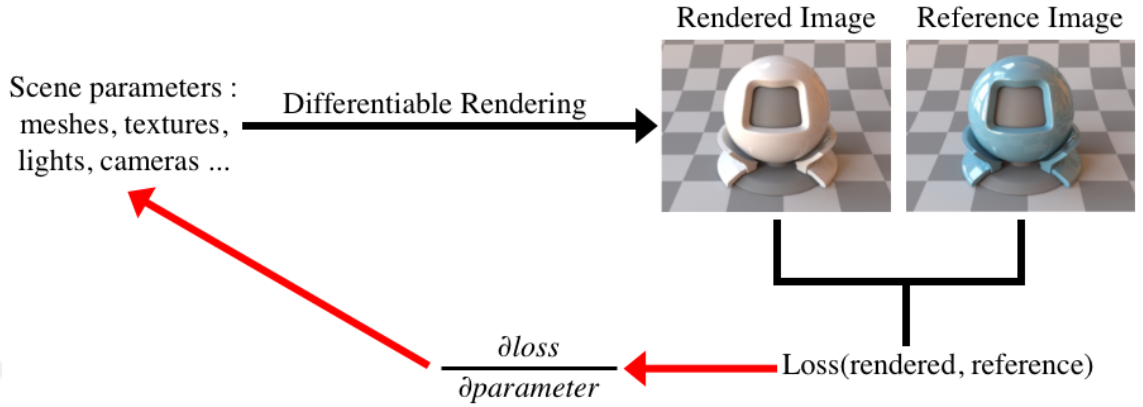


Figure 13: Visualization of a Differentiable Rendering Loop: The figure presents the continuous loop of the differentiable rendering process. Starting with a set of parameters, the scene is rendered, and the result is compared with a target image. The difference is then used to adjust the initial parameters through back-propagation, closing the loop and initiating a new rendering process. This iterative process continues until the rendered image closely matches the target.

Differentiable rendering (DR) is a method that aims to seamlessly integrate the rendering process into end-to-end optimization loops by obtaining gradients of the rendering process. DR enables the optimization of 3D objects by utilizing 2D projections. Optimization of 3D scene parameters can be achieved through backpropagation of gradients with respect to the rendering output. To incorporate the rendering layer into the predicted scene parameters, the loss is computed by an image-based loss function which compares the rendered and input images in various ways. A widely used optimization loop is shown at Figure 13.

2.8 Discontinuities in Differentiable Rendering

The objective of differentiable rendering techniques is to compute the partial derivatives of rendered image pixels with respect to the scene parameters. However, this is a challenging task since changes in visibility are inherently non-differentiable. There are several reparameterization methods which mitigate the non-differentiability of

the partial derivatives of rendered image pixels. To achieve this, Vicini *et al.* [1] proposed a modification of sphere tracing that dynamically constructs reparameterizations to enable accurate differentiable SDF rendering. Additionally, Bangaru *et al.* [2] proposed a family of area integrals that is useful for handling discontinuities in differentiable rendering through convolution. Finally, to avoid discontinuities in differentiable rendering, Loubet *et al.* [19] proposed a method that applies carefully chosen changes of variables to remove the dependence of discontinuities on scene parameters.

2.9 Novel View Synthesis

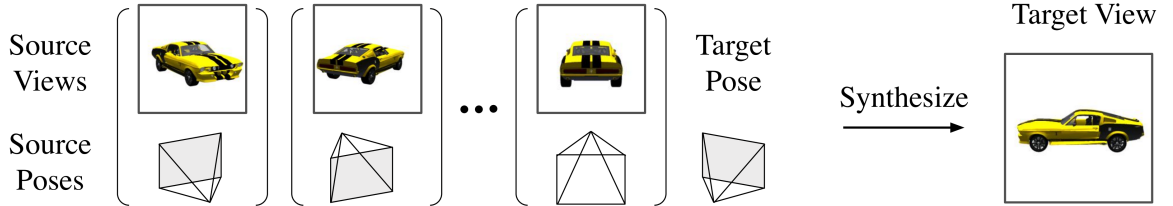


Figure 14: Visualization of the Novel View Synthesis Task: The figure illustrates the process of generating a target view from a given target pose, using the provided source views and source poses. Initially, several source views and their corresponding poses exist. Given a new target pose, the process synthesizes a novel view, referred to as the target view, representing the scene as it would appear from the perspective of the target pose. Figure taken from [6].

Novel view synthesis is a task within computer vision and graphics which aims to generate unseen perspectives from existing visual data. The procedure requires single or multiple images of a target object and their associated camera poses as input, denoting the precise position, orientation, and parameters of the camera at the time of image capture. The primary objective of this task is to generate a rendering of the target object as viewed from an unobserved viewpoint, employing an understanding of object geometry, texture, and the interaction of these elements with environmental variables such as light and shadow. Figure 14 illustrates the inputs and outputs of

the novel view synthesis task.

2.10 3D Scene Reconstruction

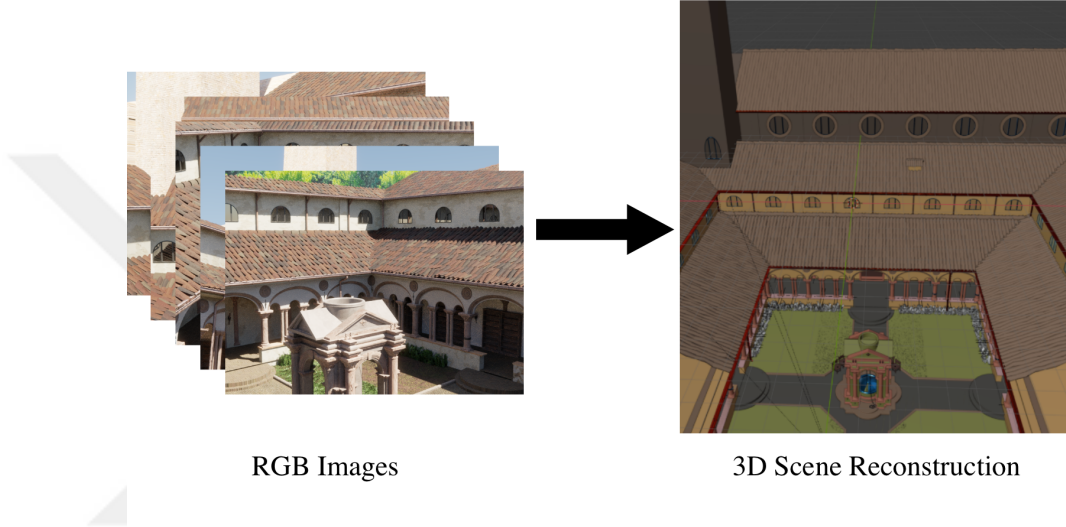


Figure 15: Visualization of the 3D Scene Reconstruction Task: The figure demonstrates the process of reconstructing a 3D scene from multiple input images. On the left, several input images are displayed, which serve as the basis for the reconstruction. On the right, the resulting image showcases the successfully reconstructed 3D scene, providing a comprehensive view of the spatial arrangement and depth of the scene’s elements.

3D scene reconstruction is a fundamental task within the field of computer vision and graphics. It aims to generate high-fidelity 3D scenes from 2D images, essentially deriving depth information from flat images to reconstruct the original scene in three dimensions. Within the computer vision literature, this task is often likened to the reversal of a traditional 2D image creation process, where 3D scenes are projected onto a 2D plane. Whereas in computer graphics, the task is also related to the inverse rendering problem. Figure 14 illustrates the 3D scene reconstruction task.

The essence of the problem lies in the complexity of inferring 3D spatial information from 2D images. Unlike 2D imaging, where a scene is directly captured onto a flat surface, 3D scene reconstruction aims to also estimate the depth and geometric characteristics of objects in the scene from 2D views, which presents considerable technical challenge. The fidelity of the 3D reconstruction can vary based on several factors, including the quality and quantity of the 2D input images, the accuracy of the depth estimation and the accuracy of the provided camera positions.

2.11 *Ambiguities in Interpreting 3D Shapes from a 2D Perspective*

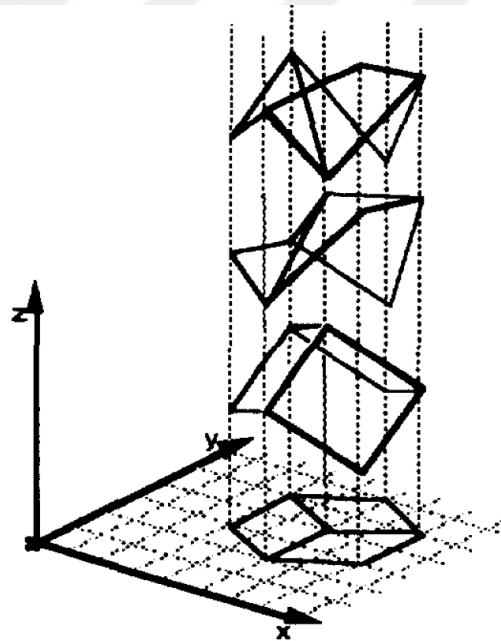


Figure 16: Interpreting 3D Shapes from a 2D Perspective: This figure illustrates the inherent ambiguity in interpreting 3D geometries from a 2D projection. Given a single 2D perspective, an infinite array of 3D geometries could potentially align with the observed projection. Figure sourced from [7].

Observing a 3D space from a 2D perspective without the aid of additional priors presents a significant interpretive challenge due to the infinite array of 3D geometries that can align with the given projection. This difficulty magnifies the complexity of 3D vision and graphics tasks, as it introduces numerous ambiguities. To counteract

this, we can incorporate additional priors that help resolving these ambiguities. One such method involves the use of advanced depth cameras capable of capturing not only the 2D projection of the scene, but also depth information. Additionally, refining the illumination settings of the scene can further assist in resolving the majority of ambiguities present within the 3D space. A visual representation of this phenomenon can be seen in Figure 16.



CHAPTER III

LITERATURE REVIEW

In the following section, we present a concise survey of the methods used for scene representation, differentiable rendering, 3D reconstruction and currently available datasets.

3.1 Representing 3D Geometry and Scene with NeRFs

With recent advancements in computer vision and graphics, alternative methods of representing scenes have emerged. The concept of representing scenes via a fully-connected deep network was first introduced by Mildenhall *et al.* [9]. Since then, several enhancements have been implemented. Martin *et al.* [10] presents an extension of NeRF that relaxes its strict consistency assumptions to resolve the severe ghosting, oversmoothing, and other artifacts that occur in the real-world cases where the input images are captured in different illumination settings and have occlusions caused by moving objects.

Although the original neural radiance field methods can accurately capture view-dependent effects such as reflection, refraction, and translucency, they require dense view sampling of the scene, which is not always feasible in real-world scenarios. Suhail *et al.* [20] proposed a four-dimensional representation of the light field to more effectively represent the effects of reflection, refraction, and translucency with sparser sampling of the scene.

Recent improvements in generative methods, when combined with state-of-the-art NeRF representations, can construct scenes from as few as a single image. Melas *et al.* [21] introduced an integration of a conventional image generator and NeRF, enabling the model to "dream up" novel views of an object, thus effectively increasing

the scene’s sampling rate. Despite their capacity to render realistic scenes, these generative-based approaches rely on prior data, and the estimations made within the scene are not guaranteed to reflect reality.

3.2 Representing 3D Geometry and Scene with SDFs

Surfaces represented by meshes can be rendered efficiently; however, their fixed topology limits their ability to represent arbitrary surfaces. Alternatively, volumetric representation is another commonly used method. Volumetric representation is capable of representing surfaces with arbitrary topology; however, compared to rendering triangular meshes, it is more challenging to efficiently apply surface rendering to surfaces represented with volumetric representation.

Another alternative to explicit surface representation methods is the utilization of implicit representation methods. Occupancy functions are commonly used in the literature to implicitly represent shapes and surfaces. Recently, Lamb *et al.* [22] proposed the utilization of occupancy functions to learn restorations for fractured shapes.

Similar to occupancy functions, signed distance functions (SDFs) are also employed for the implicit representation of surfaces. There has been a renewed interest in SDFs as a promising alternative to triangular meshes and volumetric representations for representing the geometry of shapes. Jiang *et al.* [23] and Liu *et al.* [24] have proposed the use of SDFs for scene reconstruction through differentiable rendering. More recently, Vicini *et al.* [1] presented a novel reparameterization method that enables efficient utilization of SDF-based surface reconstruction via differentiable rendering without the need for complex priors such as depth information.

3.3 Scene Optimization via Differentiable Rendering

There have been multiple methods proposed to achieve differentiable rendering. Loper *et al.* [25] introduced DIB-R, a differentiable rendering framework that allows for the

computation of analytical gradients for every pixel, overcoming the non-differentiability of traditional rendering pipelines.

Kato *et al.* [26] addressed the challenge of integrating polygon mesh rendering into neural networks for 3D modeling of 2D images. The authors proposed an approximate gradient for rasterization, enhancing 3D mesh reconstruction and enabling novel applications like 2D-to-3D style transfer. Their method enabled the potential of incorporating a mesh renderer into neural networks.

Liu *et al.* [27] proposed a novel differentiable rendering framework that converts the non-differentiable rasterization step in traditional graphics renderers into a differentiable form. By treating rendering as an aggregation function, it directly renders colorized mesh and back-propagates supervision signals to mesh vertices from various image representations, including silhouette, shading and color images. Notably, it also conveys gradients to occluded and distant vertices. This approach enhances 3D reconstruction and image-based shape fitting tasks.

Laine *et al.* [28] proposed a high-performance, modular differentiable renderer design that leverages existing graphics pipelines. The proposed design is applicable within automatic differentiation frameworks and efficiently solves inverse rendering problems while maintaining high geometric accuracy.

Additionally, a differentiable version of ray-tracing-based rendering has been recently proposed by Li *et al.* [29]. The proposed solution introduces a novel differentiable ray tracer, capable of computing derivatives of scalar functions over a rendered image with respect to various scene parameters. Utilizing a unique edge sampling algorithm and efficient importance sampling methods based on spatial hierarchies, the method can handle non-differentiable visibility terms, generating gradients efficiently despite scene complexity and desired precision.

3.4 3D Reconstruction

3D Reconstruction, within the realms of computer vision and graphics, is the process of generating a 3D depiction or model of an object or setting from 2D images or alternate data streams such as video. The objective of 3D reconstruction is to capture a digital replica of a particular object or setting. Traditional approaches, such as COLMAP [30], rely on reconstructing the 3D structure from a sequence of images captured from various perspectives, often derived from a video source. Notably, these conventional methods can achieve satisfactory reconstruction results. However, in recent times, with the advancement of neural network techniques, Neural 3D reconstruction approaches have emerged as state-of-the-art solutions, delivering superior performance while maintaining manageable computational costs.

Zhang *et al.* [31] introduced Neural Reflectance Surfaces (NeRS), a novel approach for representing real-world scenes, outperforming traditional volumetric models like Neural Radiance Fields (NeRF). NeRS, tested on multi-view images from online marketplaces, provides water-tight reconstructions of surfaces and effectively learns their reflectance properties, demonstrating potential for creating scalable, high-quality libraries of real-world shapes and materials.

Recently, Li *et al.* [32] presented a physically-based network for reconstructing the 3D shape of transparent objects using a limited number of natural images. The approach introduces novel methods for modeling complex light transport, refining transparent shapes, and reconstructing 3D point clouds.

More Recently, Sun *et al.* [33] proposed a novel method for surface reconstruction from internet photo collections under varying illumination, using a hybrid voxel and surface guided sampling technique. The authors also present a novel benchmark for evaluation, demonstrating the performance of existing classical and neural reconstruction methods.

3.4.1 Multi-View 3D Reconstruction

Wang *et al.* [34] introduced a unified feature extraction and view fusion in a single transformer network that enables the exploration of view-to-view relationships using self-attention among multiple unordered inputs to better capture the underlying scene.

Recently, Hu *et al.* [35] proposed a NeRF-based method that leverages depth information to regularize both multi-view and single-view 3D consistency among pixels. Tong *et al.* [36] present a novel method for recovering the 3D geometry of objects within transparent enclosures by modeling the scene as two distinct sub-spaces and developing a hybrid rendering strategy.

Additionally, Xiao *et al.* [37] proposed a neural incremental structure-from-motion approach. This method estimates the camera poses and scene geometry from a set of uncalibrated images by learning coordinate multi-layer perceptrons for the implicit surfaces and the radiance fields from established keypoint correspondences. Not only does this approach enable camera pose estimation and scene geometry reconstruction without known camera extrinsics, but it also presents a promising way for neural implicit rendering without requiring prior knowledge of camera extrinsics.

3.4.2 Single-View 3D Reconstruction

3D reconstruction from a single-RGB image in unconstrained real-world scenarios presents numerous challenges due to the inherent diversity and complexity of objects and environments. Thus this problem is still an ill-posed problem today without prior knowledge.

Deng *et al.* [38] have proposed a method that leverages off-the-shelf vision-language models and introduces a dual-component language guidance as conditioning inputs to the diffusion model. This approach proves particularly beneficial for enhancing multi-view content coherence, as it refines the general image prior conditioned on

the semantic and visual features of the single-view input image. Additionally, they introduce a geometric loss based on estimated depth maps to regulate the underlying 3D geometry of the utilized NeRF.

Furthermore, Shen *et al.* [39] proposed a comprehensive framework that integrates visual-language models with object segmentation to transition objects from 2D to 3D, leading to a robust system for single-view reconstruction tasks. The proposed framework initially employs language-image pretraining to estimate the textual description of the image. Subsequently, a segment-anything model identifies the object’s region of interest. The segmented object and its description are then utilized to execute the 3D reconstruction task.

3.4.3 Differentiable Rendering Based 3D Reconstruction

Recent advancements in differentiable rendering techniques has led to the development of differentiable rendering based methods for 3D reconstruction of objects and scenes.

Bangaru *et al.* [2] introduced an efficient, unbiased differentiable rendering algorithm, circumventing the need for explicit boundary sampling. They converted the boundary integral into an area integral using the divergence theorem, which is then suitable for Monte Carlo rendering. Their method integrates seamlessly into traditional path tracers without requiring specialized data structures for boundary sampling.

Recently, Vicini *et al.* [1] proposed an enhanced sphere tracing algorithm for differentiable rendering of SDFs, enabling precise computation of shape parameter derivatives. The technique efficiently addresses challenges of shape inversion and outperforms previous methods by reconstructing synthetic objects using per-pixel RGB loss, eliminating the need for complex regularization or priors.

Similar to the methods based on NeRFs, these techniques also work on a per-scene

basis. This means they need continuous adjustment or re-optimization of the underlying neural network or scene representation structure. Unlike generative methods, these techniques do not create non-existent features, as they do not estimate but derive the scene parameters from the input images and are more explainable compared to neural network-based methods.

3.5 3D Reconstruction Using SDFs

Combining gradient descent, differentiable rendering, and SDFs, it has been shown that scene reconstruction is achievable through optimization of scene parameters.

Dong *et al.* [40] proposed a novel 3D spatial representation, the Probabilistic Signed Distance Function (PSDF), for data fusion and scene reconstruction. The PSDF utilizes a hybrid data structure combining voxel and mesh and allows for the real-time extraction of high-quality surfaces.

Park *et al.* [41] proposed a learned continuous Signed SDF for high-quality 3D shape representation and completion. It outperforms traditional methods by representing an entire class of shapes and significantly reducing model size, achieving state-of-the-art performance even with partial or noisy data.

Zhang *et al.* [42] presented a novel neural surface reconstruction framework that employs stereo matching and feature consistency to optimize implicit surface representation. Using a signed distance field for scene geometry and a surface light field for appearance, the method shows improved geometry estimation and complex topology reconstruction.

Recently, Vicini *et al.* [1] used SDFs to represent the geometry and texture of the objects to achieve scene parameter optimization via gradient descent using per-pixel RGB loss.

The use of SDFs for representing scenes offers significant advantages, particularly

in terms of accommodating topological variations that may arise during scene optimization. Additionally, another feature that enhances the utility of this method is its compatibility with surface rendering techniques.

3.6 Datasets

Table 1: Overview of available datasets.

Dataset Name	Scene Count	Scene Type	Includes Camera Location	Includes Light Location	Is Synchronised	Controlled Environment
DTU MVS [43]	80	Real	Yes	No	No	Yes
BlendedMVS [44]	113	Virtual	Yes	No	Yes	Yes
CO3D [8]	19000	Real	Yes	No	No	No
Ours	10	Real	Yes	Yes	Yes	Yes

In the literature, various datasets can be found that are useful for the task of 3D reconstruction [8, 44, 43]. Jensen *et al.* [43] presents a dataset known as the DTU multi-view stereopsis (MVS) dataset, which was created using a carefully designed controlled environment featuring a stereo camera mounted on a 6 degrees of freedom (DoF) robotic arm. This setup facilitates the collection of stereo images of the target objects, along with corresponding camera location information.

Another dataset that has recently been made available is BlendedMVS [44]. This dataset is generated using a pipeline that utilizes 3D reconstruction of a densely captured real-world scene, followed by the generation of rendered counterparts of the input images using the reconstructed scene. Finally, the rendered and input images are blended to enhance the global illumination of the final images.

The Common Objects in 3D dataset [8] is another recently released dataset that contains real multi-view images of target objects, as well as camera location and 3D point cloud data.

Table 1 provides a summary of the main features and characteristics of the mentioned datasets. Although each dataset has its advantages, our specific use case requires light location and direction information in addition to camera parameters

to enable the virtual replication of real scenes. To collect the necessary data, we have constructed a controlled environment, which will be discussed in detail in the following sections.

3.7 Applications of 3D Reconstruction

In the realm of computer vision and graphics, 3D reconstruction has many applications in a range of topics, from object detection, autonomous driving, and augmented reality to virtual reality. Ma *et al.* [45] proposed a monocular 3D object detection framework for autonomous driving, which solves this problem in the reconstructed 3D space in order to exploit 3D contexts explicitly. Tian *et al.* [46] proposed a novel occlusion handling method based on 3D reconstruction for augmented reality applications. Sing *et al.* [47] proposed a mixed reality experience that combines both virtual reality and augmented reality, and lets players transform their environment into a virtual environment by utilizing stereoscopic rendering and 3D reconstruction.

CHAPTER IV

METHODOLOGY

In the following sections, we provide a comprehensive overview of our data acquisition process for constructing real-world scenes and their virtual counterparts. We then offer a detailed exposition of the essential data and information of our benchmark scenes. Subsequently, we delve into the details of our baseline 3D reconstruction methods and the modifications we made to the loss calculation to improve the reconstruction in real-world scenes.

4.1 Creating Virtual Scenes

Accurately capturing 3D models from the real world requires collecting data that is both precise and representative of the target object. However, given the complexity of the real world, it is not always feasible to model every physical phenomenon, such as the location of cameras and light sources, using common RGB cameras and tools. To better understand and address the current limitations, we apply existing 3D reconstruction methods [1, 2] to scenes captured via a single turntable video using a smartphone camera, which we further discuss in the following sections. Despite the ease of application, this method is not able to capture the light source and direction information used in the inverse rendering process of state-of-the-art methods [1, 2]. In order to capture the required light information alongside scene and camera data, we have constructed a controlled environment with multiple cameras and light sources, where we have knowledge of all their respective parameters.

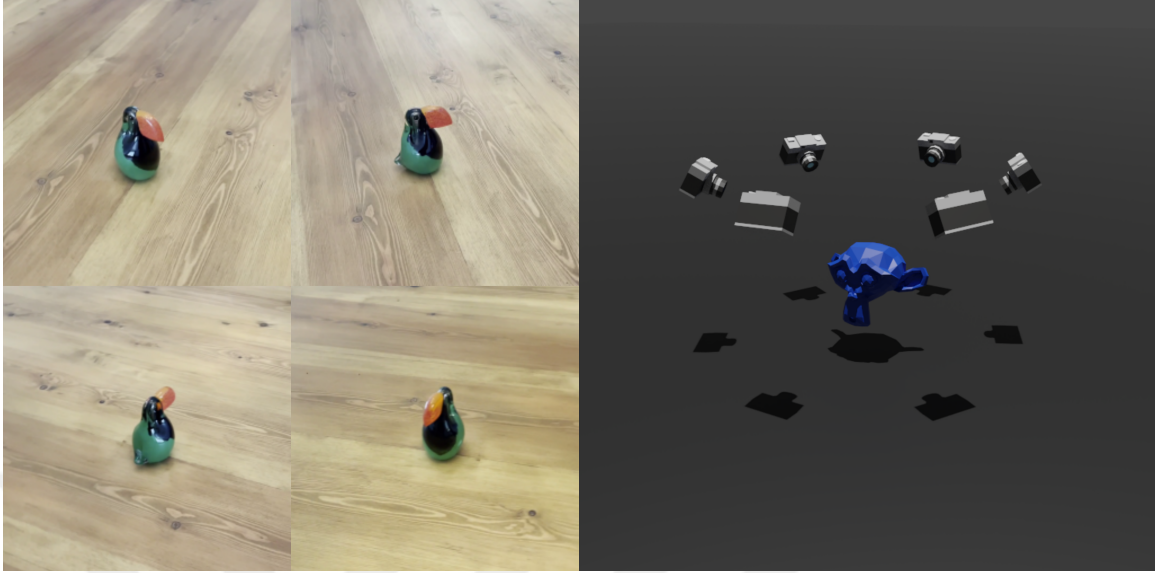


Figure 17: Illustration of the data collection process using the methodology proposed by Reizenstein *et al.* [8]. Images of the target object from several viewpoints are shown on the left side. On the right side, an illustration of the data collection process is shown, where the blue object represents the target object, and the camera models represent the trajectory of the real camera while capturing a video of the target object.

4.1.1 Collecting Scene Parameters via Video

To facilitate the reconstruction of real-world scenes from a single turntable video, similar to the CO3D dataset [8], we generated scenes using the following methodology: (1) Capturing a turntable video of a rigid object, (2) Estimating extrinsic and intrinsic camera parameters with the help of COLMAP [30], (3) Scaling and translating the scene, (4) Removing the background, and (5) Merging and exporting the selected background-free frames, along with their corresponding camera parameters, in Blender 3D [48]. After the creation of the virtual scene, each frame of the captured video becomes effectively a novel viewpoint inside the virtual scene. Depending on the need, a desired number of frames can be selected to be used as input for the 3D reconstruction method. Since our scope is to reconstruct objects where a minimal number of viewpoints are available, we have selected 6 viewpoints that are equally spaced apart from each other in terms of the number of frames. Figure 17 provides a

visualization of this process. Furthermore, the precision of the resulting scene significantly affects the final output, since any errors that occur in this stage are directly accumulated in the optimization phase of the used 3D reconstruction method. In this case, the precision of the resulting scene is bound by COLMAP [30], which is used to estimate the camera parameters.

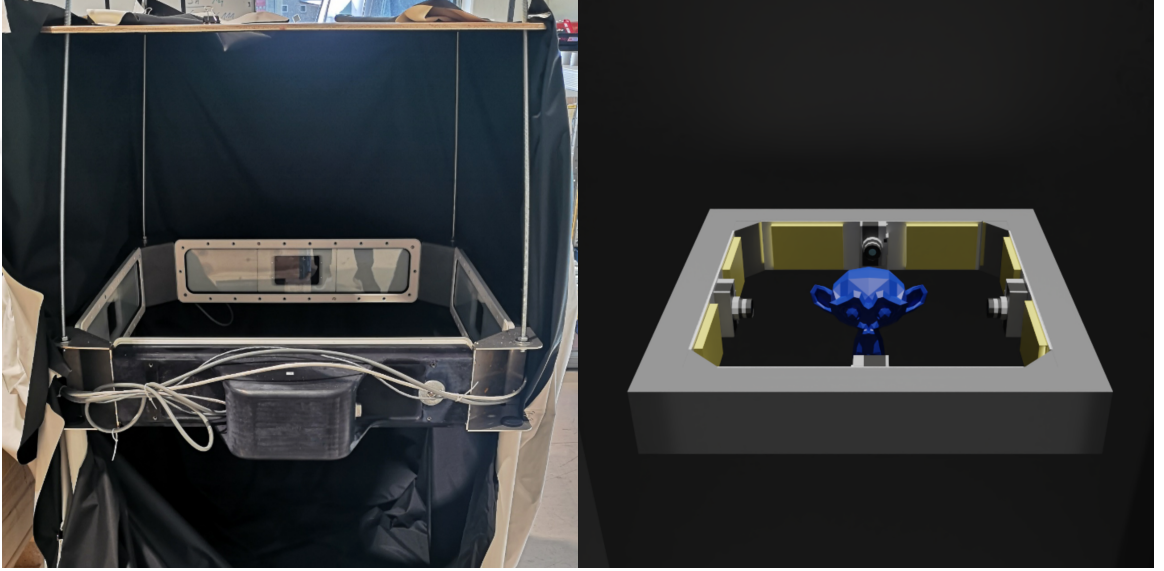


Figure 18: Our multi-camera setup which is used to collect data for our benchmark scenes. The left side shows the setup in the real world, while the right side displays the virtual counterpart of the test setup. In the virtual scene, the camera models represent virtual cameras, while the yellow boxes represent diffused LED bars, and the blue object represents the target object for 3D reconstruction.

4.1.2 Collecting Scene Parameters via Multiple Cameras

To capture the required light information alongside scene and camera data, we have constructed a controlled environment consisting of 4 cameras and 8 LED light sources, the location and direction of each instance of which we know. For the process of creating this environment, we use the following methodology: (1) We custom designed, manufactured, and assembled the housing for the electronic hardware. (2) We isolated the entire environment from outside light sources using an opaque, non-reflective material. (3) Then, we built a digital replica of the real environment in Blender3D [48]

by referring to the CAD model used for the production of the electronic housing. (4) Finally, we placed the target object inside our real-world scene to capture the necessary reference images. A visualization of our controlled environment is presented in Figure 18.

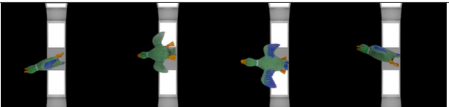
For reproducibility purposes, we provide the details of our setup. All four cameras have a resolution of 1920 by 1200 pixels, a sensor size of 11.3 by 7.1 mm, and a focal length of 6 mm. Synchronization between cameras is achieved by utilizing their gigabit Ethernet connection to ensure that the captured images are taken at the same instance. All LED lights are calibrated to emit 450 lumina of light with the diffuser panels mounted on. The housing for the cameras and other electronics is manufactured using off-the-shelf parts with a combination of custom CNC machined parts. The housing is a square frame where the distance from one inner wall to the opposing inner wall is 800 mm. The entire setup is isolated from outside light sources by using an opaque, black-colored material.

While replicating the real-world setup virtually, we utilized the precise camera parameters, light settings, and enclosure dimensions from our CAD model for the corresponding fields in Blender3D [48]. Consequently, any potential discrepancies between the actual scene and the virtual scene are within the production tolerances, which is ± 10 mm.

4.2 *Benchmark Scenes*

To effectively compare different 3D reconstruction methods, the use of benchmark scenes is imperative. As mentioned in Chapter 3, standard objects, scenes, and datasets available in the literature are suitable for serving as benchmark scenes for 3D reconstruction. The available real-world datasets provide scenes with known camera parameters [43, 44, 8] and synchronized images [44] containing a wide variety of objects [8] in a controlled environment [43, 44]. Furthermore, by utilizing the currently

Table 2: Details of the benchmark scenes. For each scene, we used four synchronized cameras and eight LED light sources with their respective intrinsic and extrinsic parameters. The original images were captured in the real-world scene, while the reference images were obtained by transferring the object segment onto a render of the empty virtual-scene from the corresponding virtual-camera.

Scene Name	Original Images	Reference Images
The Duck		
Bottle Black		
Box		
Box Wooden		
Box Big		
Cup White		
Cup White 2		
Cup Gray		
Bottle Black Shifted		
The Duck Shifted		

available datasets, the current NeRF-based methods [10] can reconstruct target scenes with variable lighting in high fidelity. However, while being resilient to appearance variations and transient occluders, these methods require densely collected images from a high number of viewpoints, which may not be available in certain scenarios.

On the contrary, state-of-the-art SDF-based methods [2, 1] can reconstruct scenes with a limited number of views, but they require precise camera and light source parameters. Since the current datasets do not include light location and direction alongside camera parameters, current results for these methods are only presented using synthetic scenes.

To address this problem, we have prepared several scenes using the setup described in Section 4.1.2. As target objects, we have selected representative samples to cover convex and concave cases. Placing the target object inside the camera rig, we captured synchronized images using our cameras, which we refer to as original images. In order to utilize the placed object as a reference image for scene optimization, it is necessary to transfer it from the physical setup to our virtual scene. To achieve this, we transferred the segments containing the target object from each original image to their corresponding renderings within the virtual scenes. The resulting images are referred to as reference images within the scope of our study. Detailed information about the benchmark scenes is presented in Table 2.

4.3 Baseline 3D Reconstruction Methods for Real-World Objects

We have selected two state-of-the-art 3D reconstruction methods to present baseline results for our benchmark scenes. Our first method is primarily based on the work proposed by Vicini *et al.* [1], which was originally developed for reconstructing objects within virtual environments. This method focuses on the inversion of shape parameters. However, the discontinuous nature of visibility introduces significant bias in computed derivatives, and traditional shape representations like triangle meshes face additional difficulties. To overcome these challenges, SDFs are used to represent shapes, and their zero level set is gradually adapted during optimization. Additionally, this method employs a reparameterization technique that allows for accurate computation of shape parameter derivatives. This approach has the capability to

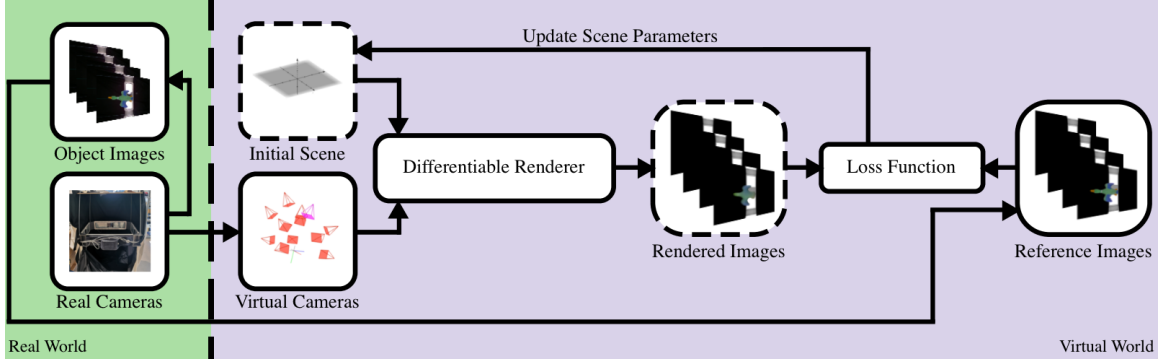


Figure 19: Overview of our pipeline for real-world 3D reconstruction. Boxes with dashed lines represent updated elements throughout the optimization loop. We capture target object images using our real-world environment, and use them to generate reference images. Virtual initial scenes and cameras are constructed to match their real-world counterparts. After initializing the virtual cameras and scene, scene parameters are updated based on the difference between the rendered and reference images.

reconstruct virtual objects solely by utilizing image-based loss functions.

Our second method is primarily based on the work proposed by Bangaru *et al.* [2]. Similar to the approach proposed by Vicini *et al.* [1], this method is also demonstrated only in virtual environments. They propose a novel method for differentiable rendering using area sampling. By deriving the area integral from first principles, they demonstrate an efficient and consistent estimator for the derivative, as well as a version that produces an unbiased estimate. Their method does not require dedicated data structures for selecting silhouette edges and can be easily integrated into a traditional unidirectional path tracer by tracing auxiliary rays. Unlike other approaches [19], their method does not introduce approximations to the integral and produces consistent and unbiased estimates. Thus, it enables the optimization of complex scene parameters to reconstruct the 3D shape and texture of the target object. Thus, similar to the first method, this approach also possesses the capability to reconstruct virtual objects using image-based loss functions.

In order to apply the mentioned methods to real world objects, it is necessary to

employ our scene creation method outlined in the Section 4.1.2. After creating a virtual representation of the real-world environment, the virtual scene can be optimized directly to obtain the 3D reconstruction of the target object within the real-world scene. An overview of our 3D reconstruction methodology is presented in Figure 19.

To calculate the loss between rendered and reference images, we have utilized multiple loss functions. Similar to Vicini *et al.* [1], our initial approach for determining the loss value is the direct calculation of the L1 loss between the reference and rendered images, as demonstrated in Equation 8.

$$L_{L1}(I_{ref}, I_{ren}) = \sum |I_{ref} - I_{ren}| \quad (8)$$

Additionally, we have employed the multi-scale L1 loss function, which involves averaging the L1 losses among decimated reference and rendered images. The formula for this loss function can be found in Equation 9.

$$L_{multi-L1}(I_{ref}, I_{ren}) = \sum_{i=0}^N (L1(I_{i-decimated-ref}, I_{i-decimated-ren})) / N \quad (9)$$

To further optimize performance, we have integrated the multi-scale L1 loss function with structural losses, specifically, structural similarity loss (SSIM) [49] and silhouette loss. Our multi-scale L1 + SSIM loss can be found in Equation 10, while our multi-scale L1 + silhouette loss is defined in Equation 11. This combination effectively amplifies the influence of structural details on the loss value.

$$L_{multi-L1+ssim}(I_{ref}, I_{ren}) = L_{multi-L1}(I_{ref}, I_{ren}) + SSIM(I_{ref}, I_{ren}) \quad (10)$$

$$L_{silhouette}(I_{ref}, I_{ren}) = L_{multi-L1}(I_{ref}, I_{ren}) + L_{L1}(X_{ref-sil}, Y_{ren-sil}) \quad (11)$$

CHAPTER V

RESULTS AND DISCUSSION

To provide baseline results for our benchmark scenes, we conducted a series of experiments using the aforementioned 3D reconstruction methods. Within these experiments, we utilized several loss functions to calculate the difference between the renderings of the current state of the scene and the reference images. We quantitatively measured the effectiveness of each method using the metrics L1, L2, Peak Signal to Noise Ratio (PSNR), Structural Similarity (SSIM), Fréchet Inception Distance (FID) [50] and Learned Perceptual Image Patch Similarity (LPIPS) [51] enabling an assessment of the reconstruction quality from different camera viewpoints. Furthermore, we conducted an ablation study to examine the effects of the employed loss functions, the number of viewpoints, and correct illumination on the performance of 3D reconstruction.

5.1 Experiments and Results on Single Turntable Video

To validate the effectiveness of our 3D reconstruction methods, we generated two real-world scenes following the methodology presented by Reizenstein *et al.* [8]. The first scene, referred to as "the toucan," features a glass, toucan-like object as the target, while the second scene, referred to as "the rubber duck" contains a red rubber duck as the target object. Both scenes share an identical surrounding environment, with only the target object being different. Due to the capture technique, virtual versions of these scenes lack the surrounding objects or the light location and direction, which are essential for both of the aforementioned 3D reconstruction methods.

We employed the Multi-scale L1 loss as the loss function to compute the error between the reference image and the rendered image of the specific state within the

virtual scene. For both scenes, we used 6 viewpoints for the optimization process. We optimized the virtual scenes for 512 epochs. The details of the parameters we used for the experiments conducted with the toucan and the rubber duck scenes are provided in Table 3. Figure 20 illustrates the evolution of the virtual scenes throughout the optimization process.

Table 3: Experiment Parameters

Scene Name	Number of Viewpoints	Learning Rate	Number of Iterations	SPP	SDF Resolution	Render Resolution
The Toucan	6	4e-2	512	128	128	256
The Rubber Duck	6	4e-2	512	128	128	256
Benchmark Scenes	4	4e-2	512	128	128	256

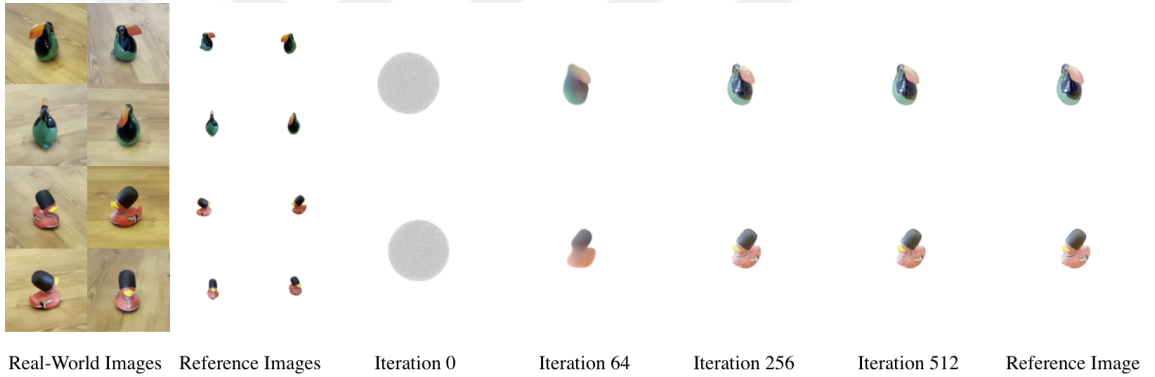


Figure 20: Single video results obtained using a turntable video of the target object. The first row presents the results generated from the toucan scene, while the second row displays the results obtained from the rubber duck scene. The optimization is conducted for 512 iterations with the multi-scale L1 loss. The illumination in the scene is assumed to be constant at every point.

As depicted in Figure 20, the mentioned methods are capable of performing 3D reconstruction without relying on light information or the surroundings of the scene. However, since the shading and luminance parameters of the scene are unknown, the resulting 3D reconstruction lacks the level of detail and visual quality presented in [2] and [1].

Table 4: Quantitative results for the toucan and the rubber duck scenes.

Experiment Name	L1	L2	SSIM	PSNR	LPIPS	FID
The Toucan	0.0012	0.0001	0.9947	38.5752	0.0096	0.1101
The Rubber Duck	0.0014	0.0002	0.9936	37.7716	0.0172	0.1161

5.2 *Experiment Results on Benchmark Scenes*

In order to demonstrate the 3D reconstruction performance of the aforementioned methods while integrating surrounding objects and illumination data from the real-world scene into the virtual scene, we conducted a series of experiments using our benchmark scenes. During the scene optimization process, we employed several loss functions to compute the error between the reference images and the rendered images of the virtual scene. For all benchmark scenes, we utilized four viewpoints of the target object for optimization. We optimized the virtual scene for 512 epochs in all benchmark scenes. The details of the parameters we used for the experiments conducted with benchmark scenes are provided in Table 3. Figure 21 showcases the evolution of the virtual scenes throughout the optimization process, as well as the turntable viewpoints of the 3D reconstructed objects. Turntable videos of these objects are also available as supplementary material. In the appendix, we present additional results, which can be seen in Figures 27, 28, and 29.

Furthermore, we collected the differences between the current state of the rendered images and the reference images using several metrics to quantitatively measure the quality of the 3D reconstruction. Regardless of the employed loss function in the scene optimization process, we collected the same metrics for each experiment. We have compiled the average and standard deviation from all benchmark scenes. These results, gathered after the final iteration of the optimization loop, are presented in Table 5. Additionally, we present the raw data for each experiment in Table 6.

Our findings reveal that the presented quantitative results are internally consistent within each scene, thereby demonstrating the reliability of our experimental setup.

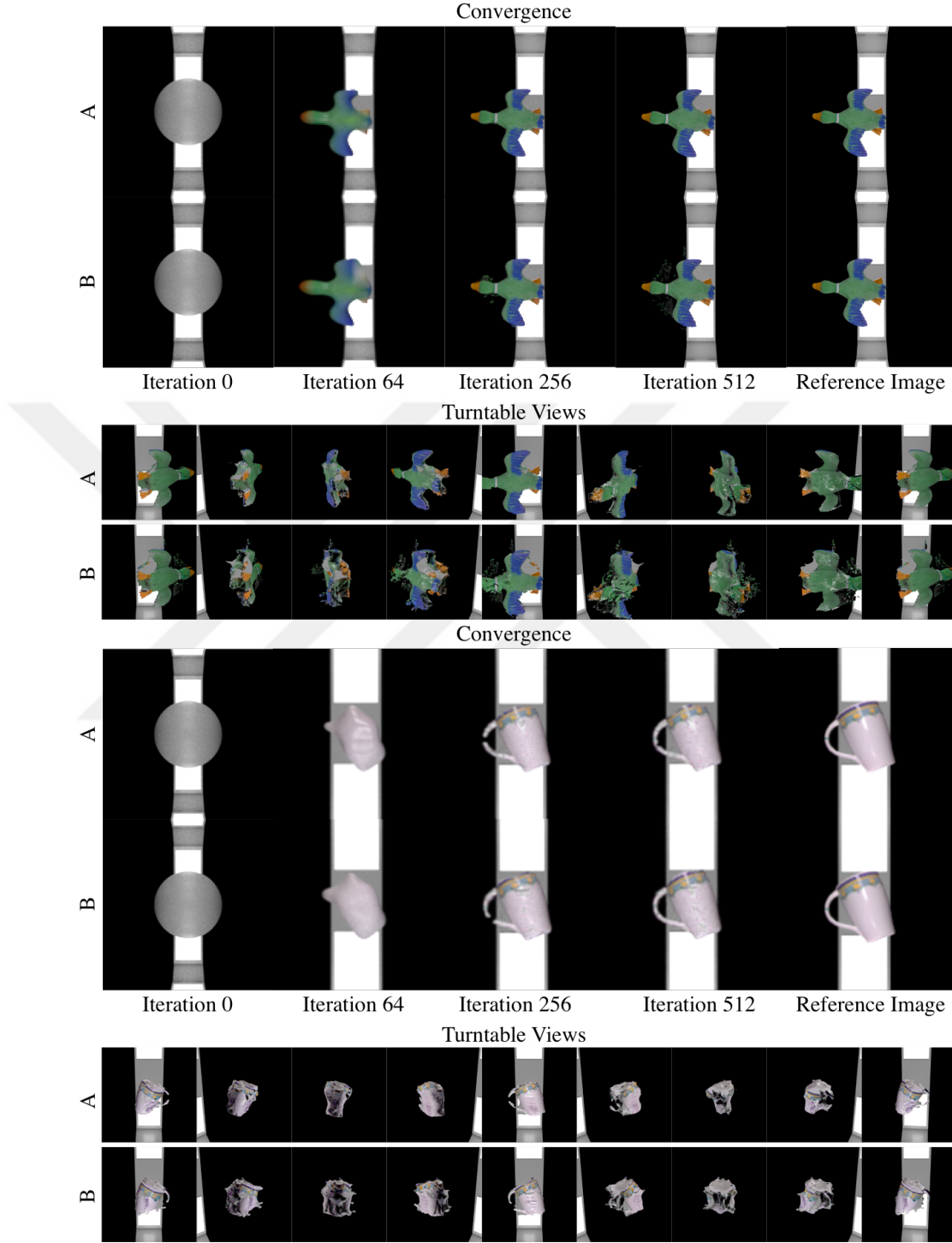


Figure 21: Multi-camera setup results obtained using four synchronized images of the target object. The rows marked with A presents the results obtained using proposed method by Vicini *et al.* [1]. The rows marked with B presents the results obtained using proposed method by Bangaru *et al.* [2].

Table 5: Quantitative results of the experiments conducted using our benchmark scenes. For each method and metric, we report the average and standard deviation across all benchmark scenes. The experiment names follow the format: "loss function used in optimization - reparameterization method". In each column, the best results are denoted in bold typeface and highlighted in **green**, while the second-best results are highlighted in **yellow**.

Loss Function - Reparameterization Method	L1	L2	SSIM	PSNR	LPIPS	FID	L1	L2	SSIM	PSNR	LPIPS	FID
	Averages						Standard Deviations					
L1 - Vicini <i>et al.</i> [1]	0.0048	0.0011	0.9724	31.9009	0.0492	0.0642	0.0032	0.0015	0.0262	3.6782	0.0482	0.1068
L2 - Vicini <i>et al.</i> [1]	0.0056	0.0012	0.9671	30.9960	0.0554	0.0619	0.0030	0.0015	0.0253	3.5049	0.0416	0.1055
Multi-Scale L1 - Vicini <i>et al.</i> [1]	0.0048	0.0010	0.9725	31.9798	0.0488	0.0560	0.0032	0.0016	0.0257	3.6639	0.0463	0.1083
Multi-Scale L1 + SSIM - Vicini <i>et al.</i> [1]	0.0049	0.0011	0.9719	31.9299	0.0547	0.0535	0.0031	0.0016	0.0249	3.5767	0.0470	0.1087
Multi-Scale L1 + Silhouette - Vicini <i>et al.</i> [1]	0.0054	0.0011	0.9695	31.2069	0.0500	0.0684	0.0035	0.0015	0.0274	3.7065	0.0481	0.1571
L1 - Bangaru <i>et al.</i> [2]	0.0049	0.0010	0.9690	31.8029	0.0591	0.0665	0.0030	0.0015	0.0264	3.4565	0.0514	0.1141
L2 - Bangaru <i>et al.</i> [2]	0.0058	0.0012	0.9625	30.8708	0.0663	0.0602	0.0028	0.0015	0.0284	3.3814	0.0473	0.1087
Multi-Scale L1 - Bangaru <i>et al.</i> [2]	0.0051	0.0011	0.9679	31.6376	0.0603	0.0470	0.0032	0.0015	0.0279	3.5172	0.0515	0.1062
Multi-Scale L1 + SSIM - Bangaru <i>et al.</i> [2]	0.0052	0.0011	0.9680	31.6075	0.0605	0.0221	0.0035	0.0015	0.0277	3.5038	0.0524	0.0335
Multi-Scale L1 + Silhouette - Bangaru <i>et al.</i> [2]	0.0051	0.0010	0.9693	31.6437	0.0585	0.0712	0.0034	0.0015	0.0262	3.5005	0.0498	0.1616

Interestingly, we observed no significant quantitative improvement when adding a structural term to the loss function. This observation opens up a discussion on the evaluation of 3D shape quality from a 2D perspective. However, comparing the two methods, our quantitative analysis demonstrated that the reconstruction outcomes achieved using Vicini *et al.*'s [1] method consistently outperformed those obtained with Bangaru *et al.*'s [2] method.

Upon analyzing the runtime per scene for both methodologies, we observe that Vicini *et al.*'s [1] method exhibits an average runtime of ≈ 2 hours, whereas Bangaru *et al.*'s [2] method requires ≈ 4 hours. Evidently, Vicini *et al.*'s [1] approach surpasses the efficiency of Bangaru *et al.*'s [2] method by a significant margin.

We conducted our experiments using multiple Nvidia RTX Quadro 6000 GPUs, with each experiment running on a single GPU. In total, we spent ≈ 312 GPU hours to complete all the experiments we present.

5.3 Effects of Different Loss Functions

Although our quantitative analysis presented in Table 5 does not indicate any significant improvement with current metrics, a closer examination of the qualitative results reveals that the fine details of the target object are captured slightly better.

Figure 22 illustrates the impact of different loss functions on the performance of

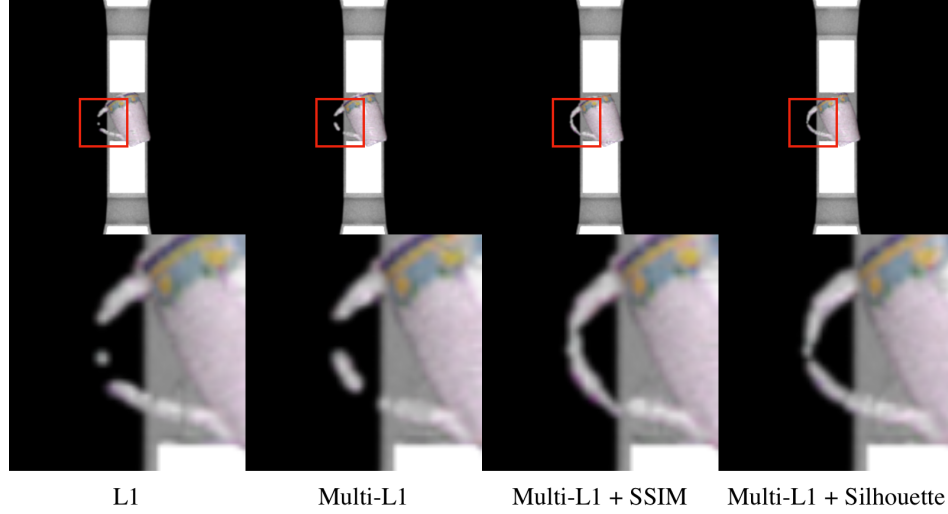


Figure 22: The first row shows the 3D reconstruction results using L1, Multi-scaled L1, Multi-scaled L1 + SSIM, and Multi-scaled L1 + Silhouette loss functions, respectively. The second row indicates the magnification of the section inside the red box in the same column.

3D reconstruction. As depicted, when using L1 and Multi-Scale L1 loss functions, the handle of the cup is partially reconstructed. However, the addition of a structural term, specifically SSIM or silhouette loss, leads to a significant improvement to the reconstruction quality of the handle.

This contradiction between the qualitative and quantitative data suggests that solely evaluating the quality of the reconstructed 3D structure through image-based loss functions is not ideal, particularly in scenarios with limited viewpoints. Consequently, this remains an open research problem.

5.4 *Effects of Number of View*

In order to assess the limitations of the currently available reconstruction methods, we formulated three additional scenes with varying numbers of available viewpoints. As depicted in Figure 23, the camera count was systematically reduced from four to one. For the case where two viewpoints were available, we utilized two opposing cameras. Our experiments indicate that while existing 3D reconstruction methods are capable

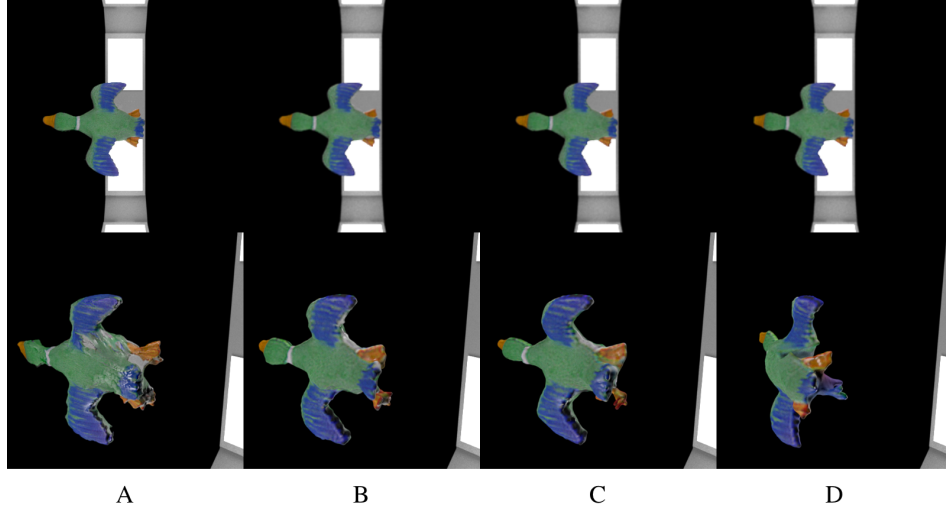


Figure 23: Qualitative results of the number of view experiment: The first row is rendered from an exact viewpoint of one of the cameras. The second row represents a viewpoint different from the camera viewpoints used in the optimization. Rows A, B, C, and D correspond to 4, 3, 2, and 1 number of cameras, respectively, which were used in the optimization.

of reconstructing the target object with only a single viewpoint, an increase in the number of viewpoints directly enhances the quality of the resulting 3D reconstruction.

5.5 *Effects of Correctly Capturing Illumination of The Real-World Scene*

To further validate the benefits of capturing the illumination parameters along with camera and scene parameters, we have designed two scenes. One scene has the correct illumination setting, identical with the real-world, while the other scene has a slightly different illumination setting. Figure 24 displays the resulting reconstructions from the camera’s viewpoint and a novel viewpoint that was not used in the scene optimization process.

Upon examining the reconstructions of the two scenes, it is evident that the renderings obtained from the camera’s viewpoint are highly similar to each other. However, when we further investigate the resulting image from the novel viewpoint, we observe severe artifacts in the reconstructed object within the scene that has the altered

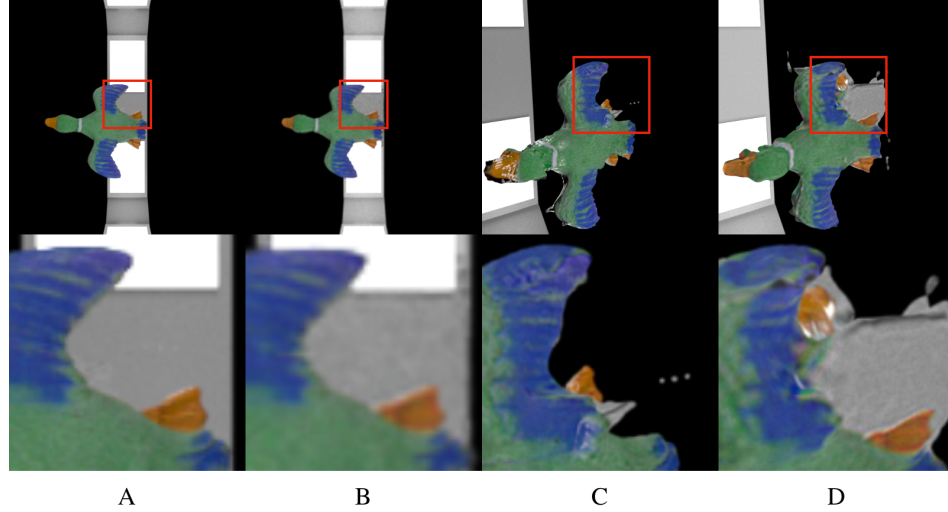


Figure 24: Qualitative results of the illumination ablation study: Columns A and C are rendered after 512 iterations in a correctly matched illumination setting with the real-world scene. Columns B and D are rendered after 512 iterations in a non-matching illumination setting with the real-world scene. The second row shows the magnification of the section inside the red box in the first row, in the same column.

illumination setting.

The 3D reconstruction method proposed by Vicini *et al.* [1] utilizes secondary derivatives to enhance the reconstruction of the target object. In this regard, correctly illuminating the scene forces the gradients, ensuring that the resulting reconstruction contains a minimal amount of artifacts. Thus, when the illumination is arbitrarily chosen or assumed to be constant, the amount of artifacts or imperfections are significantly increased. Therefore, including illumination parameters in the optimization process of the scene, which contains the target object, enhances the fidelity of the 3D reconstruction when limited number of views are available.

5.6 Current Limitations

Both 3D reconstruction methods have demonstrated their ability to accurately reconstruct real-world objects. However, it should be noted that these methods may yield sub-optimal results under certain circumstances. The presence of optical illusions introduces complexities not easily mitigated by image-based loss, particularly when

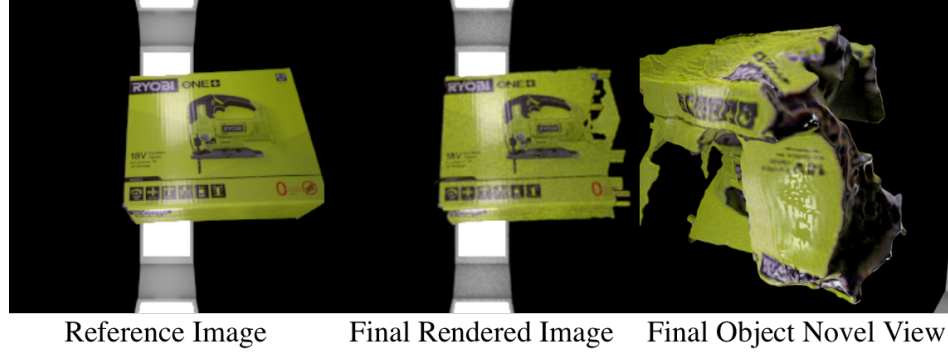


Figure 25: The first image is the reference image generated using real-world data. The second image is the state of the scene at the end of the optimization from the viewpoint of the virtual camera. The third image is the state of the optimized scene from a novel viewpoint.

the number of available viewpoints is limited. Figure 25 illustrates an example of this issue. Even after achieving convergence between the optimized state of the scene viewed from the virtual camera and the reference image, a further analysis of the 3D reconstruction from a novel perspective reveals discrepancies from the intended shape.

When analyzing turntable videos of the 3D reconstructed objects, the mentioned artifacts become more apparent. These artifacts are often the result of optical illusions and shape ambiguities derived from the camera’s two-dimensional perspective. Moreover, upon examining the turntable videos of ‘The Duck’ scene using 1, 2, 3, and 4 cameras, it becomes evident that the magnitude of the artifacts in the final 3D reconstruction increases as the number of cameras decreases. Thus, in cases with limited viewpoints, the effective evaluation of 3D shape from a 2D perspective becomes crucial.

As previously stated, assessing the geometry of an object using current image-based loss functions in limited viewpoint situations is not ideal. A potential solution to mitigate this issue might involve incorporating additional priors, such as depth information, into the scene optimization process.

CHAPTER VI

CONCLUSION

In this study, we initially investigate the real-world performance of state-of-the-art 3D reconstruction methods [1, 2] using the current methodology for real-world data acquisition [8], where only a limited number of viewpoints are available. However, it is worth noting that despite the ability of these methods to reconstruct the target object with only 4 viewpoints, the fidelity of the real-world results falls short in comparison to the reported results in virtual scenes. This discrepancy can be attributed to the complex surroundings and illumination settings encountered in real-world scenes.

At present, the existing real-world datasets solely capture camera parameters along with scene parameters, neglecting the crucial inclusion of illumination settings. Nevertheless, to attain performance levels similar to those observed in virtual scenes, the integration of illumination settings becomes imperative for state-of-the-art methods. In an effort to address this concern, we propose a methodology for capturing real-world scenes as virtual scenes, which incorporates both light source and camera data to more accurately represent reality in the virtual environment. Furthermore, through the implementation of our proposed real-world scene capture methodology, we have curated a collection of real-world scenes alongside their corresponding virtual counterparts, specifically designed for benchmarking purposes.

Additionally, we present a baseline method for 3D object reconstruction within these benchmark scenes and share the obtained results as a baseline for the reconstruction of real-world objects with limited viewpoints.

Our empirical investigations demonstrate that in scenarios characterized by limited viewpoints, current 3D reconstruction methods can successfully retrieve the target object with an acceptable level of quality. However, the inclusion of illumination data in conjunction with camera and scene parameters notably enhances the fidelity of the resulting reconstructions.

Given the considerable improvement in fidelity achieved by incorporating illumination data alongside camera and scene parameters, it is evident that there exists significant potential for further advancements in the domain of 3D reconstruction. One such potential improvement lies in the incorporation of depth information within our proposed methodology. Thus, the authors of this study intend to explore and report on the implementation of such an enhancement in the near future.

APPENDIX A

ADDITIONAL RESULTS

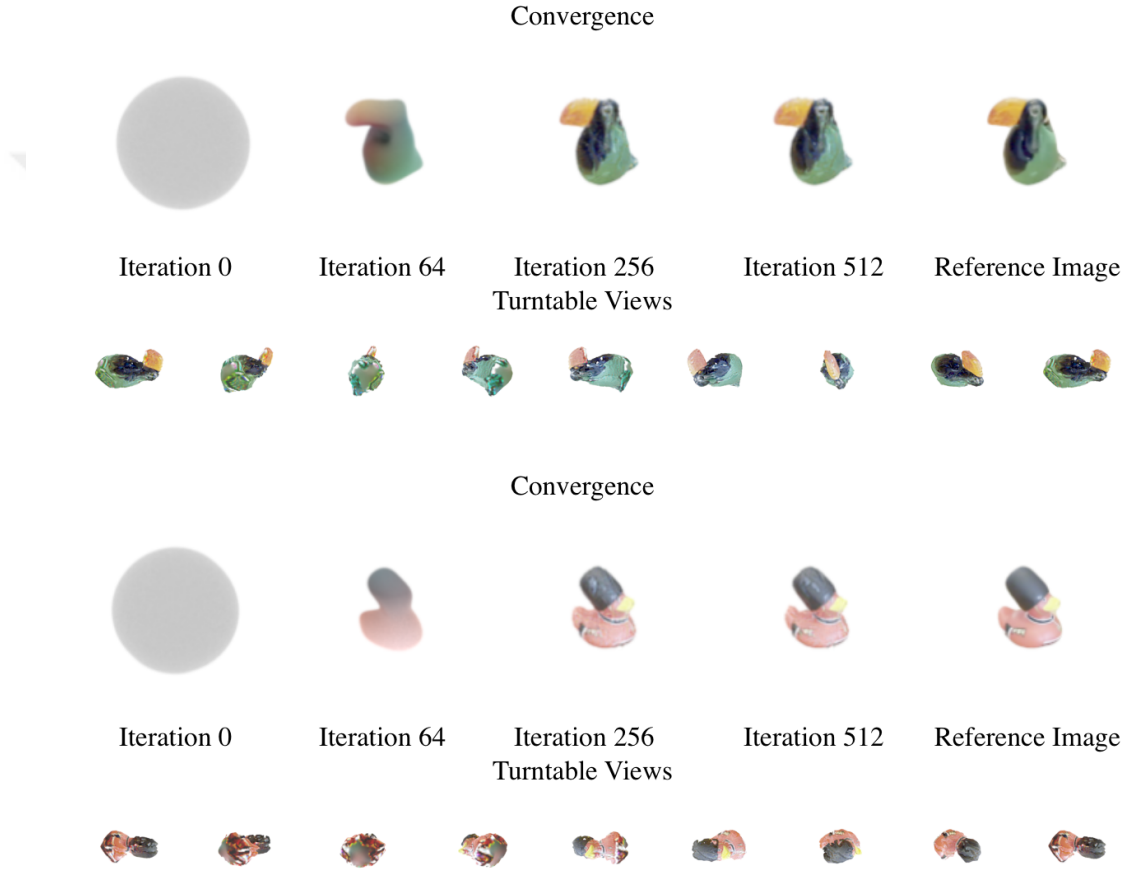


Figure 26: Single video setup results obtained using six non-synchronized images of the target object. Obtained using proposed method by Vicini *et al.* [1].

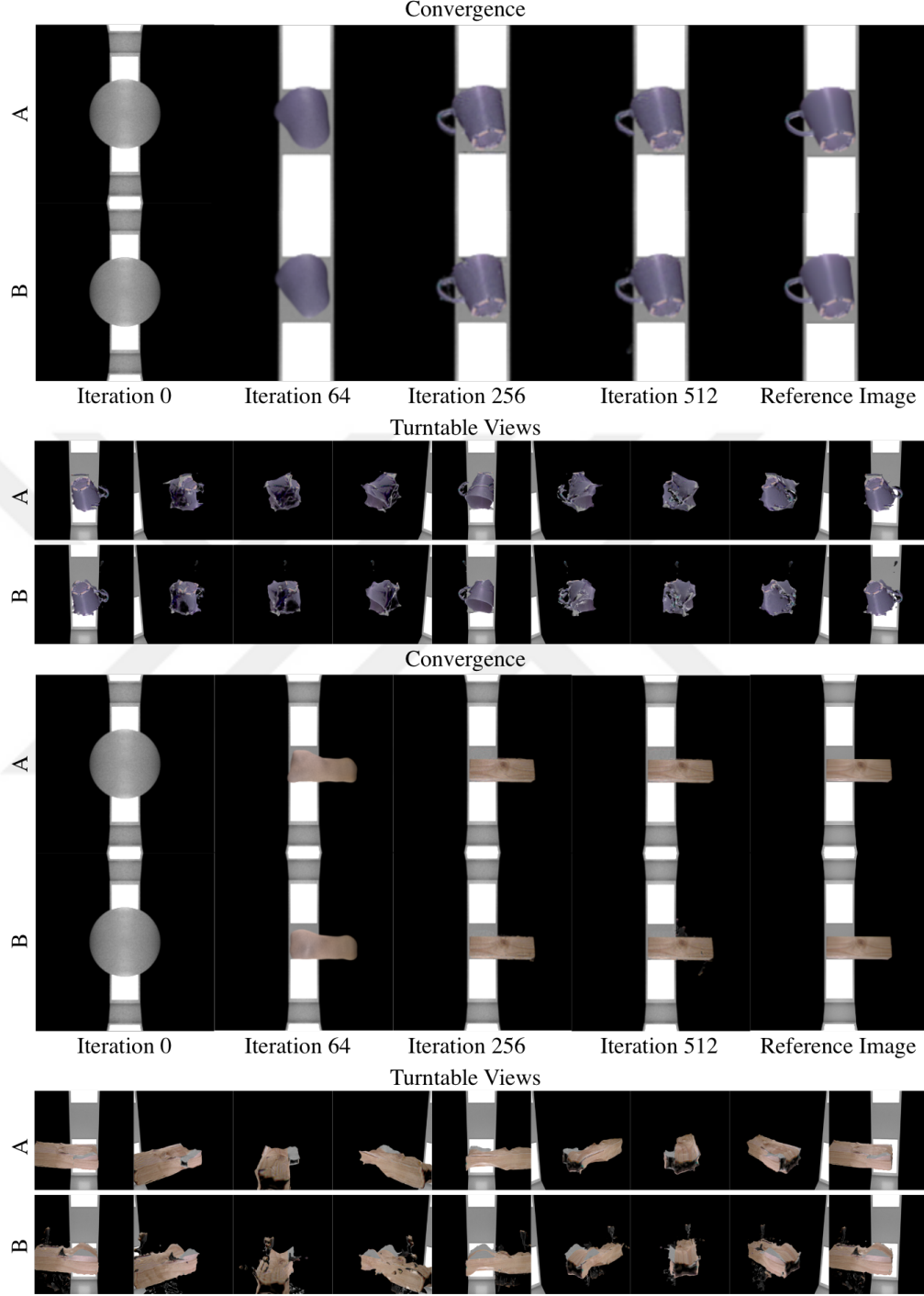


Figure 27: Multi-camera setup results obtained using four synchronized images of the target object. The rows marked with A presents the results obtained using proposed method by Vicini *et al.* [1]. The rows marked with B presents the results obtained using proposed method by Bangaru *et al.* [2].

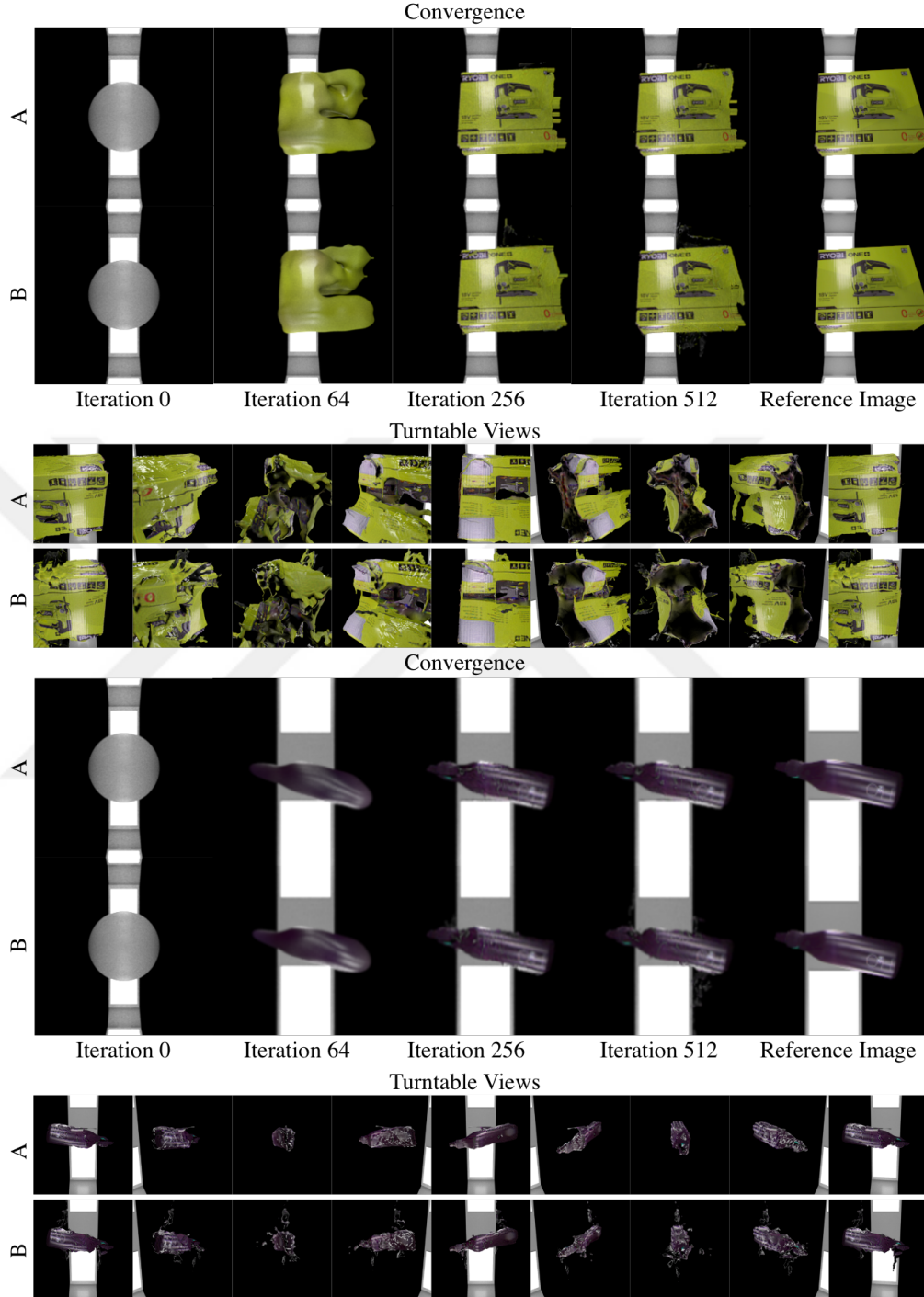


Figure 28: Multi-camera setup results obtained using four synchronized images of the target object. The rows marked with A presents the results obtained using proposed method by Vicini *et al.* [1]. The rows marked with B presents the results obtained using proposed method by Bangaru *et al.* [2].

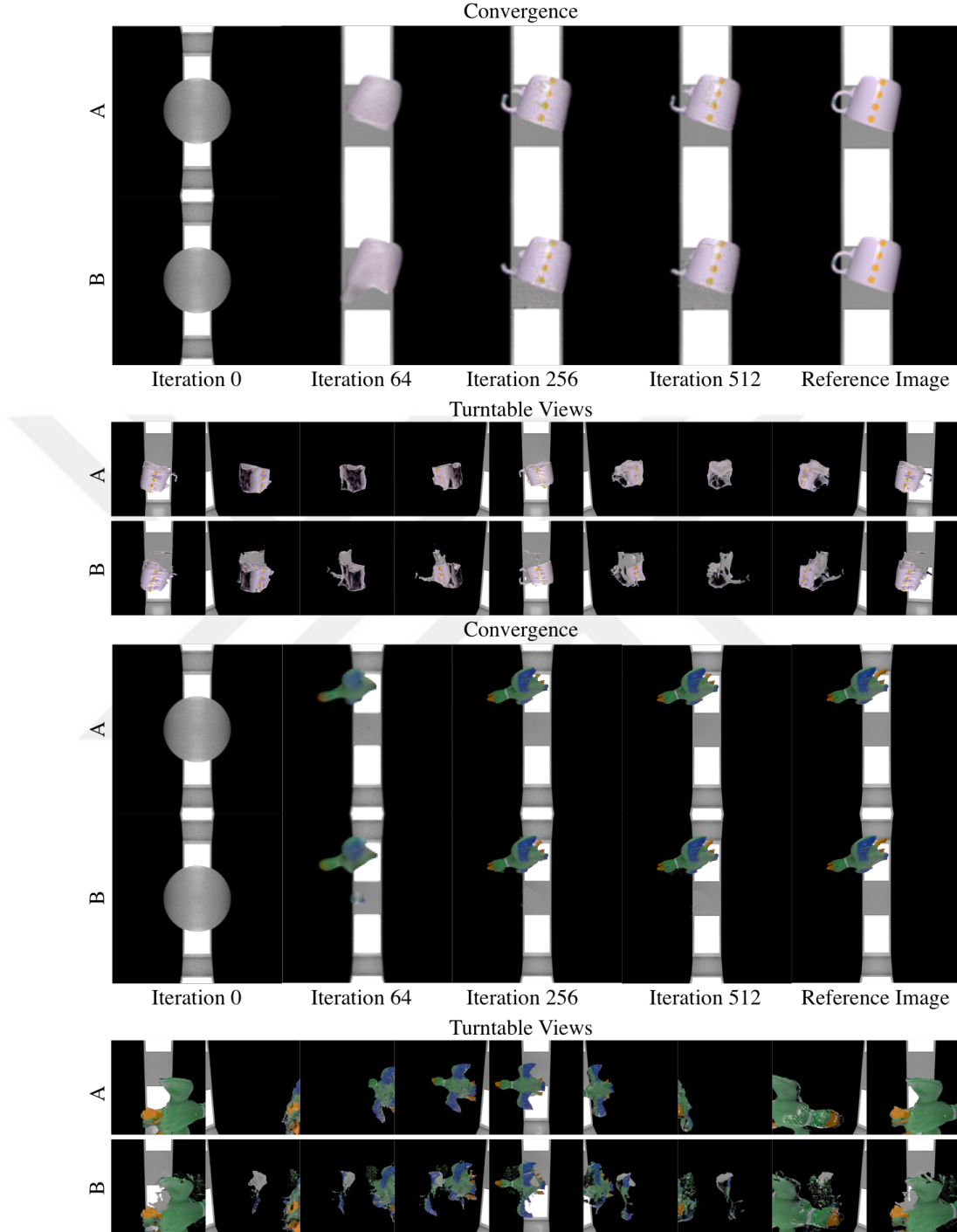


Figure 29: Multi-camera setup results obtained using four synchronized images of the target object. The rows marked with A presents the results obtained using proposed method by Vicini *et al.* [1]. The rows marked with B presents the results obtained using proposed method by Bangaru *et al.* [2].

Table 6: Raw quantitative results of the experiments conducted using our benchmark scenes. The experiment name is presented in the following format: Scene Name - Experiment Preset. Experiment presets are denoted as following: A: L1 - Vicini *et al.* [1] B: L2 - Vicini *et al.* [1] C: Multi-Scale L1 - Vicini *et al.* [1] D: Multi-Scale L1 + SSIM - Vicini *et al.* [1] E: Multi-Scale L1 + Silhouette - Vicini *et al.* [1] F: L1 - Bangaru *et al.* [2] G: L2 - Bangaru *et al.* [2] H: Multi-Scale L1 - Bangaru *et al.* [2] I: Multi-Scale L1 + SSIM - Bangaru *et al.* [2] K: Multi-Scale L1 + Silhouette - Bangaru *et al.* [2]

Experiment Name	L1	L2	SSIM	PSNR	LPIPS	FID	Experiment Name	L1	L2	SSIM	PSNR	LPIPS	FID
The Duck - A	0.0029	0.0006	0.9825	32.3782	0.0216	0.0059	Box - A	0.0082	0.0009	0.9465	30.4263	0.1084	0.2295
The Duck - B	0.0030	0.0005	0.9815	32.6064	0.0240	0.0078	Box - B	0.0081	0.0008	0.9494	30.8390	0.1010	0.1672
The Duck - C	0.0029	0.0006	0.9823	32.3849	0.0224	0.0081	Box - C	0.0081	0.0008	0.9489	30.8427	0.1021	0.1422
The Duck - D	0.0030	0.0006	0.9817	32.4386	0.0232	0.0083	Box - D	0.0087	0.0009	0.9443	30.3809	0.1087	0.1064
The Duck - E	0.0029	0.0006	0.9825	32.3342	0.0221	0.0061	Box - E	0.0081	0.0008	0.9500	30.7157	0.1029	0.1115
The Duck - F	0.0031	0.0006	0.9760	32.4246	0.0371	0.0079	Box - F	0.0078	0.0008	0.9475	31.0918	0.1155	0.1869
The Duck - G	0.0031	0.0006	0.9786	32.2955	0.0348	0.0085	Box - G	0.0075	0.0006	0.9405	32.2965	0.1232	0.1507
The Duck - H	0.0031	0.0006	0.9762	32.4814	0.0358	0.0086	Box - H	0.0080	0.0007	0.9413	31.3798	0.1197	0.0421
The Duck - I	0.0032	0.0006	0.9785	32.0330	0.0318	0.0083	Box - I	0.0081	0.0007	0.9406	31.3147	0.1233	0.1126
The Duck - K	0.0032	0.0006	0.9757	32.4130	0.0393	0.0087	Box - K	0.0078	0.0007	0.9450	31.3194	0.1196	0.1132
The Duck Shifted - A	0.0032	0.0005	0.9813	33.3641	0.0264	0.0073	Box Big - A	0.0111	0.0011	0.9088	29.7323	0.1621	0.2938
The Duck Shifted - B	0.0034	0.0005	0.9797	33.2728	0.0285	0.0088	Box Big - B	0.0112	0.0009	0.9050	30.4940	0.1508	0.3271
The Duck Shifted - C	0.0033	0.0005	0.9804	33.0457	0.0275	0.0087	Box Big - C	0.0111	0.0010	0.9089	29.9450	0.1595	0.3403
The Duck Shifted - D	0.0033	0.0005	0.9803	33.3706	0.0286	0.0093	Box Big - D	0.0106	0.0013	0.9126	30.7104	0.1633	0.3503
The Duck Shifted - E	0.0034	0.0005	0.9846	32.7813	0.0254	0.0075	Box Big - E	0.0130	0.0008	0.9019	28.7616	0.1668	0.5059
The Duck Shifted - F	0.0034	0.0005	0.9761	32.9419	0.0412	0.0081	Box Big - F	0.0106	0.0009	0.9028	30.5406	0.1825	0.3498
The Duck Shifted - G	0.0041	0.0008	0.9742	31.1081	0.0438	0.0099	Box Big - G	0.0111	0.0008	0.8914	30.8494	0.1764	0.3435
The Duck Shifted - H	0.0038	0.0007	0.9738	31.6943	0.0440	0.0098	Box Big - H	0.0117	0.0012	0.8991	29.3771	0.1831	0.3477
The Duck Shifted - I	0.0038	0.0007	0.9748	31.4961	0.0452	0.0101	Box Big - I	0.0128	0.0011	0.8995	29.5403	0.1850	0.0407
The Duck Shifted - K	0.0036	0.0006	0.9788	32.5878	0.0384	0.0082	Box Big - K	0.0124	0.0011	0.9047	29.5001	0.1749	0.5215
Cup Gray - A	0.0023	0.0003	0.9912	35.3724	0.0183	0.0070	Box Wooden - A	0.0029	0.0003	0.9871	34.6131	0.0267	0.0125
Cup Gray - B	0.0061	0.0011	0.9695	29.6506	0.0533	0.0133	Box Wooden - B	0.0030	0.0003	0.9865	34.7289	0.0263	0.0203
Cup Gray - C	0.0023	0.0003	0.9913	35.3780	0.0179	0.0085	Box Wooden - C	0.0030	0.0003	0.9864	34.8032	0.0278	0.0200
Cup Gray - D	0.0024	0.0003	0.9906	35.2570	0.0194	0.0061	Box Wooden - D	0.0031	0.0004	0.9852	34.5243	0.0285	0.0199
Cup Gray - E	0.0024	0.0003	0.9902	35.0134	0.0208	0.0052	Box Wooden - E	0.0029	0.0003	0.9865	34.8391	0.0272	0.0149
Cup Gray - F	0.0024	0.0003	0.9907	34.7224	0.0179	0.0055	Box Wooden - F	0.0034	0.0004	0.9747	34.3886	0.0558	0.0238
Cup Gray - G	0.0061	0.0011	0.9695	29.6506	0.0533	0.0133	Box Wooden - G	0.0033	0.0003	0.9787	34.6835	0.0443	0.0214
Cup Gray - H	0.0024	0.0003	0.9906	34.8930	0.0190	0.0069	Box Wooden - H	0.0033	0.0004	0.9772	34.4478	0.0514	0.0209
Cup Gray - I	0.0025	0.0003	0.9887	34.9149	0.0235	0.0056	Box Wooden - I	0.0033	0.0004	0.9784	34.3541	0.0473	0.0084
Cup Gray - K	0.0023	0.0003	0.9909	34.7388	0.0181	0.0050	Box Wooden - K	0.0034	0.0004	0.9742	34.4061	0.0522	0.0218
Cup White - A	0.0031	0.0005	0.9865	33.1885	0.0243	0.0049	Bottle Black - A	0.0026	0.0004	0.9880	34.3386	0.0231	0.0671
Cup White - B	0.0031	0.0005	0.9866	33.6878	0.0262	0.0055	Bottle Black - B	0.0065	0.0014	0.9590	28.6664	0.0653	0.0501
Cup White - C	0.0031	0.0005	0.9863	33.2131	0.0254	0.0067	Bottle Black - C	0.0026	0.0004	0.9882	34.4553	0.0240	0.0059
Cup White - D	0.0032	0.0004	0.9858	33.0785	0.0264	0.0074	Bottle Black - D	0.0028	0.0004	0.9864	33.8056	0.0653	0.0085
Cup White - E	0.0030	0.0004	0.9870	33.3934	0.0249	0.0076	Bottle Black - E	0.0065	0.0014	0.9590	28.6664	0.0269	0.0066
Cup White - F	0.0034	0.0005	0.9847	32.8805	0.0297	0.0052	Bottle Black - F	0.0027	0.0004	0.9863	33.7939	0.0266	0.0546
Cup White - G	0.0034	0.0005	0.9850	33.4463	0.0294	0.0069	Bottle Black - G	0.0065	0.0014	0.9590	28.6664	0.0653	0.0283
Cup White - H	0.0033	0.0005	0.9848	33.1484	0.0302	0.0078	Bottle Black - H	0.0029	0.0004	0.9842	33.6245	0.0346	0.0065
Cup White - I	0.0033	0.0005	0.9836	33.0124	0.0315	0.0079	Bottle Black - I	0.0029	0.0004	0.9856	33.7468	0.0305	0.0081
Cup White - K	0.0033	0.0005	0.9843	32.9680	0.0301	0.0080	Bottle Black - K	0.0028	0.0004	0.9858	33.9423	0.0300	0.0069
Cup White 2 - A	0.0032	0.0005	0.9866	32.8887	0.0255	0.0055	Bottle Black Shifted - A	0.0088	0.0054	0.9656	22.7067	0.0551	0.0083
Cup White 2 - B	0.0032	0.0005	0.9865	33.3093	0.0252	0.0048	Bottle Black Shifted - B	0.0088	0.0054	0.9672	22.7052	0.0529	0.0145
Cup White 2 - C	0.0033	0.0005	0.9859	33.0153	0.0270	0.0048	Bottle Black Shifted - C	0.0087	0.0054	0.9668	22.7147	0.0548	0.0143
Cup White 2 - D	0.0034	0.0005	0.9851	33.0264	0.0284	0.0051	Bottle Black Shifted - D	0.0088	0.0054	0.9667	22.7064	0.0555	0.0138
Cup White 2 - E	0.0033	0.0005	0.9857	32.8465	0.0288	0.0050	Bottle Black Shifted - E	0.0087	0.0053	0.9674	22.7176	0.0539	0.0141
Cup White 2 - F	0.0033	0.0006	0.9853	32.5448	0.0282	0.0050	Bottle Black Shifted - F	0.0088	0.0054	0.9661	22.7001	0.0568	0.0185
Cup White 2 - G	0.0035	0.0005	0.9845	33.0076	0.0305	0.0049	Bottle Black Shifted - G	0.0089	0.0054	0.9639	22.7036	0.0622	0.0145
Cup White 2 - H	0.0034	0.0005	0.9853	32.6221	0.0281	0.0055	Bottle Black Shifted - H	0.0088	0.0054	0.9666	22.7073	0.0569	0.0142
Cup White 2 - I	0.0037	0.0007	0.9827	32.9497	0.0297	0.0052	Bottle Black Shifted - I	0.0088	0.0054	0.9674	22.7134	0.0570	0.0138
Cup White 2 - K	0.0033	0.0005	0.9861	31.8513	0.0268	0.0050	Bottle Black Shifted - K	0.0086	0.0053	0.9671	22.7101	0.0554	0.0137

REFERENCES

- [1] D. Vicini, S. Speierer, and W. Jakob, “Differentiable signed distance function rendering,” *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–18, 2022.
- [2] S. P. Bangaru, T.-M. Li, and F. Durand, “Unbiased warped-area sampling for differentiable rendering,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–18, 2020.
- [3] B. Curless and M. Levoy, “A volumetric method for building complex models from range images,” in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 303–312, 1996.
- [4] V. Sitzmann, S. Rezhikov, B. Freeman, J. Tenenbaum, and F. Durand, “Light field networks: Neural scene representations with single-evaluation rendering,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 19313–19325, 2021.
- [5] A. Rana, A. S. Rawat, A. Bijalwan, and H. Bahuguna, “Application of multi layer (perceptron) artificial neural network in the diagnosis system: a systematic review,” in *2018 International conference on research in intelligent and computing in engineering (RICE)*, pp. 1–6, IEEE, 2018.
- [6] S.-H. Sun, M. Huh, Y.-H. Liao, N. Zhang, and J. J. Lim, “Multi-view to novel view: Synthesizing novel views with self-learned confidence,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 155–171, 2018.
- [7] P. Sinha and E. Adelson, “Recovering reflectance and illumination in a world of painted polyhedra,” in *1993 (4th) International Conference on Computer Vision*, pp. 156–163, 1993.
- [8] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotny, “Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10901–10911, 2021.
- [9] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “Nerf: Representing scenes as neural radiance fields for view synthesis,” *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.
- [10] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, “Nerf in the wild: Neural radiance fields for unconstrained photo collections,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7210–7219, 2021.

- [11] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, “Nerf–: Neural radiance fields without known camera parameters,” *arXiv preprint arXiv:2102.07064*, 2021.
- [12] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, “D-nerf: Neural radiance fields for dynamic scenes,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10318–10327, 2021.
- [13] H. Turki, D. Ramanan, and M. Satyanarayanan, “Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12922–12931, 2022.
- [14] V. S. Popescu, *Forward rasterization: A reconstruction algorithm for image-based rendering*. The University of North Carolina at Chapel Hill, 2001.
- [15] M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler, and M. H. Gross, “Real-time ray-casting and advanced shading of discrete isosurfaces,” in *Computer graphics forum*, vol. 24, pp. 303–312, Citeseer, 2005.
- [16] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan, “Interactive ray tracing for isosurface rendering,” in *Proceedings Visualization’98 (Cat. No. 98CB36276)*, pp. 233–238, IEEE, 1998.
- [17] M. Pharr, W. Jakob, and G. Humphreys, *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.
- [18] X. Tong, *Inverse Rendering*, pp. 1–4. Cham: Springer International Publishing, 2020.
- [19] G. Loubet, N. Holzschuch, and W. Jakob, “Reparameterizing discontinuous integrands for differentiable rendering,” *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–14, 2019.
- [20] M. Suhail, C. Esteves, L. Sigal, and A. Makadia, “Light field neural rendering,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8269–8279, 2022.
- [21] L. Melas-Kyriazi, I. Laina, C. Rupprecht, and A. Vedaldi, “Realfusion: 360deg reconstruction of any object from a single image,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8446–8455, 2023.
- [22] N. Lamb, S. Banerjee, and N. K. Banerjee, “Deepmend: Learning occupancy functions to represent shape for repair,” in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*, pp. 433–450, Springer, 2022.

- [23] Y. Jiang, D. Ji, Z. Han, and M. Zwicker, “Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1251–1261, 2020.
- [24] S. Liu, Y. Zhang, S. Peng, B. Shi, M. Pollefeys, and Z. Cui, “Dist: Rendering deep implicit signed distance function with differentiable sphere tracing,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2019–2028, 2020.
- [25] M. M. Loper and M. J. Black, “Opendr: An approximate differentiable renderer,” in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part VII 13*, pp. 154–169, Springer, 2014.
- [26] H. Kato, Y. Ushiku, and T. Harada, “Neural 3d mesh renderer,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3907–3916, 2018.
- [27] S. Liu, T. Li, W. Chen, and H. Li, “Soft rasterizer: A differentiable renderer for image-based 3d reasoning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7708–7717, 2019.
- [28] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila, “Modular primitives for high-performance differentiable rendering,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–14, 2020.
- [29] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, “Differentiable monte carlo ray tracing through edge sampling,” *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–11, 2018.
- [30] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 0–0, 2016.
- [31] J. Zhang, G. Yang, S. Tulsiani, and D. Ramanan, “Ners: neural reflectance surfaces for sparse-view 3d reconstruction in the wild,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 29835–29847, 2021.
- [32] Z. Li, Y.-Y. Yeh, and M. Chandraker, “Through the looking glass: Neural 3d reconstruction of transparent shapes,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1262–1271, 2020.
- [33] J. Sun, X. Chen, Q. Wang, Z. Li, H. Averbuch-Elor, X. Zhou, and N. Snavely, “Neural 3d reconstruction in the wild,” in *ACM SIGGRAPH 2022 Conference Proceedings*, pp. 1–9, 2022.
- [34] D. Wang, X. Cui, X. Chen, Z. Zou, T. Shi, S. Salcudean, Z. J. Wang, and R. Ward, “Multi-view 3d reconstruction with transformers,” in *Proceedings of*

- the *IEEE/CVF International Conference on Computer Vision*, pp. 5722–5731, 2021.
- [35] S. Hu, K. Zhou, K. Li, L. Yu, L. Hong, T. Hu, Z. Li, G. H. Lee, and Z. Liu, “Consistentnerf: Enhancing neural radiance fields with 3d consistency for sparse view synthesis,” *arXiv preprint arXiv:2305.11031*, 2023.
 - [36] J. Tong, S. Muthu, F. A. Maken, C. Nguyen, and H. Li, “Seeing through the glass: Neural 3d reconstruction of object inside a transparent container,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12555–12564, 2023.
 - [37] Y. Xiao, N. Xue, T. Wu, and G.-S. Xia, “Level-S $\hat{\}$ 2\$ fM: Structure From Motion on Neural Level Set of Implicit Surfaces,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17205–17214, 2023.
 - [38] C. Deng, C. Jiang, C. R. Qi, X. Yan, Y. Zhou, L. Guibas, D. Anguelov, *et al.*, “Nerdi: Single-view nerf synthesis with language-guided diffusion as general image priors,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20637–20647, 2023.
 - [39] Q. Shen, X. Yang, and X. Wang, “Anything-3d: Towards single-view anything reconstruction in the wild,” *arXiv preprint arXiv:2304.10261*, 2023.
 - [40] W. Dong, Q. Wang, X. Wang, and H. Zha, “Psdf fusion: Probabilistic signed distance function for on-the-fly 3d data fusion and scene reconstruction,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 701–717, 2018.
 - [41] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “DeepSDF: Learning continuous signed distance functions for shape representation,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.
 - [42] J. Zhang, Y. Yao, and L. Quan, “Learning signed distance field for multi-view surface reconstruction,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6525–6534, 2021.
 - [43] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs, “Large scale multi-view stereopsis evaluation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 406–413, 2014.
 - [44] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan, “Blendedmvs: A large-scale dataset for generalized multi-view stereo networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1790–1799, 2020.

- [45] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan, “Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6851–6860, 2019.
- [46] Y. Tian, Y. Long, D. Xia, H. Yao, and J. Zhang, “Handling occlusions in augmented reality based on 3d reconstruction method,” *Neurocomputing*, vol. 156, pp. 96–104, 2015.
- [47] K. H. Sing and W. Xie, “Garden: A mixed reality experience combining virtual reality and 3d reconstruction,” in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pp. 180–183, 2016.
- [48] B. O. Community, *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.
- [49] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [50] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” p. 6629–6640, 2017.
- [51] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

VITA

Doğa Yılmaz earned his Bachelor of Science in Computer Science from Özyeğin University in 2020 and subsequently began pursuing his Master's degree in Artificial Intelligence at the same institution in the following year.

Yılmaz is a member of the Vision and Graphics Laboratory, working under the guidance of Assist. Prof. Dr. Furkan Kırac. His primary areas of interest are computer vision, computer graphics, and deep learning, with a current research focus on 3D reconstruction of real-world objects in scenarios with limited viewpoints.