



3D Reconstruction Based on NeRF and SDF Methods: A Comparative Evaluation Using RGB-D Data

Zoi Dimou

(Registration Number: aivc22003)

Thesis submitted for the degree of
Master of Science in

Artificial Intelligence & Visual Computing
at the University of West Attica
in cooperation with the University of Limoges

Supervisor: Lazaros Grammatikopoulos, Associate Professor at UNIWA.

Athens, 2025

Μέλη εξεταστικής επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή

Α/α	ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
1	Γραμματικόπουλος Λάζαρος (Επιβλέπων)	Αναπληρωτής Καθηγητής, ΠΑΔΑ	
2	Πέτσα Ελένη	Καθηγήτρια, ΠΑΔΑ	
3	Γεώργιος Σφήκας	Επίκουρος Καθηγητής, ΠΑΔΑ	

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

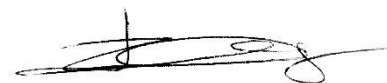
Η κάτωθι υπογεγραμμένη Δήμου Ζωή του Αντωνίου, με αριθμό μητρώου 22003 μεταπτυχιακή φοιτήτρια του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Η Δηλούσα

Δήμου Ζωή



This thesis was carried out at the University of West Attica under the supervision of Lazaros Grammatikopoulos, to whom I am deeply grateful for his guidance and trust. I would also like to thank Andreas El Saer for his invaluable help and patience. A heartfelt thank you goes to my friends, who are my chosen family, for their unwavering support. Last but certainly not least, I want to thank my family, who have stood by me throughout the years, supporting all my decisions. I would also like to extend my gratitude to Professor Elli Petsa for the knowledge she imparted, both at the undergraduate and postgraduate levels.

Abstract

This thesis explores the process of 3D scene reconstruction using a depth (RGB-D) camera, combined with advanced methodologies in artificial intelligence and visual computing. The research involves capturing real-world scenes using the RGB-D camera, followed by exporting each frame through multiway registration using the Open3D library to ensure accurate alignment and reconstruction. The core of this work is conducted within SDFStudio, an extension built on the NeRF studio framework, which facilitates the development and experimentation of methods involving Signed Distance Fields (SDFs). SDFs are crucial for representing 3D shapes and surfaces with precision, making them ideal for applications requiring accurate geometric computations.

Leveraging the modular design and features of SDFStudio, the research implements and compares three state-of-the-art SDF-based algorithms: Neural Unsigned Distance Fields - facto (NeuS-facto), UNISURF, and MonoSDF. These methods are tested on datasets comprising depth and RGB images along with known camera parameters (poses, and intrinsic). The performance and accuracy of the algorithms are systematically evaluated by adjusting key parameters, such as SDF grid resolution, number of iterations, and learning rates, to assess their impact on 3D reconstructions quality.

Keywords: Depth camera, 3D reconstruction, Neural network, Signed Distance Fields (SDF), Neural Radiance Field (NeRF), Photogrammetry, Visual Computing.

Contents

Abstract.....	6
Thesis Structure	16
1. Introduction	18
2. RGB-D Sensors.....	19
2.1. Why RGBD?	20
2.2. Historical Development and Evolution of RGBD Technology	20
3. Depth Sensing Technologies	21
3.1. Stereo Vision	22
3.1.1. Stereo Vision Principles.....	22
3.1.2. Passive stereo camera	26
3.1.3. Active stereo vision	27
3.1.4. Passive Vs Active stereo vision.....	29
3.2. Time of Flight (ToF) camera	29
3.3 D455 Intel RealSense	30
3.3.1. Coordinate Axes Description.....	32
4. 3D Registration.....	33
4.1. RGBD Images to Point cloud	33
4.2. Local refinement	35
4.2.1. ICP (Iterative Closest Point) Point-to-Plane	36
4.2.2. Colored ICP.....	37
4.2.3. Pose Graph.....	38
5. Novel 3D reconstruction approaches	39
5.1. NeRF	40
5.1.1. Neural Network Architecture of NeRF.....	40
5.2. SDF (Signed Distance Fields)	45
5.2.1. Ray Marching & SDFs	46
5.2.2. Surface Normals and Lighting	47
5.2.3. Constructive Solid Geometry (CSG)	48
6. Applications of NeRF and SDF in 3D Reconstruction	49
6.1. NeRF-studio.....	50
6.1.2. NeRF studio 's Core Components	50
6.2. 3D Reconstruction with SDFStudio	54
6.2.2. Surface Reconstruction Methods Supported by SDF Studio	55
7. Object Scanning Process	64

7.1. Sunlight Interference Issue	65
8. Pose Extraction and Camera Alignment	66
9. Parameter Selection and Preparation for Reconstruction	69
9.1. Comparison of Training Results and Reconstruction Quality of MonoSDF and UniSurf for the Selection of Number of Iterations	71
9.1.1. MonoSDF num iteration comparison.....	72
9.1.2. UniSurf num iteration comparison	74
9.2. Selection of Parameter Values for Comparative Model Analysis	75
10. Experiments	76
10.1. Models Comparison and Results for Each Parameter Adjustment	76
10.2. Individual Model Evaluation for Each Parameter Adjustment	97
10.3. Comparison of Ground Truth with the Best Reconstructed Mesh of each Algorithm	110
10.3.1. Neus-facto.....	111
10.3.2. MonoSDF.....	113
10.3.3. UniSurf	115
11. Conclusions	117
12. Future Works	118
Bibliography	120

List of figures

Figure 1: RGB and Depth export from a depth camera (Tang, et al., 2016)	19
Figure 2: Evolution of RGBD Technology	20
Figure 3: Existing Depth Sensing Technologies	21
Figure 4: Mismatch in Stereo vision (DEPTH SENSING TECHNOLOGIES OVERVIEW, n.d.)	22
Figure 5: Baseline length of Intel RealSense Depth cameras	23
Figure 6: Calibration Targets	24
Figure 7: Example of correspondence points between an image pair (Ma, et al., 2018).....	25
Figure 8: Example of relative distance between 2 corresponding pixels	25
Figure 9: Example of depth map	26
Figure 10: Equation for calculating depth in a stereo camera system	26
Figure 11: Illustration of a stereo vision system	27
Figure 12: Structured light patterns.....	27
Figure 13: How 3D reconstructions affected by sunlight examples	28
Figure 14: LiDAR distance measurement technique.....	29
Figure 15: Intel RealSense D455 sensors distances	30
Figure 16: Intel RealSense D455 camera	31
Figure 17: D455 infrared pattern	31
Figure 18: Lasers are categorized into 4 classes based on their optical power levels.....	31
Figure 19: Specifications of the Intel RealSense D455 capabilities.....	32
Figure 20: Coordinate system of D455	32
Figure 21: Scan metadata from .json file	33
Figure 22: Point cloud calculated from depth map	34
Figure 23: Depth and Color frame	34
Figure 24: Colored point cloud from depth and rgb frames.....	35
Figure 25: DownSampled colored point cloud.....	35
Figure 26: Transformation matrix (Matrix Transformations, n.d.)	36
Figure 27: Point to Plane method	36
Figure 28: Colored ICP geometric term	37
Figure 29: Colored ICP color term.....	37
Figure 30: Colored ICP balance term	38
Figure 31: Pose Graph Construction example (Pfungsthorn, 2014)	39
Figure 32: High-resolution 3d rendering from 2D images	40
Figure 33: Multi-Layer Perceptron (MLP) architecture commonly used in NeRF (Neural Radiance Fields)	41
Figure 34: Core concepts behind Neural Radiance Fields (NeRF).....	41
Figure 35: Example of an MLP with two hidden layers.....	42
Figure 36: Illustrates of the concept of Volume Rendering.....	43
Figure 37: Positional Encoding Technique	44
Figure 38: Hierarchical Volume Sampling Technique	45
Figure 39: SDF function.....	46
Figure 40: Negative distances inside the sphere (Left) and positive distances outside (Right).	46
Figure 41: Rays from the camera use SDFs to compute the distance to the nearest surface.....	46
Figure 42: Sphere Tracing for Optimizing Ray Marching	47
Figure 43: Visualization of a gradient with a value of 1.....	48
Figure 44: Normal vector used in lighting calculations.....	48
Figure 45: Creating complex 3D objects by combining simpler geometric	49

Figure 46: Overview of NeRF studio 's Modular Framework.....	50
Figure 47: A ray bundle, in which the rays have a large spread of propagation direction.....	51
Figure 48: Visualization of Frustum Parameterization and Sampling Techniques in 3D Volume Rendering.....	51
Figure 49: Pipeline Architecture in NeRFstudio for NeRF Training and Rendering	52
Figure 50: Example of camera path in viewer	53
Figure 51: Wandb platform.....	54
Figure 52: Data structure after converting the custom data to SDFStudio format	55
Figure 53: Nerfacto pipeline model	57
Figure 54: Backpropagation method	58
Figure 55: Nerfacto Field for extraction results	58
Figure 56: Hash encoding example (Tutorialspoint, n.d.).....	59
Figure 57: Pipeline comparison of Mip-NeRF and Mip-NeRF 360	60
Figure 58: UniSurf rendering pipeline.....	61
Figure 59: Depth and normal map from one image (Unity, n.d.)	62
Figure 60: MonoSDF Model Overview	63
Figure 611: Misalign merged point cloud	65
Figure 62: Saturation issue in RGB images in sunlight exposure	66
Figure 63: Transformation matrices for rotation of Y (left) and Z (right) axes	67
Figure 64: Transformation matrix for rotation of Y and Z axes	67
Figure 65: Final combined point cloud	68
Figure 66: Sample of poses.txt file ..	68
Figure 67: Average fitness and RMSE values after registration.....	68
Figure 68: Misalignment or noisy areas of the final combined point cloud	69
Figure 69: 3D scene representation divided into grid-like cubes	70
Figure 70: MonoSDF meshes after 200,000 iter. (up) & Mesh after 50,000 iter. (down)	73
Figure 71: UniSurf meshes after 200,000 iter. (up) & Mesh after 50,000 iter. (down)	74
Figure 72: PSNR Evaluation Metrics Across Models (Suggested Parameter Values).....	78
Figure 73: Learning Rate Progression Across Models (Suggested Parameter Values)	78
Figure 74: Sensor Depth and Image Evaluation Results (Suggested Parameter Values).....	79
Figure 75: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) using suggested parameter values.	79
Figure 76: Sensor Depth and Image Evaluation Results with pipeline.model.sdf-field.use-grid-feature set to False	80
Figure 77: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline.model.sdf-field.use-grid-feature set to False	81
Figure 78: Sensor Depth and Image Evaluation Results with pipeline.model.sdf-field.hidden-dim set to 64	82
Figure 79: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline.model.sdf-field.hidden-dim set to 64	82
Figure 80: PSNR Evaluation Metrics Across Models with pipeline.model.sdf-field.num-layers set to 8	83
Figure 81: Sensor Depth and Image Evaluation Results with pipeline.model.sdf-field.num-layers set to 8	83
Figure 82: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline.model.sdf-field.num-layers set to 8.....	84
Figure 83: Sensor Depth and Image Evaluation Results with pipeline.model.sdf-field.num-layers-color set to 8	84

Figure 84: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline.model.sdf-field.num-layers-color set to 8	85
Figure 85: PSNR Evaluation Metrics Across Models with pipeline.model.sdf-field.use-appearance-embedding set to False	86
Figure 86: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline.model.sdf-field.use-appearance-embedding set to False	86
Figure 87: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline.model.sdf-field.geometric-init set to False	87
Figure 88: PSNR Evaluation Metrics Across Models with pipeline.model.sdf-field.bias set to 0.05 ...	88
Figure 89: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline.model.sdf-field.bias set to 0.05	89
Figure 90: Sensor Depth and Image Evaluation Results with pipeline.model.sdf-field.beta-init set to 0.8	90
Figure 91: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline.model.sdf-field.beta-init set to 0.8	90
Figure 92: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with trainer.steps-per-eval-image set to 25	91
Figure 93: PSNR Evaluation Metrics Across Models with pipeline.model.sensor-depth-l1-loss-mult set to 0.8	92
Figure 94: Sensor Depth and Image Evaluation Results with pipeline.model.sensor-depth-l1-loss-mult set to 0.8	92
Figure 95: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline.model.sensor-depth-l1-loss-mult set to 0.8	93
Figure 96: PSNR Evaluation Metrics Across Models with pipeline.model.sensor-depth-freespace-loss-mult 0	94
Figure 97: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline.model.sensor-depth-freespace-loss-mult 0	94
Figure 98: Sensor Depth and Image Evaluation Results with pipeline.model.sensor-depth-sdf-loss-mult set to 0	95
Figure 99: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline.model.sensor-depth-sdf-loss-mult set to 0	96
Figure 100: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline.datamanager.train-num-rays-per-batch set to 2048	97
Figure 101: Neus-facto depth reconstruction of each test	99
Figure 102: Neus-facto RGB reconstruction of each test	100
Figure 103: Neus-facto 3D reconstruction of each test	101
Figure 104: MonoSDF depth reconstruction of each test	103
Figure 105: MonoSDF RGB reconstruction of each test	104
Figure 106: MonoSDF 3D reconstruction of each test	105
Figure 107: UniSurf Depth reconstruction of each test	107
Figure 108: UniSurf RGB reconstruction of each test	108
Figure 109: UniSurf 3D reconstruction of each test	109
Figure 110: Ground truth (from Terrestrial Laser Scanner) as reference	110
Figure 111: NeuSFacto point cloud	111
Figure 112: Absolute distances between NeuSFacto's pcd and GT's pcd	111
Figure 113: Precision, Recall and F-score values for different distance thresholds	112
Figure 114: Percentage of NeuSFacto points distances from ground truth	112
Figure 115: MonoSDF point cloud	113

Figure 116: Absolute distances between MonoSDF's pcd and GT's pcd	113
Figure 117: Precision, Recall and F-score values for different distance thresholds	114
Figure 118: Percentage of MonoSDF points distances from ground truth.....	114
Figure 119: UniSurf point cloud	115
Figure 120: Absolute distances between UniSurf's pcd and GT's pcd	115
Figure 121: Precision, Recall and F-score values for different distance thresholds	116
Figure 122: Percentage of UniSurf points distances from ground truth	116

List of tables

Table 1: Example parameter values used in the 3D reconstruction process with the NeuS model....	69
Table 2: MonoSDF training metrics presented for 200,000 and 50,000 iterations	72
Table 3: UniSurf training metrics presented for 200,000 and 50,000 iterations.....	74
Table 4: Parameter values for default settings, example cases and selected values for comparison in the training process	75
Table 5: Training metrics for suggested parameter values across Neus-Facto, MonoSDF, and UniSurf models.....	77
Table 6: Training metrics with pipeline.model.sdf-field.use-grid-feature set to False.....	80
Table 7: Training metrics for models with pipeline.model.sdf-field.hidden-dim set to 64	81
Table 8: Training metrics for models with pipeline.model.sdf-field.num-layers set to 8.....	83
Table 9: Training metrics for models with pipeline.model.sdf-field.num-layers-color set to 8	84
Table 10: Training metrics for models with pipeline.model.sdf-field.use-appearance-embedding set to False	85
Table 11: Training metrics for models with pipeline.model.sdf-field.geometric-init set to False.....	87
Table 12: Training metrics for models with pipeline.model.sdf-field.bias set to 0.05	88
Table 13: Training metrics for models with pipeline.model.sdf-field.beta-init set to 0.8	89
Table 14: Training metrics for models with trainer.steps-per-eval-image set to 25	90
Table 15: Training metrics for models with pipeline.model.sensor-depth-l1-loss-mult set to 0.8	91
Table 16: Training metrics for models with pipeline.model.sensor-depth-freespace-loss-mult 0.....	93
Table 17: Training metrics for models with pipeline.model.sensor-depth-sdf-loss-mult set to 0	95
Table 18: Training metrics for models with pipeline.datamanager.train-num-rays-per-batch set to 2048	96
Table 19: Training Time and PSNR for Neus-Facto Across Different Parameter Configurations.....	97
Table 20: Training Time and PSNR for MonoSDF Across Different Parameter Configurations	102
Table 21: Training Time and PSNR for UniSurf Across Different Parameter Configurations.....	106
Table 22: Evaluations of all methods for each distance threshold.....	116

Thesis Structure

This thesis is organized into several sections that thoroughly examine the process of a 3D reconstruction using data obtained from an Intel RealSense D455 RGB-D sensor, providing both a technical and conceptual framework for understanding the depth detection and reconstruction methods applied in this study.

Initially, this thesis provides an overview of RGB-D cameras and describes several applications where their ability to provide 3D data is creating new opportunities in robotics, augmented reality as well relevant computer vision fields.

Subsequently, the 3rd chapter explores various techniques used for depth detection, such as stereoscopic vision, structured light, and time-of-flight cameras. For each technique, the advantages, disadvantages, limitations, and suitability for different applications are discussed. A detailed presentation of the Intel RealSense D455 camera is also provided, including its technical specifications, capabilities, and the way it generates accurate depth maps, emphasizing the importance of selecting the appropriate technology for reliable 3D reconstructions.

Chapter 4th presents methods for performing 3D registration, focusing on techniques such as ICP Point-to-Plane, Colored ICP, and the Pose Graph method, which optimizes these processes.

The 5th chapter examines novel methods for 3D reconstruction, such as NeRF and SDF. The capabilities of NeRFstudio are presented, followed by SDFstudio, which is based on NeRFstudio and focuses on scene reconstruction through Signed Distance Fields (SDFs). The algorithms NeuS-Facto, UniSurf, and MonoSDF are analyzed in detail.

In the 7th chapter, it describes the workflow for scanning the area of interest using the RGB-D camera, providing significant information on optimal scanning practices.

Chapter 8 details the process of extracting the camera poses during scanning, describing in depth the procedure for merging the data. While in 9th section a comprehensive analysis of the parameters adjusted during training for model comparison is also provided, aiming to improve the accuracy and quality of the reconstruction.

The 10th section presents and analyzes the experimental results. The performance of the MonoSDF, UniSurf, and Neus-Facto models is compared under various training conditions. Changes in parameters and iterations are examined, and the models' responses to challenges such as complex geometry, occlusions, and noise in the data are evaluated. Additionally, a comparison is conducted between the results (ground truth) obtained through FARO3D focus and the optimal mesh reconstruction of each model.

The concluding section summarizes the key findings of the experiments, highlighting the performance of each neural network and identifying the configurations and algorithms that yielded the best results. Finally, the thesis concludes with suggestions for future research, including potential applications and further exploration of SDFstudio and similar algorithms.

1. Introduction

The advent of RGB-D cameras has revolutionized in computer vision, by enabling simultaneous capture of color and depth information in a single frame. This indeed constitutes a wholly different way in which machines will perceive and interact with the world surrounding them. In contrast with the traditional camera, which captures only two-dimensional images based on color, RGB cameras are enabled to perceive the spatial structure of a scene. This has led to new innovations across a wide variety of applications-from 3D scene reconstruction and AR/VR robotics to autonomous systems-all demanding spatial awareness for purposes of object detection, navigation, and real-time decision-making, among other tasks.

Currently, one of the most popular RGB-D devices on the market is the Intel RealSense D455; its depth-sensing capabilities are pretty accurate. It captures depth through a stereo vision system, where two stereo cameras calculate the distance from objects in a scene. The D455 stands out by its extended depth range and the low error rate, and therefore, is an ideal tool for 3D scanning, robotic perception, and interactive digital experiences. This makes it suitable for applications acquiring large environments where there is also the need for high accuracy. The capturing of RGB-D data is only the first step in a larger process involving meaningful creation of 3D models. While raw data from RGB-D cameras are provided per-pixel as color and depth values, transforming them into a coherent, high-fidelity 3D model remains one of the challenging issues. Transformation from depth maps and point clouds into full 3D reconstructions requires robust computational methods. Conventional approaches face many practical issues such as noise, incomplete data, and irregular surfaces, seriously affecting the quality of 3D reconstructions. Moreover, traditional methods involving voxel grids and mesh-based models can be computationally expansive and inefficient for cases of large-scale scenarios or complex geometries.

This is where the integration of neural networks particularly those based on Signed Distance Fields comes in as a state-of-the-art solution. SDFs allow one of the potent ways of representation of 3-D shapes by encoding the distance of every point in space from the closest surface. By doing so, this will not only allow for more accurate surface reconstruction but also improve memory efficiency and computational performance. By utilizing deep learning techniques, it is possible for SDF-based networks to learn how to make inferences on missing data, handle noisy inputs, and produce smooth high-resolution surfaces-even in cases where traditional algorithms cannot.

2. RGB-D Sensors

Traditional cameras capture images using a two-dimensional array of pixels, where each pixel represents RGB (Red, Green, Blue) values that define its color and intensity. These classical RGB cameras excel at capturing high-resolution color images, making them indispensable in applications requiring color accuracy and high detail, such as professional photography, videography, and digital arts. Their ability to render true-to-life colors and fine details is unparalleled, making them the go-to choice when image fidelity is the primary concern.

However, despite their strengths, traditional RGB cameras are limited by their lack of depth perception. They capture the world in two-dimensional planes, lacking the spatial awareness necessary to understand the relative distances between objects in a scene. This limitation can be a significant drawback in applications that require an understanding of the three-dimensional structure of an environment, such as in robotics, augmented reality (AR), virtual reality (VR), and advanced computer vision systems.

In contrast, RGBD cameras, or depth cameras, offer a sophisticated imaging system that captures both color (RGB) and depth data simultaneously. This dual capability allows RGBD cameras to provide a richer spatial understanding of a scene by combining traditional color information with depth perception (Figure 1). The depth data is typically captured using various technologies, such as structured light, time-of-flight, or stereo vision, and it enables the camera to measure the distance between the sensor and objects in the scene with remarkable accuracy. (Beginner's guide to depth (Updated), 2019)

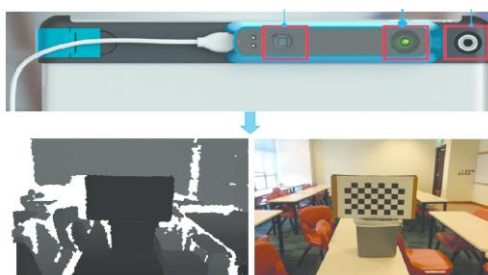


Figure 1: RGB and Depth export from a depth camera (Tang, et al., 2016)

This ability to capture depth information opens a wide range of possibilities that traditional RGB cameras cannot achieve. For instance, in robotics, RGBD cameras are essential for navigation and object recognition, allowing robots to interact more intelligently with their environment. In AR and VR, RGBD cameras enable more immersive experiences by accurately mapping the physical world into digital environments, allowing for realistic object placement and interaction. Additionally, in fields like 3D modeling and construction, RGBD cameras facilitate the creation of accurate three-dimensional representations of spaces, which can be crucial for design and planning.

The comparison between RGBD cameras and traditional RGB cameras highlights the importance of selecting the appropriate imaging system based on specific application requirements. While RGBD cameras provide depth perception and a deeper spatial understanding, traditional RGB cameras remain unmatched in color fidelity and imaging resolution. Understanding the strengths and

limitations of each technology are crucial for making informed decisions when choosing a camera system for a given task or application. (Henry, Krainin, Ren, & Fox, 2014)

2.1. Why RGBD?

Incorporating RGB data into the point cloud or depth map generated by a 3D depth sensing camera enhances the precision of object pinpointing, facilitating pattern recognition and detection. This capability proves particularly advantageous in applications requiring the identification and categorization of objects within a scene, coupled with depth measurement to those objects. Furthermore, the capacity to provide both forms of data within a single frame renders the camera especially well-suited for applications such as facial recognition-based anti-spoofing systems and people counting devices (Kumar, 2022).

The primary constraints hindering the integration of RGB-D (Red, Green, Blue, and Depth) sensors in high-precision surveying applications include the restricted depth range of the sensor, typically around three meters for structured light-based RGB-D sensors like the Structure Sensor, and approximately five meters for time-of-flight-based RGB-D sensors such as the Kinect v2. Additionally, there is a challenge with successive brittle frame registration, resulting in diminished tracking or bias in camera pose due to fewer distinctive features in frames (Darwish, Li, Tang, Li, & Chen, 2019).

2.2. Historical Development and Evolution of RGBD Technology

The historical development and evolution of RGBD (Red, Green, Blue, and Depth) technology have been marked by significant advancements, tracing back to early research in computer vision and depth sensing. These advancements have ultimately led to the commercialization of RGBD cameras, offering insights into their capabilities and limitations.

There have been significant developments in the technology of cameras with 3D imaging sensors. Figure 2 illustrates the historical progress of this development, although the related technology has been developed in recent years. The foundations were first laid in 1989. The 3D reconstruction and the RGB-D technology was appreciated a lot, where since 1970 bases related to 3D modelling and the importance of shape were introduced. In the 1980s, tools for understanding the geometry of objects were applied and in the 2000s techniques and methods related to the characteristics and textures of objects and scenes were developed. By the end of 2010, relevant algorithms had been designed and applied to dynamic environments and robotics, with the most satisfactory high-fidelity models being derived from deep learning methods.

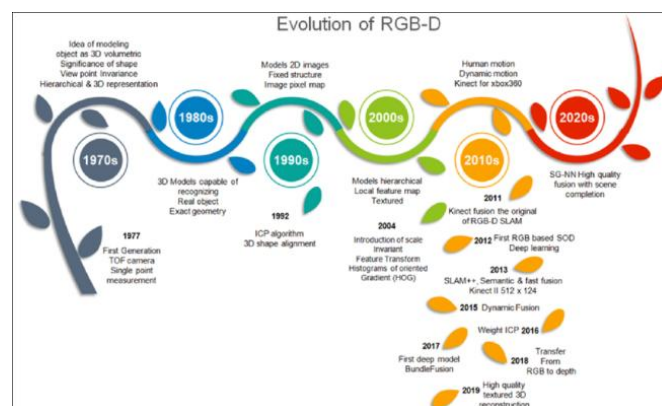


Figure 2: Evolution of RGBD Technology

Presently, research efforts are directed towards overcoming existing limitations associated with the quality fusion of scenes. Data acquisition from depth cameras plays a crucial role in further processing to produce qualitative and accurate 3D reconstructed models of the physical world. The incorporation of sensors into depth cameras is of paramount importance, with rapid evolution owing to parallel developments in technologies.

Depth cameras typically utilize two main types of sensors: active and passive, which complement each other in various implementations. While sensors offer numerous benefits, they may also introduce errors and inaccurate measurements. Calibration is often necessary to achieve a high degree of detail in 3D reconstructions.

In summary, RGB-D cameras have seen significant development over the past decades, with advancements in sensor technologies, algorithms, and applications. The ongoing evolution of RGBD technology reflects the continuous quest for improved performance, accuracy, and reliability in capturing and processing three-dimensional data. (Tychola, Tsimperidis, & Papakostas, 2022)

3. Depth Sensing Technologies

There exist various methods for obtaining data from depth cameras, which can be categorized into active and passive sensing, along with the relatively recent development of monocular depth estimation (Figure 3). Active sensing techniques involve emitting structured energy to capture objects in static environments, enabling the simultaneous capture of entire scenes (Salvi, Pages, & Batlle, 2004). This simplifies 3D reconstruction, with subcategories including time-of-flight (ToF) and structured light (SL) cameras (Alexa, 2003). Passive sensing relies on the triangulation principle and epipolar geometry to determine the depth of key points in a scene. Monocular depth estimation, on the other hand, utilizes two-dimensional images for reconstructing 3D objects (Ibrahim, et al., 2020).

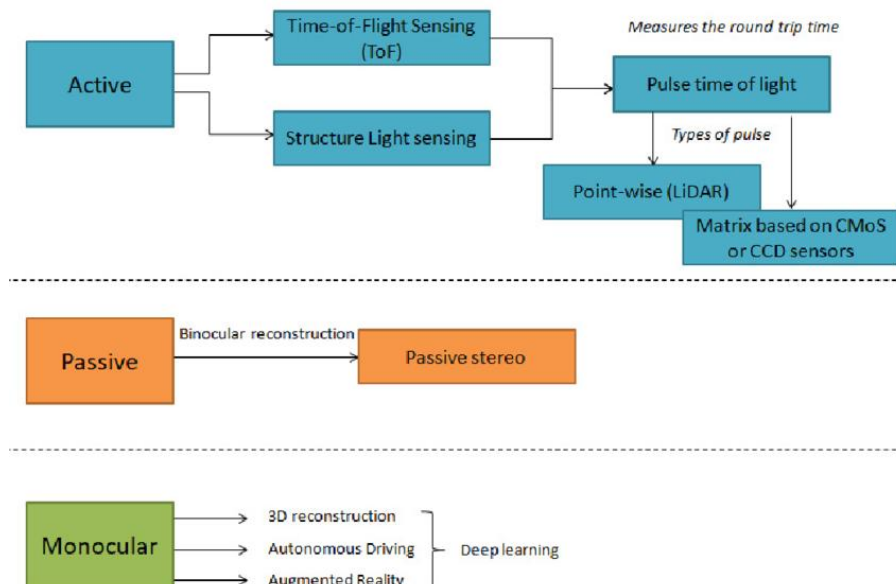


Figure 3: Existing Depth Sensing Technologies

To fully understand the different types of depth cameras and how they work, the following sections of this thesis provide a detailed analysis.

3.1. Stereo Vision

Stereoscopic vision technology mirrors the fundamental principles of human vision by leveraging the perception of depth using two cameras mounted on a common baseline, with a fixed distance between their lens centers (projection centers). Each camera captures the scene from a slightly different angle, resulting in a phenomenon known as parallax, where objects closer to the cameras exhibit a greater lateral shift of features between the two images (Figure 4). This disparity is then used to compute a depth map, which represents the measurable perception of depth.

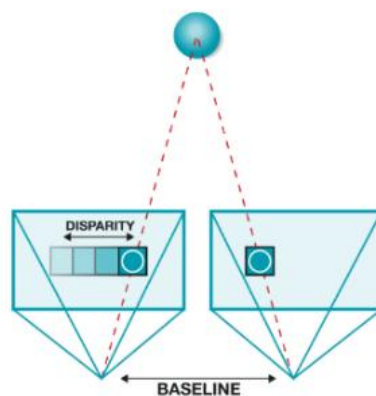


Figure 4: Mismatch in Stereo vision (DEPTH SENSING TECHNOLOGIES OVERVIEW, n.d.)

Stereo vision systems are broadly categorized into passive and active systems. Passive systems rely on ambient or artificial lighting to illuminate objects, while active systems enhance feature recognition by projecting random patterns onto the scene, often using infrared laser projectors with pseudo-random patterns or structured light. Active stereoscopic vision offers several advantages, including robustness to varying lighting conditions and the ease of installing multiple cameras without interference.

The measurement of 3D target points from multiple images within this technology is based on three main principles, with triangulation being the most prevalent. Triangulation is possible when two key conditions are met: first, the relative positions between the viewpoints must be known, which is typical in RGB-D cameras; and second, the directions from the viewpoints to the targets must be determined, which relies on the correct calibration of the camera setup. This integration of principles allows for accurate depth perception and is crucial for the effectiveness of stereoscopic vision systems. (Sugihara, 1986)

3.1.1. Stereo Vision Principles

3.1.1.1. Baseline

Manufacturers provide default baseline configurations for their stereo camera systems (Figure 5) based on typical usage scenarios, considering key factors like baseline distance and resolution. These configurations are essential for achieving optimal depth estimation across different applications. For instance, a larger baseline typically allows for better depth resolution at longer distances but may introduce challenges in close-range depth estimation, as it increases disparity error for closer objects. Conversely, a smaller baseline is advantageous for close-range depth perception but may limit depth accuracy at longer distances due to reduced disparity.

Selecting the optimal baseline and resolution is a critical decision that depends on the specific application and environment. The baseline, defined as the horizontal distance between the two cameras, directly affects both the range and accuracy of depth estimation. A larger baseline is beneficial for applications like aerial mapping, where objects of interest are far away. However, it can reduce the accuracy for nearby objects, making it unsuitable for close-range tasks like robotics. On the other hand, a smaller baseline is ideal for close-up applications but might struggle with faraway objects.

Stereo Module	Intel® RealSense™ Depth Module D410	Intel® RealSense™ Depth Module D415	Intel® RealSense™ Depth Module D430	Intel® RealSense™ Depth Module D450	Intel® RealSense™ Depth Module D401
Baseline	55 mm	55 mm	50 mm	95 mm	18 mm

Figure 5: Baseline length of Intel RealSense Depth cameras

Resolution plays an equally important role in stereo vision. Higher resolution provides more detail and can improve depth accuracy by enabling finer disparity calculations. However, it also increases computational load and memory requirements, which can be a challenge in real-time applications that require quick responses. Lower resolution, while reducing computational demands, may degrade depth estimation quality by introducing more noise and ambiguity in pixel matching. (How do you choose the optimal baseline and resolution for a stereo vision camera?, 2024)

3.1.1.2. Calibration

RGB-D camera heavily relies on the high-precision calibration of its constituent cameras. Traditional calibration methods for optical cameras based on the "pinhole" perspective model, such as Zhang's calibration method, have been extensively studied. However, the calibration of the infrared sensor in RGB-D cameras remains a significant challenge. Traditional feature extraction algorithms encounter difficulties in infrared images, and accessing auxiliary information like the disparity image for depth correction proves challenging in practice. (Yang, Danqing, Jun, Mingyi, & Yubin, 2022)

Calibration is essential for accurately computing depth information in stereo cameras. It involves determining parameters such as intrinsic matrices, distortion coefficients, and extrinsic parameters like rotation and translation between cameras. Calibration ensures proper rectification and matching of images from both cameras, enabling accurate depth computation. (Basso, Menegatti, & Pretto, 2017)

The Intel RealSense™ Depth Cameras D400-series like the one used in this thesis, are based on stereo vision and designed to maintain calibration and performance over their lifetime. However, factors like extreme temperature cycling or excessive shock and vibration can degrade performance over time. To address this, Intel provides tools for recalibrating cameras back to their factory condition, including OEM (Original Equipment Manufacturer) calibration based on targets (Figure 6) and dynamic calibration methods that restore performance in the field.

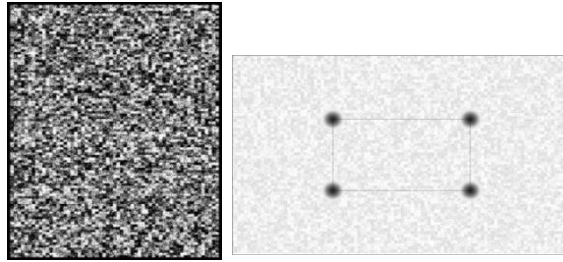


Figure 6: Calibration Targets

Intel introduces a set of components called "Self-Calibration" in its RealSense™ SDK2.0¹, aimed at restoring depth performance and improving accuracy for Intel RealSense™ Depth Camera D400 series that may have degraded over time. These components, running on any operating system or compute platform, invoke new firmware functions inside the ASIC with minimal load on the host CPU. They require no motion or repositioning during calibration and can complete in seconds. While RealSense cameras are designed to maintain calibration, these tools serve as a validation of device performance and can monitor calibration state over time without special targets. (Anders, et al.)

In summary, the calibration of stereo cameras, particularly in RGB-D systems, is critical for achieving accurate depth information and ensuring reliable performance in various applications, including robotics and computer vision. Continuous monitoring and recalibration methods provided by manufacturers are essential for maintaining optimal performance over the camera's lifetime.

3.1.1.3. Features Extraction Using key point-based Stereo Matching

Stereo matching, a cornerstone of stereo vision and computer vision, involves the task of identifying corresponding points or features across images captured from different viewpoints, commonly referred to as left and right images. This process facilitates the computation of depth information, thereby enabling the creation of 3D representations of scenes. In stereo matching, disparities, or differences between corresponding points in left and right images play a crucial role in determining object depth. Larger disparities correspond to objects closer to the camera, while smaller disparities indicate objects farther away. However, the stereo matching process is fraught with complexities, including occlusions, textureless regions, illumination variations, and the imperative of computational efficiency, especially in real-time applications. To tackle these challenges, various algorithms and techniques have been devised, including correlation-based methods, feature-based methods, and deep learning approaches. These methodologies aim to establish correspondences between image pairs (Figure 7) accurately and efficiently, thus facilitating precise depth estimation and 3D reconstruction in stereo vision applications. (Sun, Mei, Jiao, Zhou, & Wang, 2011)

¹ <https://dev.intelrealsense.com/docs/self-calibration-for-depth-cameras>

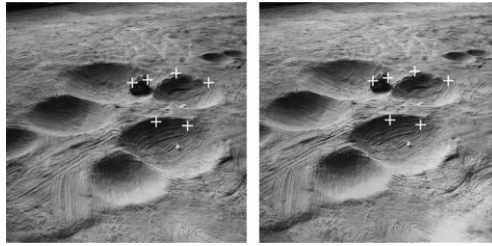


Figure 7: Example of correspondence points between an image pair (Ma, et al., 2018)

In parallel, feature extraction serves as a fundamental domain within computer vision, closely intertwined with tasks like object recognition, image matching, and synthesis. The objective of feature extraction is to pinpoint robust positions invariant to image features, sizes, camera viewpoints, and lighting conditions.

To address this limitation, Mikolajczyk's Harris Laplacian method emerges as a complementary approach, adept at detecting Harris corner points across various scales, ensuring resilience against scale variants. Shi and Tomasi further refined corner detection by proposing the Shi-Tomasi corner, which accounts for affine transformations. However, Lowe's SIFT (Scale Invariant Feature Transform) method emerges as the most renowned. SIFT employs a four-step process encompassing scale-space extrema detection, keypoint localization, orientation assignment, and keypoint description.

In the scale-space extrema detection phase, keypoints are identified as maxima/minima of the Difference of Gaussians (DoG) across multiple scales. Keypoint localization involves precise positioning, scale determination, and principal curvature ratio estimation, followed by orientation assignment based on local image gradient directions, ensuring rotation invariance. Finally, keypoint descriptors are computed to ensure distinctiveness and partial invariance to remaining variations like illumination and 3D viewpoint. (Kim, et al., 2022)

3.1.1.4. Triangulation

In the world of stereoscopic cameras, triangulation serves as a foundation for understanding depth. Stereoscopic cameras, as mentioned above, consisting of a left and right lens, project two-dimensional images of a scene. By knowing the relative distance between the cameras and the camera parameters, the depth (d) of each point (S) (Figure 8), whose projections are embedded in the corresponding pixels (L and R) in both the left and right images, can be determined.

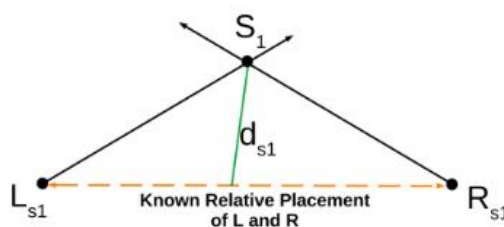


Figure 8: Example of relative distance between 2 corresponding pixels

Epipolar geometry describes the complex relationships between 3D points and their projections in 2D images. Despite the limitations of "planar projection", the differences between stereoscopic images, called "disparity", serve the perception of depth.

The essence of triangulation lies in the directional vectors emanating from the pixels in the stereoscopic images. These vectors converge at points in the 3D scene, allowing depth to be calculated perpendicular to the line joining the cameras. By applying the Pythagorean theorem, the depth of each point is accurately discerned.

Triangulation and disparity maps are the cornerstone of the implementation of stereoscopic vision. Triangulation extracts information about where objects in the image are located in three-dimensional space. Inequality maps, derived from pixel differences, help create depth maps (Figure 9), that provide information for spatial understanding.



Figure 9: Example of depth map

Today, modern computers mimic the complex process of stereoscopic vision by deriving disparity maps from pairs of stereoscopic images. Through triangulation and meticulous depth calculations, computers decipher the spatial complexities embedded in scenes, enhancing visual understanding and spatial perception. (Computer Vision: Stereo 3D Vision, 2022)

3.1.2. Passive stereo camera

Passive stereoscopic vision is a method that aims to extract three-dimensional (3D) information about the environment using a pair of cameras. The process involves taking images from a left and a right camera, with a baseline distance separating them. When corresponding points are detected in the left and right images, the difference in their positions is recorded as a difference, which is related to depth.

The relationship between dissimilarity (in pixels) and depth is expressed by a mathematical relationship (Figure 10). It illustrates that as the baseline distance and focal length increase, the depth resolution improves. The accuracy of depth is influenced by baseline distance, resolution, and lens focal length. (Sharma, Kim, & Singh, 2012)

$$\text{Depth} = \frac{\text{Baseline} \times \text{FocalLength}}{\text{Disparity}}$$

Figure 10: Equation for calculating depth in a stereo camera system

Stereo cameras operate by matching pixels between images from each camera and triangulating pixel depth using the baseline. While stereo cameras have advantages in terms of cost

and performance in bright environments, there are challenges, such as algorithmic complexity and inferior performance in low-light conditions.

Rectification is crucial in stereo cameras, affecting camera calibration and the matching of corresponding scanlines in left and right images. The provided text also touches upon the fundamental difficulty of establishing correspondence between imaged points from different viewpoints. The concept of parallax, or the displacement of image projections, is inversely proportional to distance and is employed in computing 3D geometry through triangulation.

In summary, passive stereo vision utilizes the principles of triangulation, disparity, and baseline distance to extract 3D information from images captured by a pair of cameras (Figure 11). It offers advantages in terms of cost and performance outdoors but faces challenges related to algorithmic complexity and low-light conditions. The accuracy of depth is influenced by factors like baseline distance, the resolution, and lens focal length. (Seitz, 1999)

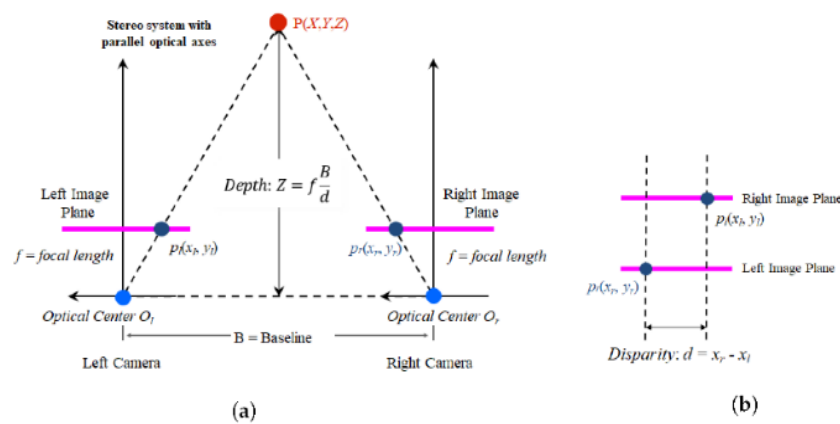


Figure 11: Illustration of a stereo vision system

3.1.3. Active stereo vision

Active stereo vision systems enhance traditional stereo vision setups by incorporating additional elements, such as projectors or structured light sources. These additions serve to simplify the stereo matching problem and improve depth perception in challenging environments. A common approach in active stereo vision involves integrating projectors into single or stereo camera systems. These projectors emit light, either in the form of structured patterns or through Time-of-Flight (ToF) technology, to aid in depth measurement and correspondence matching between camera images.

Structured light involves projecting a known light pattern, such as dots, stripes, or color-coded patterns (Figure 12), onto an object or scene using a projector, allowing the system to analyze distortions in these patterns to infer depth information.

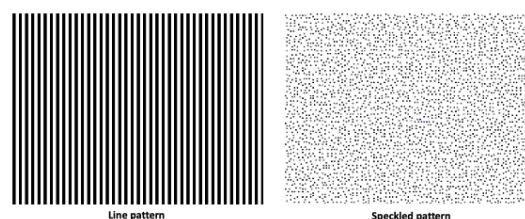


Figure 12: Structured light patterns

One notable application of structured light technology has been seen in several depth cameras. This technology revolutionized depth sensing by bringing affordable depth sensors to the mass market. Structured light sensors offer high accuracy and resolution in depth data, making them valuable for various applications such as RGB-D SLAM (Simultaneous Localization and Mapping). Despite their advantages, structured light sensors have limitations. They tend to be more expensive than stereo camera modules and require additional power to operate the projector and compute depth information. Additionally, their range is typically limited to under 5-7 meters, and they may struggle with highly reflective surfaces or outdoor environments where sunlight could overpower the projector. (Depth cameras and RGB-D camera SLAM, 2020)

Sunlight can pose significant challenges to structured light systems for several reasons, including interference from sunlight. Sunlight contains infrared light, which is similar to the light used in structured light systems. When a structured light camera operates in bright sunlight, the camera's IR sensor can pick up the IR light from the sun, which interferes with the structured light pattern. This interference can cause the camera to misinterpret the light pattern, leading to errors in depth measurements.

Another issue is reduced contrast, where the strong light from the sun can overwhelm the projector's pattern. In bright sunlight, the contrast between the structured light pattern and the background can be reduced, making it difficult for the camera to detect the subtle changes needed to measure depth accurately. This can lead to a less detailed or inaccurate depth map and expansion in the 3D reconstruction (Figure 13).

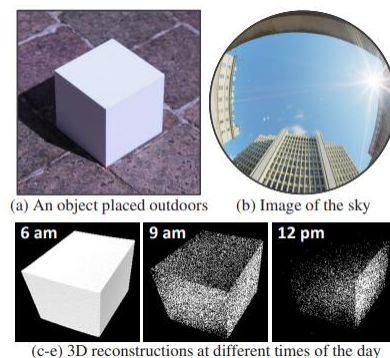


Figure 13: How 3D reconstructions affected by sunlight examples

Sensor saturation is also a problem, as the camera's IR sensor might become overloaded when exposed to direct sunlight. When the sensor receives too much light, it can't accurately detect the structured light pattern, resulting in areas of the depth image being washed out or completely blank. (Gupta, Yin, & Nayar, 2013)

In summary, active stereo vision, especially through structured light systems, provides enhanced depth perception and performance in low-light environments. However, it also introduces considerations such as cost, power consumption, and environmental limitations that must be weighed against its benefits in specific applications.

3.1.4. Passive Vs Active stereo vision

Stereoscopic vision, a fundamental principle of computer vision, mimics human binocular vision by using two or more cameras to perceive depth through triangulation. This technology is passive, requiring no active illumination, making it suitable for long-range and outdoor applications. However, stereoscopic vision struggles with flat, untextured scenes and may not provide dense point cloud representations. It relies on area-based matching and can face challenges with the matching problem.

Active stereoscopic vision builds on the principles of stereoscopic vision by incorporating an additional random pattern projector (RPP) into the system. Like stereoscopic vision, active stereoscopic vision aims to mimic human vision but offers a smaller operating range. While active stereoscopic vision is less affected by ambient light and does not require surface texture, it can face challenges in direct sunlight.

When comparing point cloud stereo vision and point cloud structured light vision, both technologies have their unique advantages and applications. Stereo cameras excel in outdoor environments and offer a passive solution for depth perception, while structured light cameras, use time-coded patterns to infer depth. The structured light approach provides high accuracy at short distances at a fair cost compared to other technologies. However, it may face limitations with transparent objects, highly reflective surfaces, or long-range applications. The choice between stereoscopic vision and structured light vision depends on the specific requirements of the application, including range, environmental conditions, and desired accuracy. (Stereo Vision for 3D Machine Vision Applications, n.d.)

3.2. Time of Flight (ToF) camera

It is also worth mentioning time-of-flight (TOF) cameras, which revolutionise 3D imaging by adopting a time-domain approach instead of a space-domain approach. These cameras, sometimes referred to as LIDAR or laser radar, measure the time delay between the emitted laser light and its reflection from the surface of an object to obtain accurate distance measurements (Figure 14).

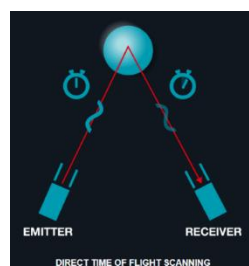


Figure 14: LiDAR distance measurement technique

There are two main techniques are used in TOF cameras.

- Pulsed laser light systems: These measure the time delay between emitted and received laser pulses, determining relative positions based on the speed of light. While offering speed, the shorter baseline can reduce depth resolution.

- Phase-based systems: These modulate a sine wave in the emitted laser beam and measure the phase difference between the emitted and reflected waves. While providing better accuracy for longer integration times, they are sensitive to ambient lighting and reflections.

TOF cameras use near-infrared light to illuminate scenes and measure the phase shift of reflected light to determine distances. They operate at high speeds, capturing entire images at video frequencies or higher. However, they typically offer lower spatial resolution, with VGA resolution being considered high-end due to manufacturing complexity.

Despite their lower theoretical accuracy compared to structured light systems at shorter distances, TOF (Time of Flight) cameras excel beyond 10 meters. They allow pixel-level processing, enhancing accuracy and avoiding loss of spatial resolution. TOF cameras represent a cutting-edge technology with tremendous potential for various applications requiring real-time 3D distance imaging.

3.3 D455 Intel RealSense

The Intel RealSense D455 depth camera, used in this research, represents the forefront of active sensor technology, offering significant advances in depth and spatial perception. Building upon the success of its predecessors, the D455 enhances range, accuracy, and versatility to meet the evolving demands of modern applications.

One of the key enhancements of the D455 is the extended distance between depth sensors, now spanning 95 mm. This extension significantly improves depth accuracy, with an impressive error rate of less than 2% at 4 meters (Figure 15). Such precision is crucial for tasks requiring reliable depth measurements, such as collision avoidance and object recognition. Additionally, the D455 provides data in real-world scale and distances, ensuring accurate and practical applications across various fields.

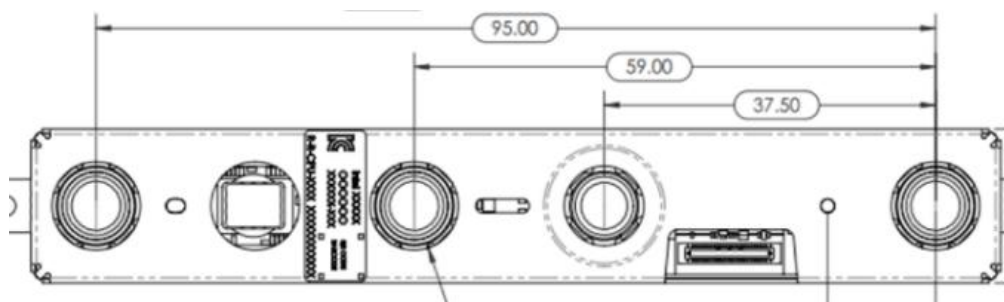


Figure 15: Intel RealSense D455 sensors distances

To complement its superior depth sensing capabilities, the D455 features an RGB sensor/imager (Figure 16) equipped with a global shutter, perfectly matched to the depth field of view. This integration ensures seamless correspondence between RGB and depth images, facilitating effortless scan recreations and enabling the creation of digital twins with unmatched precision.



Figure 16: Intel RealSense D455 camera

Moreover, the D455 incorporates an infrared projector that enhances the stereo camera system's ability to determine depth. By projecting a static infrared pattern onto the scene (Figure 17), the infrared projector increases the depth accuracy on low-texture scenes, in challenging environments.



Figure 17: D455 infrared pattern

Notably, the infrared projector meets the Class 1 laser safety standard under normal operation, ensuring user safety during use (Figure 18). In addition to depth sensing, the color sensor on the stereo depth module provides texture information, enabling various applications such as overlaying texture on a depth image to create a colorized point cloud and overlaying texture on 3D model for reconstruction.

TYPE	INJURY OR RISK	RISK LEVEL
Class 1	Low energy levels, not hazardous to skin or eyes. Safe during normal operation.	Low
Class 1M	Safe during normal operation but may cause eye injury if viewed with an optical instrument.	Low-Medium
Class 2	Visible wavelengths only, natural blink response provides eye safety.	Low-Medium
Class 2M	Visible wavelengths only, blink response provides eye safety for unaided viewing	Medium
Class 3R	Transitional zone between safe and hazardous laser products. Direct viewing of	Medium-High
Class 3B	Direct viewing and specular reflections can cause eye injury. Diffuse reflections are	High
Class 4	Can cause severe skin and eye injury through any direct exposure, specular reflections and/or sometimes from diffuse reflections.	Extreme

Figure 18: Lasers are categorized into 4 classes based on their optical power levels

Furthermore, the integration of an Inertial Measurement Unit (IMU) empowers applications to refine depth awareness, even in dynamic environments where the camera is in motion. The D455's versatility is evident in its adaptability to both indoor and outdoor environments. With its robust

global shutter technology, the camera delivers exceptional performance in various lighting conditions, ensuring reliable operation across diverse settings.

The specifications of the Intel RealSense D455 underscore its capabilities. Utilizing stereoscopic depth technology, the D455 offers a depth field of view of $87^\circ \times 58^\circ$ and outputs depth streams at resolutions of up to 1280×720 , achieving frame rates of up to 90 fps. The RGB sensor boasts a resolution of 1 MP and a field of view of $90^\circ \times 65^\circ$, capturing crisp and detailed images at 30 fps. (Figure 19). (RealSense I. , 2023)

D400 series Depth Cameras	Intel® RealSense™ Depth Camera D455 / D455f
Depth module	Intel® RealSense™ Depth module D450
Baseline	95mm
Left/Right Imagers Type	Wide
Depth FOV HD (16:9) (degrees)	H:87 / V:58 / D:95
Depth FOV VGA (4:3) (degrees)	H:75 / V:62 / D:89
IR Projector	Wide
IR Projector FOV	H:90 / V:63 / D:99
Color Sensor	OV9782
Color Camera FOV	H:90 / V:65 / D:98
IMU	6DoF
Filter	All pass/ IR-pass

Figure 19: Specifications of the Intel RealSense D455 capabilities

3.3.1. Coordinate Axes Description

The coordinate system of the D455 camera plays a crucial role in ensuring the accuracy of 3D reconstruction processes. Understanding this coordinate system is essential for the correct interpretation of the depth data produced by the camera. The D455 camera applies a specific coordinate system in which the Z-axis extends forward from the camera lens, the Y-axis is oriented downward, and the X-axis extends to the right. These axes are critical for determining the spatial relationships within the camera's field of view (Figure 20).

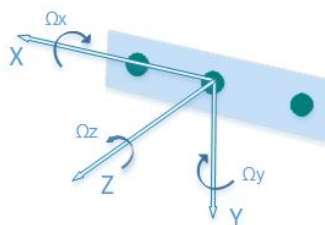


Figure 20: Coordinate system of D455

It is important to note that this coordinate system is defined from the perspective of a person standing behind the camera and looking in the direction it is facing. As a result, when the camera is viewed from the front, the left infrared sensor actually appears on the right side. This configuration is

characteristic of a left-handed coordinate system, which is a crucial factor when the data produced by the D455 camera are used in subsequent stages of analysis or processing. Proper awareness and understanding of this coordinate system are essential for fully leveraging the depth sensor capabilities of the camera in various applications. (RealSense, D435i, n.d.)

This understanding is particularly important because the data produced by the camera may require transformation due to the potential use of a different reference system in other environments where they will be further processed. Incompatibility between different reference systems can lead to errors or discrepancies in the results.

4. 3D Registration

For the process of 3D registration, preprocessing of the data exported from the depth camera is essential. RGBD images must be aligned so that each pixel in the color image corresponds to a pixel in the depth image. This alignment ensures the integrity of subsequent processes, such as the creation and alignment of point clouds.

4.1. RGBD Images to Point cloud

Each scan export from the D455 depth camera has a .bag file, which includes two folders for each frame type (color & depth) and a .json file. The JSON file provides metadata and calibration information associated with the images and depth data captured by the Intel RealSense D455 camera (Figure 21). It contains the information necessary to map the depth data to real-world distances and accurately align the color and depth images.

```
"color_format" : "RGB8",
"depth_format" : "Z16",
"depth_scale" : 999.99993896484375,
"device_name" : "Intel RealSense D455",
"fps" : 5.0,
"height" : 480,
"intrinsic_matrix" :
[
    386.3927001953125,
    0.0,
    0.0,
    0.0,
    385.97833251953125,
    0.0,
    322.4674072265625,
    247.2313232421875,
    1.0
],
"serial_number" : "146222253035",
"stream_length_usec" : 172631772,
"width" : 640
```

Figure 21: Scan metadata from .json file

An RGBD image is a composite image created by combining both color and depth information. This fusion of data allows for enhanced perception of three-dimensional environments. To achieve this, both the color and depth images must be aligned, meaning they must be registered within the same camera frame and possess identical resolutions. This alignment ensures that every pixel in the RGB image corresponds precisely to a pixel in the depth image.

The default conversion function for generating an RGBD image involves taking a pair of color and depth images as inputs. The color image in RGB format (.png) is first converted to grayscale, which

simplifies the data while preserving the luminance information. This grayscale image is then stored as a floating-point array with values normalized within the range [0, 1]. This normalization makes the data more manageable for further processing. On the other hand, the depth image (.jpg) is stored as a floating-point array representing the depth values in meters. This depth information indicates the distance from the camera to the objects in the scene at each pixel. (YodaYoda, 2020)

To calculate the three-dimensional (3D) coordinates from the depth image, it is necessary to use the intrinsic parameters of the camera. These intrinsic parameters include key values such as the focal length and the principal point (the optical center of the camera). The focal length, concerning depth image case, determines how strongly the camera converges light rays to a single point, affecting how depth is perceived. The principal point is the coordinate in the image where the optical axis intersects the image plane.

Using these intrinsic parameters, each pixel in the two-dimensional image (color or depth) can be transformed into a three-dimensional point in space. Specifically, for each pixel in the depth image, the depth value is combined with the pixel's position and the camera's intrinsic parameters to calculate its 3D coordinates in the camera's reference frame. This process allows the depth image to be converted into a 3D point cloud, where each point represents a physical location in space (Figure 22).

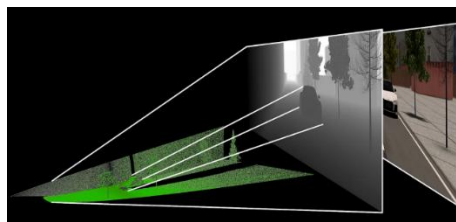


Figure 22: Point cloud calculated from depth map

To create a visually meaningful 3D representation, it's essential to colorize the point cloud. This involves combining the RGB data (color information) with the depth data. Each depth point, now represented as a 3D coordinate, is matched with its corresponding color value from the RGB image. This matching is performed using the same two-dimensional pixel coordinates (Figure 23).



Figure 23: Depth and Color frame

In essence, the color of each point in the point cloud is determined by the RGB values at the corresponding location in the color image. This results in a colored point cloud, where each point is not only positioned accurately in 3D space but also displays the correct color as seen in the original RGB image (Figure 24).



Figure 24: Colored point cloud from depth and rgb frames

Before applying any point cloud alignment techniques, it is crucial to preprocess the data. One of the primary preprocessing steps is downsampling. Downsampling reduces the number of points in the point cloud, which simplifies the dataset, reduces noise, and significantly optimizes computational efficiency (Figure 25). This step is especially important when dealing with large-scale point clouds, as it makes subsequent processing, such as alignment and registration, much faster and less prone to errors.

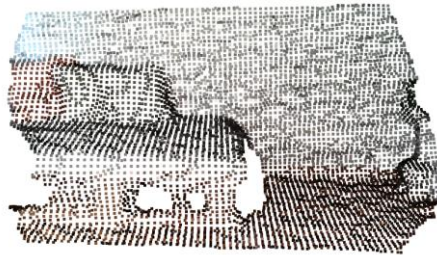


Figure 25: DownSampled colored point cloud

Down sampling typically involves selecting a subset of points based on a defined sampling criterion, such as voxel grid filtering, where the space is divided into a 3D grid and a representative point is chosen for each grid cell. This reduces the overall number of points while retaining the essential structure of the point cloud. Once down sampling is complete, point cloud alignment can be performed more effectively. Alignment is the process of registering multiple point clouds into a single coherent model, which is critical in applications such as 3D reconstruction, where multiple views of an object or scene are merged to create a complete 3D model.

4.2. Local refinement

Local refinement aims to achieve precise alignment between two point clouds by iteratively minimizing discrepancies and ensuring an accurate match of their geometric structures. This step is critical in pairwise registration, as it refines the initial rough alignment and prepares the data for more complex operations, such as multiway registration or global optimization.

The refinement process relies on advanced algorithms like ICP (Iterative Closest Point) and its variations, which introduce enhancements to address specific challenges in point cloud alignment. These methods optimize the transformation parameters (rotation and translation) to achieve accurate alignment while preserving the spatial relationships and geometry of the point clouds.

4.2.1. ICP (Iterative Closest Point) Point-to-Plane

The ICP (Iterative Closest Point) algorithm is a method used to align two 3D point clouds by iteratively minimizing the distances between them. This involves finding a transformation consisting of a rotation matrix and a translation vector to align one point cloud (source) with another (target). Through repeated iterations, the algorithm refines this transformation until the point clouds are sufficiently aligned, minimizing the error. Is a local registration algorithm that minimizes the distances between two 3D point clouds to align them. This is achieved by calculating the parameters of a rigid body transformation, represented by a 4x4 homogeneous matrix H , specifically a rotation matrix R and a translation vector t , preserving their shape and scale (Figure 26). This transformation is applied to one point cloud (source) to bring it into the same reference frame as another point cloud (target) so that their points correspond to the same spatial coordinates. (Ho, n.d.)

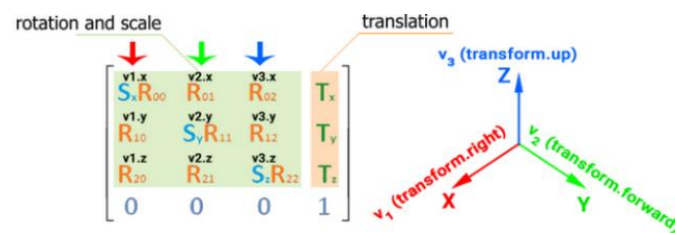


Figure 26: Transformation matrix (Matrix Transformations, n.d.)

The ICP point-to-plane2 variant enhances this process by not only minimizing the Euclidean distance between corresponding points but also considering the surface geometry of the target point cloud. Instead of aligning a point from the source to a point in the target, this method aligns the point from the source to the tangent plane of the corresponding point in the target (Figure 27). This approach generally converges faster and can be more accurate, as it incorporates the local geometry and reorientation of the surface, resulting in better matching between cloud points. (Low, 2004)

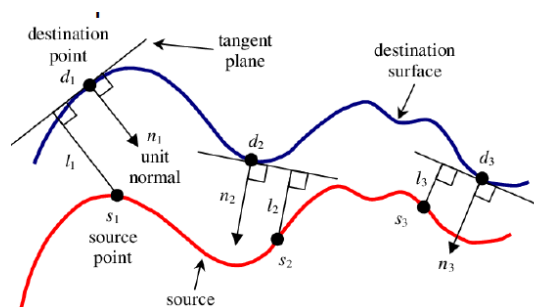


Figure 27: Point to Plane method

The ICP point-to-plane algorithm is particularly useful when point clouds represent complex surfaces. In the context of pairwise registration or multiway registration in 3D, the process begins with an initial alignment of the source and target point clouds by minimizing the distances between points and planes. Moreover, the use of a pose graph ensures that each new point cloud is aligned with the

² https://www.open3d.org/docs/release/tutorial/pipelines/icp_registration.html#Point-to-plane-ICP

set of previously registered and aligned point clouds. This maintains the overall accuracy and consistency of the combined dataset during the integration of new point clouds.

4.2.2. Colored ICP

Colored ICP³ (Iterative Closest Point) is an advanced variation of the traditional ICP algorithm used for aligning 3D point clouds. Unlike traditional ICP, which relies solely on geometric information (the positions of points in space), Colored ICP incorporates color information (RGB values) to enhance alignment accuracy. This method is particularly useful for complex scenes and large datasets, ensuring high precision. (Park, Zhou, & Koltun, 2017)

In each iteration of the Colored ICP algorithm, pairs of corresponding points between the source and target point clouds are identified. This involves finding the closest points based on both spatial distance and color similarity. The color values (red, green, and blue) of each point are utilized to help determine correspondences. Points with similar colors are considered more likely to be correct matches. By incorporating color information, the algorithm can better distinguish between points that are geometrically close but have different colors, overcoming challenges posed by repetitive or ambiguous geometry.

Once the corresponding points are identified, the algorithm refines the rigid transformation matrix T , which consists of a rotation matrix R and a translation vector t , to align the source point cloud to the target point cloud. This process iteratively reduces the misalignment by optimizing a joint energy function, which combines geometric and photometric consistency.

The Colored ICP algorithm achieves this by minimizing a joint energy function composed of two terms. The first term, the geometric term $E_G(T)$ (Figure 28), measures the point-to-plane error and ensures alignment based on spatial geometry. It minimizes the perpendicular distance between the transformed points in the source point cloud and the tangent planes at their corresponding points in the target point cloud.

$$E_G(\mathbf{T}) = \sum_{(\mathbf{p}, \mathbf{q}) \in \mathcal{K}} ((\mathbf{p} - \mathbf{T}\mathbf{q}) \cdot \mathbf{n}_p)^2,$$

Figure 28: Colored ICP geometric term

The second term, the color term $E_C(T)$ (Figure 29), measures the photometric error and ensures alignment based on color similarity, minimizing the differences in the RGB values of corresponding points.

$$E_C(\mathbf{T}) = \sum_{(\mathbf{p}, \mathbf{q}) \in \mathcal{K}} (C_p(\mathbf{f}(\mathbf{T}\mathbf{q})) - C(\mathbf{q}))^2,$$

Figure 29: Colored ICP color term

A parameter λ (Figure 30) is used to balance the contributions of the geometric and color terms in the energy function. When λ is closer to 1, the algorithm prioritizes geometric alignment,

³ https://www.open3d.org/docs/release/tutorial/pipelines/colored_pointcloud_registration.html

whereas lower values of λ place more emphasis on photometric consistency. This balance allows the Colored ICP algorithm to adapt flexibly to different scenarios, ensuring that both geometry and color are utilized effectively for robust and accurate alignment. (Jaesik, Park, Zhou, & Koltun, 2017)

$$\lambda E_G + (1 - \lambda) E_C$$

Figure 30: Colored ICP balance term

4.2.3. Pose Graph

The pose graph is a critical component in 3D reconstruction for optimizing point cloud alignment. It enables the integration of new point clouds into an already aligned dataset, ensuring accuracy and consistency. This methodology is particularly valuable for large-scale reconstructions with multiple views.

4.2.3.1. Initialization of Pose Graph

The first node in the pose graph typically represents the initial position of the sensor and is initialized with the identity matrix. This matrix indicates that the robot's starting position is at the origin of the coordinate system, with no rotation or translation applied. This initial pose serves as a fixed reference point against which all subsequent poses will be compared and connected within the graph. Once initialized, this first node is added to the pose graph as the starting point, and initially, it does not have any edges connected to it because it is the only pose in the graph at that moment.

As new nodes are added, if a neighboring node exists, an odometry edge is created, using a transformation matrix computed from the application of ICP (Iterative Closest Point) between the new node and the neighboring node. If there are non-neighboring nodes that might have overlaps, a loop closure edge is added, with the transformation matrix computed from pairwise registration between the new node and the non-neighboring node. (Grisetti, Kümmerle, Stachniss, & Burgard, 2010)

4.2.3.2. Pose Graph Construction

Spatial relationships and transformations between different poses (positions, orientations) of a sensor with respect to time are represented by the pose graph⁴. A pose graph represents the spatial relationships and transformations between different poses (positions and orientations) of a sensor over time. Each node in the pose graph represents a piece of geometry, specifically a point cloud, and has an associated pose matrix. This matrix transforms the geometry from the node's local coordinate system to the global coordinate system. The first node, or the first point cloud in this case, serves as the reference point and corresponds to the identity matrix. The remaining nodes are initialized based on transformations from neighboring nodes. (Choi, Zhou, & Koltun, 2015)

Edges connect pairs of nodes, indicating spatial measurements. Odometry provides a rough initial guess of the relative pose between consecutive frames, based on the assumption that the movement between consecutive frames is small and can be estimated. The edges in the pose graph connect two nodes and contain a transformation matrix that aligns the geometry of the source node with the geometry of the target node (Figure 31). The edges are divided into two categories:

⁴ https://www.open3d.org/docs/release/tutorial/pipelines/multiway_registration.html#Pose-graph

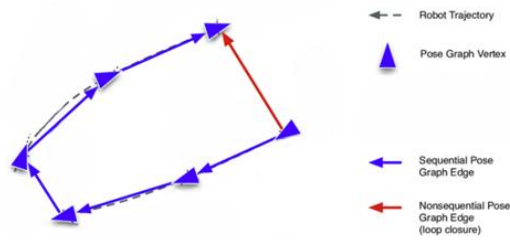


Figure 31: Pose Graph Construction example (Pfungsthorn, 2014)

- Odometry Edges:** As the sensor acquires new poses, a new node is added to the pose graph. If there is a neighboring node, meaning a pose that is temporally or spatially close, an odometry edge is created between the new node and the neighboring node. This edge represents the estimated transformation between the two poses, derived from the sensor's odometry data. The transformation matrix defining this edge is computed using the Iterative Closest Point (ICP) algorithm, which aligns two point clouds by minimizing their differences. Odometry edges connect temporally close nodes, providing a reliable initial estimate of the relative pose between consecutive frames due to the high overlap ratio between the corresponding point clouds. (Iterative Closest Point).
- Loop Closure Edges:** When the sensor revisits a previously explored area, non-neighboring nodes, nodes that are not temporally or spatially adjacent but represent the same physical location, can be identified. This identification is crucial for correcting accumulated drift in pose estimates by recognizing when the sensor has returned to a previously visited location. When a potential overlap is detected, a loop closure edge is added between the new node and the non-neighboring node. This edge, similar to an odometry edge, contains a transformation matrix computed through pairwise registration, which may involve more robust methods than ICP, such as RANSAC-based alignment or global optimization techniques. While loop closure edges are essential for maintaining global consistency in the pose graph, they are inherently less reliable than odometry edges due to the lower overlap ratio between the point clouds being aligned. Nonetheless, these edges are crucial for correcting errors accumulated over long trajectories in the 3D reconstruction process. (Simsangcheol, 2023)

At the end of the process and before the results are extracted, the pose graph is optimized using the Levenberg-Marquardt method. It further involves a global optimization algorithm to improve the poses of the nodes. The optimization aims to minimize the error on all edges of the pose graph globally and not individually as in the previous methods. Levenberg-Marquardt is an iterative method that combines the concepts of gradient descent and Gauss-Newton methods. It is particularly effective for nonlinear least squares problems. The algorithm adjusts the positions iteratively to minimize the sum of squared errors. Levenberg-Marquardt algorithm is known for its robustness and its ability to converge to a good solution even when the initial estimates are far from the optimal solution. (Henri, 2024)

5. Novel 3D reconstruction approaches

3D reconstruction has made significant progress in recent years, with new methods being developed to overcome the limitations of traditional approaches. Classic techniques, such as voxel grids and polygonal meshes, often struggle to balance quality, computational efficiency, and memory

usage. These methods require large amounts of memory to store high-resolution data and frequently fail to accurately render complex geometric details or lighting effects in intricate scenes.

To address these challenges, advanced methods such as Neural Radiance Fields (NeRF) and Signed Distance Fields (SDF) have been developed. These techniques leverage deep learning and mathematical models to achieve detailed and efficient 3D reconstructions.

5.1. NeRF

Traditional 3D scene reconstruction and rendering methods often struggle to balance quality and computational efficiency. Voxel grids require significant memory to store high-resolution details, while mesh-based methods may not accurately represent complex geometries and view-dependent lighting effects. Neural Radiance Fields (NeRF) overcome these challenges by representing a scene as a neural radiance field, a continuous function parameterized by a neural network. This innovative technique in computer vision and graphics enables the production of high-resolution, photorealistic 3D renderings from a limited number of 2D images (Figure 32).

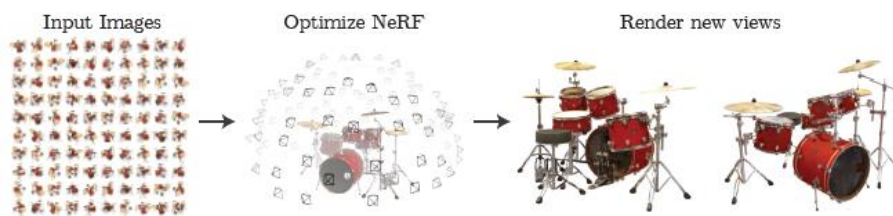


Figure 32: High-resolution 3d rendering from 2D images

Unlike traditional methods that rely on voxel grids or meshes, NeRF employs a continuous volumetric scene representation enhanced by deep learning techniques. Compared to standard methods using voxel grids or meshes, NeRF works with a continuous point density field supported by deep learning methods. This allows NeRF to form new views of complex scenes, such as scenes with complex geometry and lighting details, generated with high photorealism while maintaining reasonable runtimes.

To accurately reconstruct the 3D scene, NeRF requires overlapping images of the scene and corresponding camera poses. After capturing these images, the camera poses are commonly estimated. Once the camera poses and images are aligned, NeRF can effectively render the scene by learning how light interacts with the environment from different perspectives, ultimately producing highly realistic 3D visualizations.

5.1.1. Neural Network Architecture of NeRF

NeRF (Neural Radiance Fields) encodes a scene by using a continuous 5D function to output the color and density of a point based on its position and the direction it is viewed from. This method relies on 3D spatial coordinates (x, y, z) to represent the location within the scene and 2D angles (θ, ϕ) to define the viewing direction. This complex relationship is captured by a Multilayer Perceptron (MLP), a type of deep neural network.

The NeRF (Neural Radiance Fields) architecture uses a Multi-Layer Perceptron (MLP) to process 5D inputs, which consist of a 3D spatial location and 2D viewing direction, with the goal of predicting the color (RGB) and density (σ) for each point in a scene.

In this process, the 5D inputs are passed through 8 fully connected layers, each followed by ReLU activation functions, enabling the model to learn complex patterns and relationships, effectively mapping the inputs into a high-dimensional feature space. After these layers, the model outputs two key elements: the density (σ) of the point, which indicates how much light is absorbed or scattered at that point, and a 256-dimensional feature vector that captures rich details about the point in the scene. This 256-dimensional feature vector is then combined with the viewing direction of the camera ray, and the combined information is passed through one more fully connected layer with ReLU activation and 128 channels, which finally outputs the view-dependent RGB color for that point (Figure 34). This entire process allows the NeRF model to reconstruct scenes from different viewpoints by predicting how each point in the scene should look based on its position, viewing direction, and learned features.

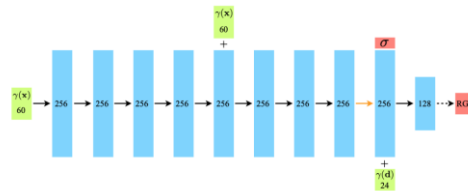


Figure 33: Multi-Layer Perceptron (MLP) architecture commonly used in NeRF (Neural Radiance Fields)

The network produces two essential outputs: volume density, which indicates how opaque a material is at a specific point in 3D space, and a feature vector (Figure 34). This feature vector is then integrated with the viewing direction to generate the RGB color, which changes based on the observation angle. This approach allows the MLP to capture intricate scene details, including effects like reflections, refractions, and shading, leading to highly realistic 3D renderings with detailed lighting and fine textures.

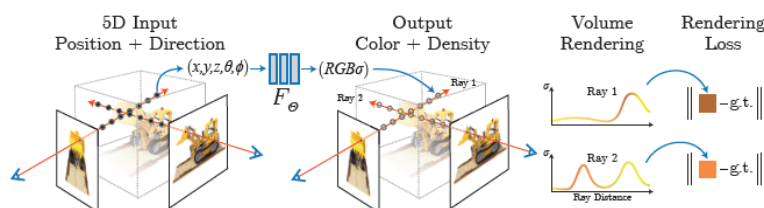


Figure 34: Core concepts behind Neural Radiance Fields (NeRF)

The MLP in NeRF effectively transforms 5D input coordinates into high-dimensional feature representations via multiple layers and non-linear activations. This allows the network to learn and represent complex interactions between 3D spatial latent locations in relation to view angles, leading to the extraction of rich lighting effects, fine details, or other scene-specific information. During training, the network's weights are optimized using backpropagation and gradient descent to improve predictions of scene details. As a result, NeRF can produce extremely detailed and photorealistic 3D renderings from a limited number of camera views.

5.1.1.1. Multilayer Perceptron (MLP)

In machine learning, a Multilayer Perceptron (MLP) is the foundational model of an artificial neural network, consisting of multiple layers. The MLP is a network made up of layers of nodes, called neurons, which are the basic units for processing and transmitting information from one layer to another. Each neuron in the network receives input data, applies a mathematical operation (typically a weighted sum followed by an activation function) on this input, and then passes it forward to neurons in the next layer. Weights connect neurons in different layers, and adjusting these weights during training optimizes the network's performance.

An MLP generally has three types of layers. The input layer, one or more hidden layers, and an output layer (Figure 35). The input layer receives raw data, such as images or text sequences, and feeds it into the network. Learning occurs in the hidden layers between the input and output layers, where information is gradually refined into a form that the output layer can use to produce the final result, which could be a classification label, numerical value, or another type of output depending on the specific task.

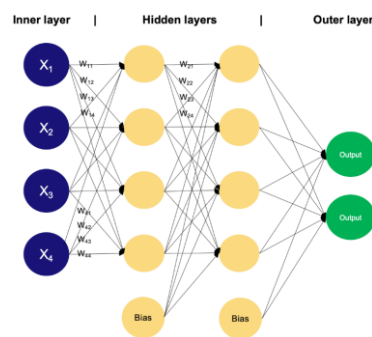


Figure 35: Example of an MLP with two hidden layers

The use of activation functions in MLPs adds non-linearity, which is crucial for learning complex patterns. These complexities are captured by modulating or transforming the passing information using activation functions such as the Rectified Linear Unit (ReLU), Sigmoid, and Tanh. Without these activation functions, an MLP would essentially be a linear model, unable to solve complex problems.

MLP training involves a supervised learning process known as backpropagation, which aims to minimize the difference between the network's predictions and target values. During backpropagation, the network determines its error and adjusts each weight based on this feedback to avoid repeating similar mistakes. The network is updated iteratively over millions of records, reducing error gradually. Optimization algorithms like Stochastic Gradient Descent (SGD) or Adam are commonly used to adjust the weights efficiently during training. (Sejal, 2024)

5.1.1.2. Volume Rendering

Volume rendering is a technique used to produce 2D images from 3D scenes by simulating how light interacts with opaque materials. Neural Radiance Fields (NeRF) enhance this process by using a 5D function that captures essential information about a scene: volume density (σ) and radiance (or color). Volume density indicates the probability of light being absorbed or scattered in a certain space, while radiance represents the color emitted from a point based on the viewing direction.

Radiance fields are valuable in NeRF because they depict intricate light behaviors in a scene, crucial for creating photorealistic images. These fields visualize how light behaves as it passes through different materials and interacts with surfaces, capturing effects like reflections and refractions. By encoding not just the color at a point but also how that color changes based on the viewing angle, radiance fields enable NeRF to generate realistic object appearances under varying lighting conditions and perspectives.

To render an image, NeRF simulates light rays passing through each pixel of the image. Each ray $r(t)$ is emitted from the camera origin o , through the pixel $C(r)$, and into the 3D scene. As the ray traverses the scene, NeRF samples multiple points along its path. For each of these points, the neural network predicts the volume density and the radiance (Figure 36). For example, when a ray intersects an object the volume density is high, indicating that the object absorbs or scatters light significantly. The radiance at this point reflects the color as seen from the specific viewing direction of the camera.

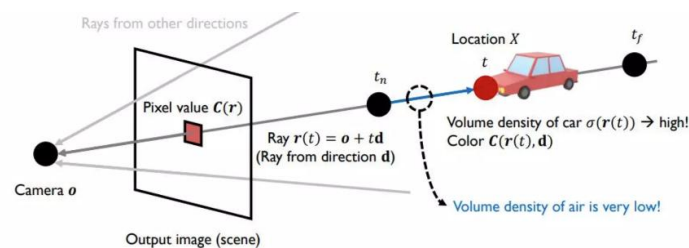


Figure 36: Illustrates of the concept of Volume Rendering

The outputs are radiance and density values from all sampled points along the ray, which the neural network combines to compute the final pixel color. This process involves aggregating contributions from every point along the ray path, considering how light is absorbed and scattered in between. The transmittance function calculates the probability that a ray will continue traveling without being blocked by particles. During color accumulation, all sampling points are summed with weights based on their transmittance values to produce the final pixel color.

Radiance fields enhance this process by allowing NeRF to model nuanced lighting changes and details, such as light filtering through a translucent material or reflecting sharply off a glossy surface. This capability is especially important in scenes with complex lighting effects, where traditional rendering methods might struggle to achieve the same realism. NeRF uses stratified sampling to estimate the integral representing the total color for each ray. This technique involves dividing the ray's path into multiple bins and sampling from each bin, ensuring that the entire range of the ray is well represented for a more accurate approximation.

Stratified sampling helps NeRF approximate a continuous scene representation by evaluating the scene at various locations during optimization, contributing to high-quality visual effects like reflections, refractions, and realistic shading. (Wu, Lee, Bhattad, Wang, & Forsyth, 2022)

5.1.1.3. Neural Radiance Field Optimization

The techniques used to optimize a Neural Radiance Field (NeRF) aim to render high-resolution and complex scenes with realistic quality. While the core elements of NeRF, such as scene modelling and rendering new views, are fundamental to the process, they alone are not sufficient to achieve the

best possible results. For this reason, two significant improvements are introduced: positional encoding and hierarchical volume sampling.

5.1.1.3.1. Positional Encoding

While training a neural network like the MLP in NeRF, it's important to understand that these networks can theoretically learn any function, given enough data and time. However, when the network directly processes raw 3D coordinates and 2D viewing angles, it often struggles to capture high-frequency details in the scene. These high-frequency details include sharp edges, intricate textures, and fine lighting variations.

Neural networks, particularly deep ones, naturally find it easier to learn general, low-frequency details than more intricate, high-frequency details. As a result, they may overlook the small, detailed aspects of a scene. To address this problem, there is a modification to the way the network handles the input coordinates. Instead of using the raw coordinates directly, first map them into a higher-dimensional space using sine and cosine functions with different frequencies, a technique known as positional encoding (Figure 37). This transformation makes each of the spatial coordinates (x, y, z) and each component of the viewing direction more complex and detailed. This allows the MLP to better approximate fine details in the scene, improving its ability to render sharper textures and edges more realistically. (Mildenhall, et al., 2022)



Figure 37: Positional Encoding Technique

5.1.1.3.2. Hierarchical Volume Sampling

Hierarchical Volume Sampling is a crucial technique used in NeRF to improve the efficiency and accuracy of rendering 3D scenes. When NeRF sends rays into a scene to sample points along each ray, it initially uses equal intervals between points. This basic approach ensures that the entire scene is covered but can be inefficient, especially in areas where there is no meaningful information, such as empty spaces or occluded regions. These regions do not contribute much to the final rendered image, leading to unnecessary computations and longer rendering times.

To address this inefficiency, Hierarchical Volume Sampling dynamically adjusts the sampling process when rays encounter objects or areas of interest within the scene. Instead of sampling points at fixed distances, this technique reduces the distance between sampled points in critical regions where important details, such as edges, textures, and lighting variations, are likely to be present (Figure 38). This finer sampling ensures that these complex features are captured more accurately, resulting in higher-quality renderings. (Skann.ai, 2023)

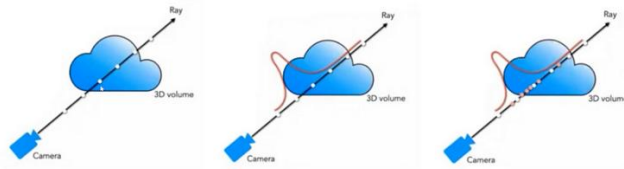


Figure 38: Hierarchical Volume Sampling Technique

The process works by using two neural networks in tandem. A "coarse" network and a "fine" network. The coarse network first evaluates a broad set of points using stratified sampling to create a rough estimate of the scene's structure. Based on this rough estimate, the fine network then focuses on a more targeted set of points, prioritizing those that are likely to contain significant details. This hierarchical approach allocates more computational resources to regions that contribute the most to the final image, thereby enhancing both the efficiency and the accuracy of the rendering process. (Mildenhall, et al., 2022)

5.1.1.3.3. Implementation details

In NeRF, every 3D scene is represented by an individual neural network that needs to be trained independently for each new view needed to generate. The training process requires a dataset of captured RGB images of the scene, along with information about camera position (camera poses) and intrinsics like focal length and sensor size. For real-world scenes, these camera details are often estimated using tools like COLMAP, a structure-from-motion (SFM) package, to recover the scene from the captured images.

During training, the process is divided into several steps. In each step, a randomly selected batch of camera rays is taken. A camera ray is essentially a straight line extending from the camera into the scene. The network samples point along these rays to determine what should be visible at each point in the final rendered image. As the network processes these points, it uses volume rendering techniques to determine the color of each ray by combining information from all sampled points to figure out the color of each pixel in the final image.

The goal of training is to make the rendered images look as close as possible to the actual images in the dataset. This is achieved by adjusting the network weights to closely match real-life images, a process guided by a loss function that quantifies the accuracy of the network's predictions. As NeRF continuously refines the network weights, it becomes increasingly proficient at generating photorealistic images, capturing fine details like textures and lighting effects with great accuracy. This careful balance of sampling, rendering, and optimization processes allows NeRF to produce realistic images from 2D data.

5.2. SDF (Signed Distance Fields)

Signed Distance Fields (SDFs) are a mathematical representation of shapes and are often used in ray marching and implicit rendering techniques. An SDF is a function that, given a particular point in space, will provide the value of the shortest distance from the nearest surface. The "signed" part means that this distance can be positive, negative, or equal to zero. A positive value of the symbol indicates that it is within the surface and a value equal to zero indicates a certainty that it is exactly

on the surface. This concept is very important in rendering techniques such as modeling, ray tracing, illumination and others.

For example, for a spherical shape with a center at the origin and a unit radius, the SDF function is defined accordingly (Figure 39).

$$f(x, y, z) = \sqrt{x^2 + y^2 + z^2} - 1$$

Figure 39: SDF function

In this case, points inside the sphere will yield a negative distance, points on the surface will return zero, and points outside the sphere will produce a positive distance (Figure 40).

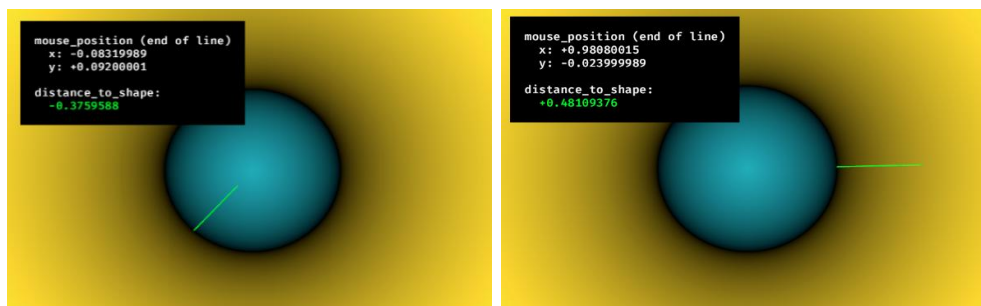


Figure 40: Negative distances inside the sphere (Left) and positive distances outside (Right).

SDFs utilize fundamental principles and tools, extending their capabilities to complex and advanced applications such as Ray Marching, surface normal calculations, and Constructive Solid Geometry (CSG). These techniques enable the efficient representation and rendering of intricate geometries in computer graphics. (Chris, 2023)

5.2.1. Ray Marching & SDFs

Ray Marching is a rendering technique that leverages Signed Distance Fields (SDFs) for efficient visualization of 3D scenes. This approach involves casting rays from a camera into a scene and using the SDF to determine the distance from the ray's current position to the nearest surface at each step (Figure 41). By advancing the ray by this precise distance, the algorithm guarantees that it will not miss any surface, making it both efficient and accurate for scenes described by SDFs.

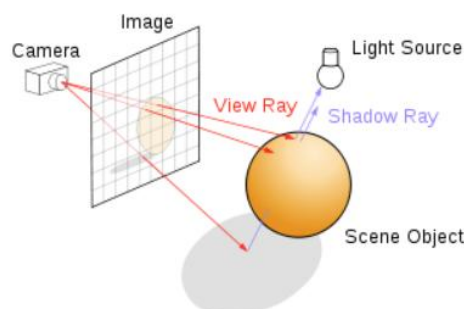


Figure 41: Rays from the camera use SDFs to compute the distance to the nearest surface. Ray Marching differs from traditional ray tracing in that it calculates intersections using Signed Distance Functions (SDFs) to implicitly define surfaces.

rather than relying on geometric primitives like triangles or spheres. An implicit surface is defined by an equation such as $f(x,y,z)=0$, where x , y , and z are the coordinates of points in 3D space. Unlike surfaces represented by parametric or geometric elements like polygonal meshes or Bezier surfaces, implicit surfaces use functions to describe scalar fields at every point in space.

Implicit surfaces are particularly effective for representing continuous and deformable objects. They allow for high geometric complexity and can be smoothly deformed by adjusting the functions' parameters, making them ideal for organic shapes and intricate designs. However, computing implicit surfaces can be demanding, especially in complex scenes with many control points. Despite this, implicit surfaces are widely used in fields such as scientific visualization, video game development, image processing, and other applications where smooth, deformable surfaces are essential. (Bourke, 1997)

Ray Marching begins by firing a ray from the camera's origin and moving it through space along a defined direction. At each step, the SDF is evaluated at the ray's current position to determine the distance to the nearest surface. The ray is then moved forward by this distance. If the SDF returns a value close to zero, the ray is considered to have hit a surface, and the algorithm stops. Otherwise, the ray continues marching until it either hits the surface or reaches the maximum number of steps.

A key optimization in Ray Marching is sphere tracing, which allows the ray to take the largest possible step without crossing any surfaces, as determined by the SDF. This approach minimizes the number of steps needed to reach a surface, improving performance. Sphere tracing also reduces computational load by enabling the ray to traverse large empty spaces quickly and slowing down as it approaches the surface (Figure 42).

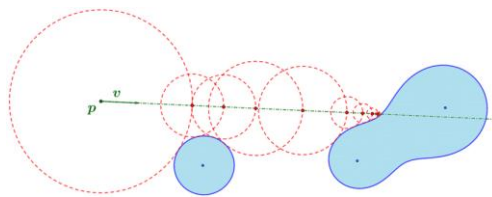


Figure 42: Sphere Tracing for Optimizing Ray Marching

Despite its efficiency, Ray Marching does face challenges when dealing with complex scenes or when the precision of the SDF is low. Inaccuracies in the SDF can lead to smaller than necessary steps, slowing down the algorithm. Additionally, complex intersections between objects can create scenarios where the ray overshoots or underestimates distances, requiring careful handling of SDF approximations to maintain accuracy and convergence. (Wong, 2016)

5.2.2. Surface Normals and Lighting

Surface normals are critical in computer graphics for calculating lighting and shading effects, as they represent the direction perpendicular to a surface at any given point. In the context of Signed Distance Functions (SDFs), calculating surface normals involves approximating the gradient of the distance field. The gradient at a point on the surface indicates the direction in which the SDF increases most rapidly, and by normalizing this gradient, the surface normal is obtained.

In an SDF, the surface normal on the surface can be estimated numerically by sampling the SDF at slightly offset points around the surface location. This measures how the distance changes as

α point is shifted slightly in various directions (x, y, z). The collection of these minor changes forms the "differential" which when normalized (adjusted to have a length of 1) provides the surface normal.

An important aspect of ensuring the accuracy and smoothness of these surface normals is the *Eikonal Loss*. This loss function plays a key role in controlling how smoothly the surface normals are calculated by enforcing a specific constraint on the gradient of the SDF. Specifically, it ensures that the magnitude of the gradient is always equal to 1 (Figure 43). By doing so, it helps the model maintain consistent surface normals, resulting in smoother, more accurate surfaces in 3D reconstructions. Without this loss, the surface normals could be irregular or noisy, leading to poor lighting and shading effects. (Yang, Sun, Sundaramoorthi, & Yezzi, 2023)

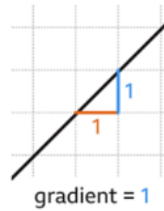


Figure 43: Visualization of a gradient with a value of 1

This normal vector is then used in lighting calculations, which are typically based on models such as Phong shading or Blinn-Phong reflection, to determine how light interacts with the surface. The dot product between the surface normal and the light source direction determines the diffuse lighting, while the angle between the normal and the viewer's perspective affects the specular highlights (Figure 44).

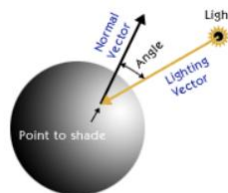


Figure 44: Normal vector used in lighting calculations

One of the main advantages of using SDFs in lighting calculations is that they provide a smooth and continuous representation of surfaces, leading to more accurate normal estimations even in complex or procedurally generated shapes. Unlike polygonal models, which approximate surfaces with flat faces and discrete normals at vertices, SDFs implicitly define surfaces, resulting in more natural lighting transitions. (Wong, 2016)

5.2.3. Constructive Solid Geometry (CSG)

Constructive Solid Geometry (CSG), a way of creating complex 3D objects by combining simpler geometric blocks via functions like union (\cup), intersection (\cap) and difference ($-$) (Figure 45). In the scenario of an SDF, these operations are condensed to some simple math adjustments in the distance value of the SDFs, enabling highly flexible and efficient shape composition.

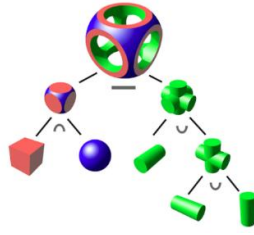


Figure 45: Creating complex 3D objects by combining simpler geometric

The union of SDFs for two objects creates a combined shape by joining the surfaces of both objects. This can be useful for producing objects held in place by multiple surfaces, but it also has the same limitations as union, where it may introduce errors at areas where two shapes meet. SDF generation may not be as accurate if the surfaces do not nicely intersect with one another.

The intersection between two objects involves taking the pointwise maximum of their distance functions. The resulting SDF describes the overlap between these two objects—where, for any given point in space, it will be considered inside this shape only if it lies within both shapes. This operation is helpful for forming capsulators bound to two areas, yet like union, it can create errors where both shapes meet. The accuracy of the generated SDF depends on how smoothly the surfaces intersect.

A difference operation subtracts one object from another by taking the maximum of one shape and the negative of the other shape's distance. This carves out the second shape from the first one, leaving behind complex geometry. The subtraction operation is useful for creating objects with voids or hollowed-out shapes. However, like union and intersection, the generated SDF may be inexact, particularly at the boundaries of objects.

CSG operations on SDFs are computationally efficient because they translate complex Boolean operations into simple minimum-maximum calculations. This method is very flexible and efficient, capable of quickly generating complex geometries and dynamic shapes suitable for real-time applications. For instance, CSG lends itself to procedural modeling, where complex structures can be constructed from simple shapes assembled hierarchically with Boolean set operators.

One of the major issues in CSG with SDFs is maintaining an accurate enough field after several operations. The resulting SDF may not exactly be a "signed distance function" which could slow down rendering algorithms like sphere tracing. More recent research has focused on bounding these inaccuracies to ensure that convergence is maintained for complex CSG operations in the context of sphere tracing. Specific distance function estimates (SDFEs) are introduced to effectively handle more complex geometries in an approximate but reasonably accurate manner. (Bálint, Valasek, & Gergo, 2023)

6. Applications of NeRF and SDF in 3D Reconstruction

3D reconstruction has made significant progress, introducing new tools that simplify the use of complex techniques. NeRF Studio and SDFStudio are two frameworks that utilize Neural Radiance Fields (NeRF) and Signed Distance Fields (SDF) to create efficient and detailed 3D representations. These tools are flexible and user-friendly, with features that address various needs, such as photorealistic rendering and surface reconstruction.

6.1. NeRF-studio

NeRF studio⁵ is an easy-to-use framework that helps develop and deploy Neural Radiance Fields (NeRFs). It was created by students at the KAIR lab in Berkeley AI Research (BAIR) and launched in October 2022. Now, it has grown into a robust open-source project supported by both researchers and community contributors.

NeRF studio's design includes several key parts, each essential to the NeRF process (Figure 46). These parts can be easily adapted to different needs. It provides an easy-to-use environment necessary for creating, training and testing NeRFs. In addition, NeRFstudio also consists of an API to make it even more flexible for developers. (Tancik, Weber, Ng, Li, & Yi, Nerfstudio, 2023)

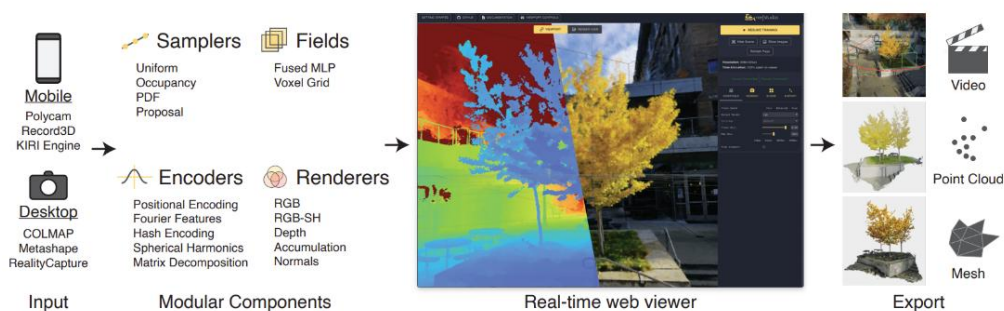


Figure 46: Overview of NeRF studio's Modular Framework

6.1.2. NeRF studio's Core Components

NeRF studio's architecture consists of several core components that aid the NeRF pipeline. These elements are designed to be modular, making it easier to customize and extend them.

6.1.2.1. Data Management and Parsing

The first critical component in the NeRFstudio framework is the DataManager, which is responsible for converting posed images into Ray Bundles. Ray Bundles are essentially slices of 3D space originating from the camera. The process begins with the DataParser, a subcomponent designed to load input images and camera data. NeRFstudio supports a variety of data formats. This includes mobile apps like Record3D, Polycam, KIRI Engine, and 3D tools such as Metashape and Reality Capture. This wide compatibility makes the framework accessible to a broader audience. Once the images and data are loaded, the DataManager processes them to generate Ray Bundles and ground truth supervision, and it can also optimize camera poses during training. (Tancik, et al., Nerfstudio: A Modular Framework for Neural Radiance Field Development, 2023)

6.1.2.2. Ray Bundles, Ray Samples, and Frustums

For 3D rendering and Neural Radiance Fields (NeRFs), Ray Bundles are a core tool for describing 3D space. These bundles encapsulate how light interacts with objects and scenes as recorded through various camera views. Ray Bundles, which consist of rays with similar directions but minimal spread, harmoniously represent a slice of 3D space (Figure 47). While each ray within a bundle

⁵ <https://docs.nerf.studio/>

has a specific direction, the entire complex of rays can be seen as spreading over multiple directions. This allows for a more robust sampling of light interactions in the scene, and the high coherence within a Ray Bundle ensures that the rays behave more like synchronized vectors.

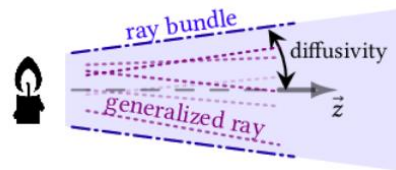


Figure 47: A ray bundle, in which the rays have a large spread of propagation direction

Each Ray Bundle is characterized by several parameters such as the starting point (origin) of each ray within the bundle, typically corresponding to the position of the camera or light source, and a vector indicating the path along which the ray travels from its origin. This direction is crucial as it defines how the ray intersects with objects in the scene. (Steinberg, et al., 2023)

Ray Samples are the result of converting Ray Bundles by sampling the 3D space along the paths defined by the rays. This sampling process is determined by the interval bin spacing, which sets the distance between consecutive samples along a ray, affecting both the resolution and detail of the samples. Smaller intervals lead to higher resolution and more detailed sampling but increase computational complexity and processing time. Ray Samples are further encapsulated into geometric shapes known as frustums. A frustum in this context is a segment of the Ray Sample that captures the 3D space it represents. Frustums can be represented as point samples or Gaussian distributions (Figure 48). Representing frustums as point samples means that each frustum of a ray is treated as an individual point in space. (Relja & Andrew, 2021)

On the other hand, representing frustums with Gaussian distributions (Gaussians) is a more advanced method that allows for the accurate capture of spatial variations of a Ray Sample, enabling the sample to describe not just a point but also the distribution around it, thereby capturing more detailed information about the 3D space. (Kate, 2023)

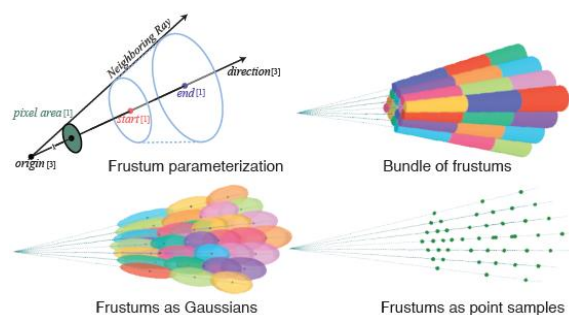


Figure 48: Visualization of Frustum Parameterization and Sampling Techniques in 3D Volume Rendering 6.1.2.3. Models & Fields

The models supported by NeRF studio play a crucial role in sampling the Ray Bundles into Ray Samples, which are then processed by Fields to extract meaningful quantities such as color and density (Figure 49). The NeRF studio framework’s support for a diverse array of models and field components enables it to handle various NeRF applications with efficiency and precision. The models, including MLPs, voxel grids, hybrid models, and real-time optimized models, provide the necessary

computational power and flexibility. The effectiveness of models in NeRF studio is further enhanced by various feature encoding schemes like Fourier Features, hash encoding, spherical harmonics and matrix decompositions, which transform input data into formats that models can process more efficiently.

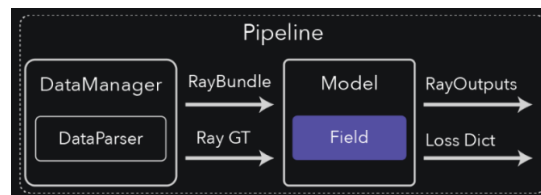


Figure 49: Pipeline Architecture in NeRFstudio for NeRF Training and Rendering

Fields are essential model elements that turn Ray Samples into meaningful attributes like color or density, which are used to create photorealistic 3D renderings. Fields associate a region of space with some quantity, using activation functions to introduce non-linearity, which helps capture complex patterns in data. Voxel grids provide a spatial representation of the scene, with each voxel storing information about the 3D space. Mixed MLPs combine multiple tasks or feature sets into a single network, enhancing computational efficiency and effectiveness. Surface normal MLPs predict surface normals, which are vectors perpendicular to points on surfaces, crucial for rendering realistic lighting and shading effects. Spatial distortions simulate variations in a scene, providing more accurate real-world behaviour, while temporal distortions account for changes over time, allowing the framework to handle dynamic scenes where lighting conditions or objects change. Fields work with models to produce the final, highly detailed and realistic rendered image of 3D scenes. (Tancik, et al., Nerfstudio: A Modular Framework for Neural Radiance Field Development, 2023)

6.1.2.4. Real-Time Viewer

The NeRFstudio Real-Time Viewer, inspired by Instant NGP, is a comprehensive tool designed to enhance the workflow for those working with Neural Radiance Fields (NeRFs). It offers a dynamic and interactive environment where users can not only visualize the ongoing training process but also manipulate various aspects of the 3D scene directly from the viewer interface.

The NeRF studio Viewer features a powerful viewport that allows users to view the entire scene, including the perspective of each training camera and all objects within the scene. This tool is invaluable for troubleshooting and optimizing the training process. Users can switch camera views on and off, offering flexibility in how they analyze the scene. Additionally, the viewport enables users to toggle the visibility of scene objects, making it easier to focus on specific elements within the NeRF model. The viewer also supports various visualization modes, enabling users to switch between different output forms such as RGB, depth and others. These options are essential for understanding how the model interprets the scene and for fine-tuning the training parameters.

Moreover, the viewer isn't just a passive tool, it also offers several settings that can directly impact the speed and efficiency of the training process. These options allow users to optimize their workflow by adjusting training parameters, helping to accelerate model convergence or improve the quality of the results.

Once a NeRF model is trained, the NeRF Studio Viewer offers intuitive tools for exporting results. There are two primary methods for exporting: rendering a video or generating a point cloud.

To render a video, a camera path must first be created within the viewer (Figure 50). While NeRF models are not inherently designed to create point clouds, the NeRF studio Viewer offers an option to generate them. (Tancik, et al., Nerfstudio: A Modular Framework for Neural Radiance Field Development, 2023)



Figure 50: Example of camera path in viewer

6.1.2.5. Outputs

NeRF studio provides extensive support for exporting various types of geometry. Among the supported formats, point clouds can be exported as PLY files, which are commonly used in 3D modeling and visualization. For mesh exports, NeRFstudio offers multiple methods tailored to different needs. The TSDF Fusion method converts truncated signed distance functions into mesh representations, producing OBJ files. This approach is compatible with all models, making it a versatile option for many applications. Additionally, Poisson surface reconstruction is available, providing high-quality mesh generation by solving the Poisson equation from point cloud data. This method is particularly effective when working with models like Nerfacto that predict normals, enabling the creation of detailed and accurate meshes.

For users who need to refine and process their meshes further, NeRFstudio also supports advanced texturing workflows. After simplifying or smoothing a mesh offline, users can reapply textures using NeRF. This process involves dense sampling of the texture image and employs barycentric interpolation to accurately map 3D point locations on the mesh. By rendering short rays along the surface normals, NeRFstudio captures RGB values, ensuring that the textured mesh maintains a high level of detail and visual fidelity. (Nerfstudio: A Modular Framework for Neural Radiance Field Development, 2023)

NeRF studio's export capabilities are designed to be flexible and adaptable, supporting various formats and methods to meet the diverse needs of creators and developers. Whether exporting point clouds, generating meshes through TSDF Fusion or Poisson surface reconstruction, or applying textures with Nerf, NeRFstudio provides the tools necessary for precise and high-quality 3D geometry outputs. This flexibility makes it an ideal choice for professionals who require integration with downstream software and the ability to extend and customize their export processes. (Kazhdan, Bolitho, & Hoppe., 2006)

6.2. 3D Reconstruction with SDFStudio

The 3D reconstruction of the scanned scene will be carried out using SDFStudio, a specialized framework built on the NeRFstudio framework. SDFStudio is designed for working with Signed Distance Fields (SDFs), a mathematical concept widely used in 3D graphics to represent shapes and surfaces. By using the flexible design and tools from NeRFstudio, SDFStudio makes it easier to create, test, and use methods that work with Signed Distance Fields (SDFs). These methods are important for tasks that need accurate 3D shapes and detailed surface measurements.

SDFStudio inherits the modularity, flexibility, and ease of use that NeRF studio is known for, allowing users to easily swap out different components such as data parsers, rendering methods, and models, depending on the specific needs of their project. This shared foundation between SDFStudio and NeRFstudio simplifies the process of testing new algorithms and quickly iterating on designs.

While NeRFstudio primarily focuses on Neural Radiance Fields (NeRFs) to synthesize realistic images from 2D photos by modeling light interactions with objects, SDFStudio takes a different approach by using SDFs to directly represent the geometry of objects. This makes SDFStudio particularly powerful for tasks where the exact shape and structure of an object are more important than its appearance under various lighting conditions. (Yu, Chen, Antic, Peng, & Bhattacharyya, SDFStudio: A Unified Framework for Surface Reconstruction, 2022)

In addition to its form viewer, SDFStudio is integrated with Weights & Biases (W&B), a machine learning platform designed to help developers build better models faster. W&B offers lightweight, interoperable tools to quickly track experiments, version and iterate on datasets, evaluate model performance, reproduce models, visualize results, and spot regressions (Figure 51). This integration enables users of SDFStudio to seamlessly track, visualize, and analyze their experiments, facilitating better workflow management and the ability to share findings with colleagues. (Weights & Biases Documentation, n.d.)

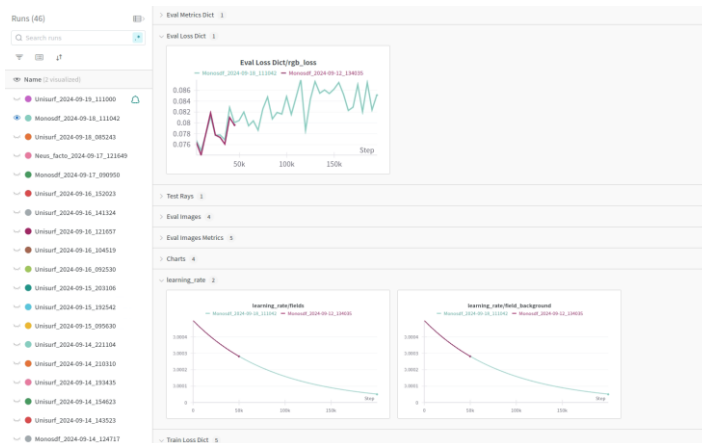


Figure 51: Wandb platform

At SDFStudio, the choice of NeuSfacto, MonoSDF, and UNISURF methods was based on the unique techniques each one offers for reconstructing three-dimensional scenes from images. These three methods were selected because they approach the extraction of 3D information differently, enabling the studio to respond to various challenges. NeuSfacto and UNISURF use techniques inspired by NeRF (Neural Radiance Fields), adapted for more demanding real data and environments. NeuSfacto includes improvements in camera position estimation and advanced sampling techniques,

while UNISURF incorporates surfaces and radiance fields to produce very detailed and accurate 3D representations. On the other hand, MonoSDF uses the Signed Distance Functions (SDFs) technique, utilizing predicted monocular depths and normal maps as supervision to improve the accuracy of 3D representations. The use of SDFs allows MonoSDF to provide solutions when information from multiple cameras is not available, thus offering an alternative when detailed spatial representation from a single perspective is critical.

This different approach by each method allows SDFStudio to be flexible and efficient in a wide range of reconstruction scenarios, offering the ability to handle various types of data and detail requirements in the representations.

SDFStudio has native support for RGB-D data meant for high-quality 3D reconstructions from images co-captured with both color information (RGB) and depth information (Depth). Converting one's own RGB-D data⁶ into the appropriate SDFStudio format (Figure 52), is required because any model in SDFStudio needs to use it. This conversion involves aligning the camera poses with the system used by SDFStudio, normalizing the scene's geometry, and ensuring that the RGB images and depth maps are correctly formatted for accurate 3D reconstruction.

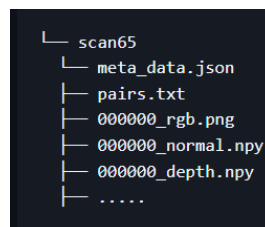


Figure 52: Data structure after converting the custom data to SDFStudio format

6.2.2. Surface Reconstruction Methods Supported by SDF Studio

SDF Studio provides a unified and modular framework for neural implicit surface reconstruction, supporting a range of cutting-edge methods in the field. It integrates several prominent surface reconstruction techniques, offering flexibility in how scenes are represented and sampled. The key methods supported by SDF Studio include UniSurf, VoSDF, and Neus, which all tackle the challenge of reconstructing 3D surfaces from input data but employ different strategies for handling surfaces and sampling points within a scene. The framework also allows for modular combination of different techniques, making it easy to experiment with new ideas and compare results across methods.

SDFStudio supports a variety of scene representations such as Multi-Layer Perceptrons (MLPs), Tri-plane features, and Multi-Resolution Feature Grids, providing flexibility in how geometry and appearance are encoded. Furthermore, different point sampling strategies such as surface-guided sampling (used in UniSurf) and voxel-surface guided sampling (from NeuralReconW) enable efficient and accurate surface reconstructions.

This study explores three advanced methods for surface reconstruction NeuS-facto, UniSurf and MonoSDF by systematically comparing their performance under varying parameters. Each method represents a unique approach to extracting 3D geometry from 2D images. We will examine how modifications to network architectures, sampling strategies, and the integration of monocular cues or multi-view consistency impact the quality and accuracy of the reconstructed surfaces. The

⁶ <https://github.com/autonomousvision/sdfstudio/blob/master/docs/sdfstudio-data.md#rgbd-data>

selection of these methods was strategic, as their distinct techniques offer complementary strengths for tackling different challenges in 3D scene reconstruction. NeuS-facto focuses on neural implicit surfaces, MonoSDF incorporates monocular depth estimation, and UniSurf emphasizes consistency across multiple views. By evaluating their performance in SDFStudio, this study aims to provide insights into how these methods address the complexities of surface reconstruction in various contexts. (Yu, Chen, Antic, Peng, & Bhattacharyya, SDFStudio: A Unified Framework for Surface Reconstruction, 2022)

NeuSFacto and UNISURF use techniques inspired by NeRF (Neural Radiance Fields), adapted for more demanding real data and environments. NeuSFacto includes improvements in camera position estimation and advanced sampling techniques, while UNISURF incorporates surfaces and radiance fields to produce very detailed and accurate 3D representations. On the other hand, MonoSDF uses the Signed Distance Functions (SDFs) technique, utilizing predicted monocular depths and normal maps as supervision to improve the accuracy of 3D representations. The use of SDFs allows MonoSDF to provide solutions when information from multiple cameras is not available, thus offering an alternative when detailed spatial representation from a single perspective is critical. This different approach by each method allows SDFStudio to be flexible and efficient in a wide range of reconstruction scenarios, offering the ability to handle various types of data and detail requirements in the representations.

6.2.2.1. NeuS-Facto Model

The NeuS-facto model is a new and improved method for creating 3D scenes by combining two powerful techniques: neural surface reconstruction (from NeuS) and neural radiance fields (from Nerfacto). It uses a smart approach, inspired by mip-NeRF360, to pick fewer but better samples along rays of light, making the process faster and more accurate. (Yu, et al., Methods, 2022)

By reducing the number of samples needed, NeuSfacto finds a good balance between building detailed 3D surfaces and using efficient rendering techniques. This leads to quicker, high-quality 3D scene creation with better details.

6.2.2.1.1. NeuS

NeuS combines two important concepts: the Signed Distance Function (SDF) and techniques used in NeRF (Neural Radiance Fields). While NeuS primarily relies on the use of SDF to represent surface geometry, it borrows elements from NeRF for rendering color and light. At the same time, NeRF focuses on modeling light and color by predicting light interaction with objects through camera rays. NeuS uses similar techniques for volume rendering (as in NeRF) to enhance rendering and link SDF with color representation. (Wang, καλ ουν., 2021)

The rendering process of NeuS involves two main stages. Volume rendering and surface rendering. Initially, volume rendering is used to understand the general structure of the scene. This is done by sampling the volume of the scene with a 3D grid, where each voxel contains information about local geometry and appearance. As the model improves, the process shifts to surface rendering, which focuses on improving the detail of the surface defined by the SDF and the computations around these surfaces, achieving better detail and reducing computational burden. To enhance the efficiency of the rendering process and address real-world challenges, NeuS uses advanced sampling techniques. Importance sampling focuses on critical areas of the scene, such as edges or areas with significant

detail. Adaptive sampling changes the sample density according to the complexity of the area, offering more samples in complex areas and fewer in simpler ones.

A significant challenge in 3D reconstruction is accurately estimating the camera's position. NeuS addresses this by allowing continuous adjustment of the camera position during training. Instead of using static camera positions, NeuS adjusts these positions through an iterative process to reduce errors in the rendered images.

For accurate 3D model reconstruction, it is crucial to select the appropriate weight function that links output colors to the Signed Distance Function (SDF). This function must have two key characteristics. It should give greater importance to the points where the SDF surface intersects the camera plane (unbiased). In other words, the function should reach a local maximum at the intersection with the zero-level set of the SDF, ensuring that this surface contributes proportionally more to the pixel color. Additionally, the weight function should give more importance to points closer to the camera, especially when there are multiple surface intersections along a ray (occlusion-aware). This means that if there are multiple surfaces along the same line of sight, the function should prioritize the points closer to the camera over those further away. These requirements ensure that the weight function will contribute accurately to the color representation of the 3D models, improving the quality and fidelity of the reconstruction.

6.2.2.1.2. Nerfacto

Nerfacto is a 3D scene reconstruction model designed to address the challenges encountered when working with real-world data, such as photos or videos, in contrast to NeRF (Neural Radiance Fields), which performs well primarily with synthetic or ideal data. Although it is based on NeRF (Neural Radiance Fields), Nerfacto introduces new techniques that make it more efficient and more suitable for real-world applications (Figure 53). (Tancik, et al., Nerfacto, 2022)

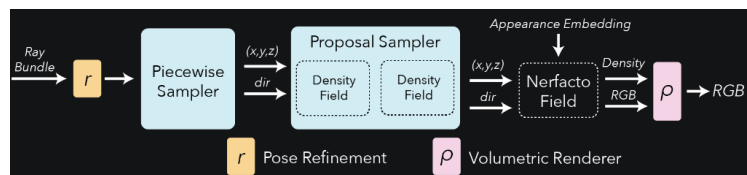


Figure 53: Nerfacto pipeline model

In contrast to NeRF, one of the primary changes introduced in Nerfacto is an enhancement in camera pose refinement. While NeRF uses fixed, predetermined viewpoints, Nerfacto improves camera positions during training. The cameras adjust their locations through backpropagation to reduce the error observed in the results.

Backpropagation is a method used after processing a batch of data to update each neuron's contribution to the model's output. It works in two phases. In the forward pass, input data is passed through the network to produce an output (Figure 54). In the backward pass, the loss between the actual and predicted output is calculated and propagated back through the network. The algorithm then adjusts its weights and biases to improve future predictions. Backpropagation is a fundamental method for training artificial neural networks. (Backpropagation in Neural Network, 2024)

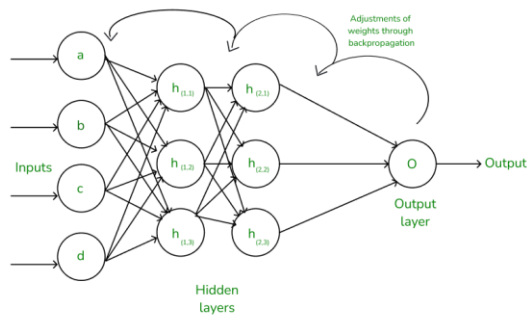


Figure 54: Backpropagation method

This method is a critical step for using real-world data, as camera positions may be inaccurate. It corrects these positions during training, improving the quality of the reconstruction and reducing blurry and low-quality results. In contrast to NeRF, where sampling is done uniformly along the rays without considering the distance from the camera or the importance of points, Nerfacto introduces more advanced techniques like Proposal Sampling and Piecewise Sampling, making it more efficient for reconstructing scenes from real-world data.

Piecewise Sampling in Nerfacto divides samples into two categories. Close samples are taken more densely near the camera to accurately capture the details of nearby objects, which are more visible and significant for the scene's reconstruction. Distant samples, on the other hand, are sampled more sparsely as they move away from the camera, saving resources in areas where detail is less critical. This strategy allows Nerfacto to render close-up details with precision while maintaining efficiency in distant areas of the scene, where high resolution isn't as necessary.

At the same time, Proposal Sampling further enhances the sampling process. This sampler focuses on areas of space that contribute more to the final render, such as points where rays first intersect with surfaces. This approach reduces the waste of resources on irrelevant areas of the scene, unlike NeRF, which samples uniformly along the rays without prioritizing the area's most important for the result.

The core of Nerfacto is the Nerfacto Field (Figure 55), which consists of a series of stages and encoding techniques that contribute to producing the results, such as density and RGB colors. This process begins with sampling the scene and encoding spatial and directional information to render the scene with both accuracy and efficiency.

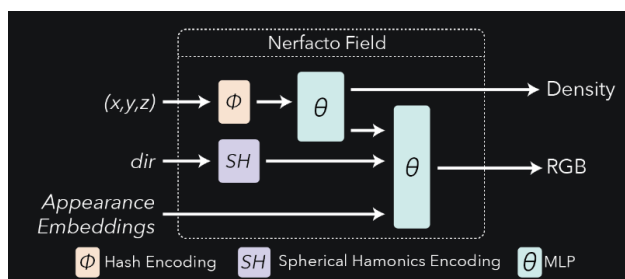


Figure 55: Nerfacto Field for extraction results

In Nerfacto⁷, the density field takes a more efficient and focused approach. Instead of capturing high-frequency details across the entire scene, it provides a coarser representation that guides the sampling towards important areas. High detail is not necessary at this stage; the field only needs to be accurate enough to direct sampling to regions that matter most for the final render. The 3D coordinates of the scene (x, y, z) are processed through a hash encoding method, allowing the system to represent spatial information more efficiently without needing large memory to store all scene values. The result of this encoding is fed into a small Multi-Layer Perceptron (MLP).

Viewing direction is encoded using spherical harmonics, a method that efficiently breaks down direction into a limited number of angle-based components, preserving the required accuracy. Appearance information, representing characteristics like lighting or color variations, is combined with other inputs like position and direction. This technique improves the quality of the final render by accounting for variations in the appearance of each image. The outputs of these encodings are fed into a small multi-layer neural network (MLP), which predicts two key elements: the scene's density based on position (x, y, z) and the colors of each scene point, using the viewing direction and appearance information.

In addition, Nerfacto uses two encoding techniques to improve efficiency. Hash encoding and hash table encoding. Hash encoding converts categorical variables into numerical values using a hashing function (Figure 56). While one-hot encoding can be useful, it is not suitable for large datasets because it requires substantial memory, and other encoding techniques are slower. Hash encoding assigns a unique integer value to each category within a predefined range using a hash function. Collisions can degrade model performance, but this can be mitigated by increasing the hash space or using a different hashing function. Nerfacto also employs hash table encoding to achieve faster and less memory-consuming storage and retrieval of density information (Swayam, 2023). This approach avoids the need for large, detailed tables, saving the memory required to represent the scene. Spatial density is learned using a small multi-layer neural network (MLP) in Nerfacto, which facilitates training in a simplified manner without compromising important regions. This helps Nerfacto sample meaningful directions while simplifying the density field, making it smoother and less expensive while retaining high-frequency information without needing to learn at every pixel.

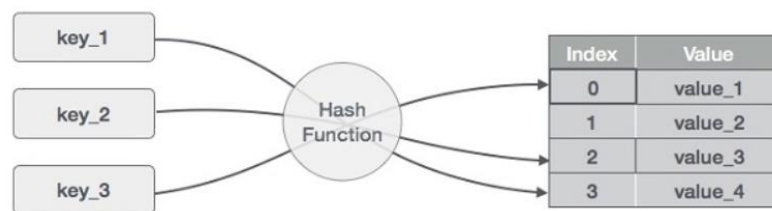


Figure 56: Hash encoding example (Tutorialspoint, n.d.)

This reduction in detail does not compromise quality, as the density field guides sampling in high-probability areas of the scene, while higher-resolution details are captured by other stages in the reconstruction. Nerfacto combines a sequence of encoding and machine learning techniques to synthesize high-quality scenes, maintaining fidelity and quality while optimizing resource efficiency through density field-guided sampling and hash encoding. (Tancik, et al., Nerfacto, 2022)

⁷ <https://github.com/autonomousvision/sdfstudio/blob/master/docs/sdfstudio-methods.md#neus-facto>

6.2.2.1.3. Mip-NeRF 360

Mip-NeRF 360 builds on conventional Neural Radiance Fields (NeRF) technology, introducing a more advanced method for representing and rendering images through deep learning across unbounded scenes. This approach is designed to represent and render images through deep learning across scenes that are not restricted by a fixed viewpoint or location. Unlike traditional methods, Mip-NeRF 360 effectively handles scenes that extend to infinite depth-distance, which helps address common issues encountered in these types of environments.

The core innovation of Mip-NeRF 360 is its novel scene parameterization, which is optimized for better handling of unbounded spaces. This parameterization operates in non-linear space, allowing more computational power to be allocated to elements closer to the camera, while distant objects, which contribute less to the overall image, consume fewer resources. This selective focus enhances efficiency without sacrificing visual quality. Mip-NeRF 360 introduces a new sampling and scene representation strategy that employs two separate Multi-Layer Perceptrons (MLPs). The first of these, called the "proposal MLP," is responsible for calculating the density distribution of all locations within a scene, without considering color. It generates rays and passes them to a density prediction model, referred to as the "Density MLP," which samples the densities. These sampled rays are then sent to a second network, the "NeRF MLP," which predicts the final image outputs for each ray (Figure 57).

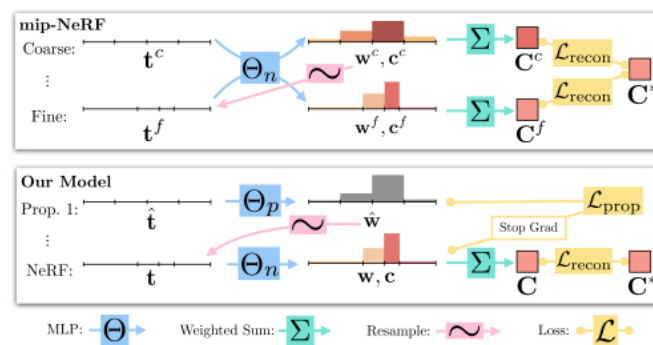


Figure 57: Pipeline comparison of Mip-NeRF and Mip-NeRF 360

By separating these processes, computational efficiency is improved on the Mip-NeRF 360. The "MLP proposal" addresses well the maximization of scene volumes, so that it is possible to control volumes at real-time scale by emphasizing quality, especially when different graded thresholds for scenes are proposed according to their importance. While "NeRF MLP" uses an MLP to focus more intensively on the reconstruction of image samples by paying more attention to the context of the images that need to be captured, thus achieving higher scene reconstruction performance.

Moreover, the training process of Mip-NeRF 360 is streamlined by remapping entire scenes to represent Gaussians inside a predetermined bounding box. Rather, the shallow Proposal MLP is used to read from Gaussian-encoded intervals: instead of producing weights and colors in this setting, we go directly for outputting weights. These weights are resampled and passed into the NeRF MLP to predict high-fidelity colors (and more fine-scale weights) for rendering pixel colors. The coupled proposal loss function enforces consistency between the histograms generated by a shallow Proposal MLP and those produced from NeRF, offering tighter control during rendering. The most significant advancement in Mip-NeRF 360 is a novel regularization to suppress typical artifacts such as "floaters" and "background collapse." Model-generated data artifacts are frequently found where the model is trying to explain some of the sparsest, smallest regions by representing them with tiny high-density sticks, or this often happens when dust clouds are being represented as semi-transparent go closer to

the camera. Thus, Mip-NeRF 360 introduces a novel regularizer that is based on a step function which parameterizes the distance of rays. The regularizer tries to localize densities around rays, narrowing down the spread of density and eliminating the semi-transparent regions that usually cause these artifacts. In addition, this scheme of regularization for each ray histogram helps in generating less unphotorealistic views.

The addition of these new methods has resulted in a considerable enhancement of the model's performance. Mip-NeRF 360 introduces an architecture that unifies the synthesis of plausible images and detailed depth maps for complex scenes without bounds, leading to more realistic yet high-quality results when generated samples are already comparably accurate to those synthesized by using a model with identical training time. (Barron, Mildenhall, Verbin, Srinivasan, & Hedman, 2011)

6.2.2.2. UniSurf Model

UNISURF introduces a revolutionary approach by merging neural implicit surface models with radiance field techniques. This integration facilitates a dual capability where the model not only predicts the geometry using implicit surfaces but also captures the appearance and lighting variations through radiance fields. Implicit Surface Models are utilized to define the geometry of the scene as zero level sets of a Signed Distance Function (SDF) parameterized by a neural network. While implicit surfaces model geometry, radiance fields are used to model the color and light interactions within the scene. By sampling points along camera rays and predicting the color and density at each point, radiance fields enable photorealistic rendering from novel viewpoints.

UNISURF's rendering pipeline uniquely combines volume rendering and surface rendering (Figure 58). Initially, the entire scene volume is sampled to capture the coarse structure of the scene. This stage uses a 3D grid where each voxel stores a feature vector that captures local geometric and appearance properties. As training progresses and the model's predictions become more accurate, the rendering focus shifts more towards surface rendering. Here, the surface is defined precisely by the implicit function, and rendering computations are concentrated around these surfaces to enhance detail and reduce computational overhead.

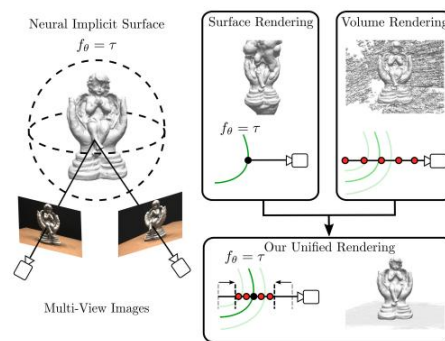


Figure 58: UniSurf rendering pipeline

To optimize the rendering process and handle real-world complexities, UNISURF implements several advanced sampling strategies. Importance Sampling focuses on regions that significantly impact the rendered image, such as edges or areas with high detail. By prioritizing these areas, UNISURF can allocate more computational resources where they are most needed, improving both efficiency and output quality. Adaptive Sampling adjusts the density of sample points dynamically

based on the complexity of the region being rendered. Areas with complex geometry or higher visual importance receive more samples, while simpler regions are sampled less densely.

UNISURF addresses one of the common challenges in 3D reconstruction the inaccuracies in camera pose. Unlike traditional methods that fix camera poses, UNISURF allows for their refinement during training using gradient descent. By backpropagating the errors from the rendered images back to the camera parameters, UNISURF iteratively adjusts the camera positions to minimize rendering errors.

The backbone of UNISURF's capability lies in its neural architecture and the training process. The Multi-Layer Perceptron (MLP) is crucial for modeling the continuous fields required for both geometry and appearance. It is designed to be deep enough to capture complex patterns but also efficient to prevent overfitting. The training leverages a composite loss function, including photometric loss (ensuring rendered and actual images match) and regularization terms (encouraging smoothness and continuity in the predicted surfaces). To enhance the model's sensitivity to detail, positional encoding is applied to the inputs of the MLP. This technique helps in representing high-frequency functions and capturing fine details in the scene. (Oechsle, Peng, & Geiger, 2021)

6.2.2.3. MonoSDF Model

MonoSDF, or Monocular Single View Depth Fusion, is an innovative tool that creates accurate 3D surfaces from a single image. Unlike traditional techniques that require multiple images from various angles to gather 3D information, MonoSDF achieves the same result by using unique geometric cues from just one image. It does this by extracting implicit surfaces. From each image, the necessary geometric cues required for reconstruction are extracted, which include depth maps and normal maps (Figure 59). A normal map uses RGB values to represent the direction that surfaces in the scene are facing. Essentially, it informs the model about the orientation of each part of the object's surface. A depth map is a crucial factor for accurately reconstructing the scene's geometry, as it affects how light interacts with the objects, which in turn influences the appearance and realism of the rendered 3D model. (Understanding the normal mapping process., n.d.)

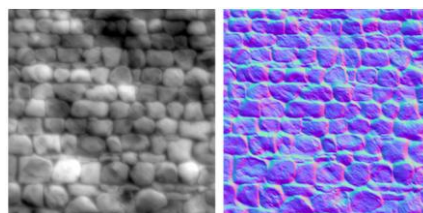


Figure 59: Depth and normal map from one image (Unity, n.d.)

The two main components in the neural network of MonoSDF predict, respectively, the distance from any point to the nearest surface and the colors. This structure enhances the model's understanding of both the shape and appearance of objects. This approach highlights the model's strength in utilizing advanced computer vision techniques to extract spatial depth and surface orientation from limited visual data, particularly when multiple views are not available.

More specifically, MonoSDF uses a Signed Distance Function (SDF) to represent the geometry of the scene (Figure 60). The options for the SDF representation involve a simple MLP, a dense SDF grid, the Single-Resolution structure and two hybrids, the Feature Grid with MLP Decoder and the

Multi-Resolution Feature Grids with MLP Decoder. In addition to the 3D data, the color values are also estimated based on an MLP (Multi-Layer Perceptron).

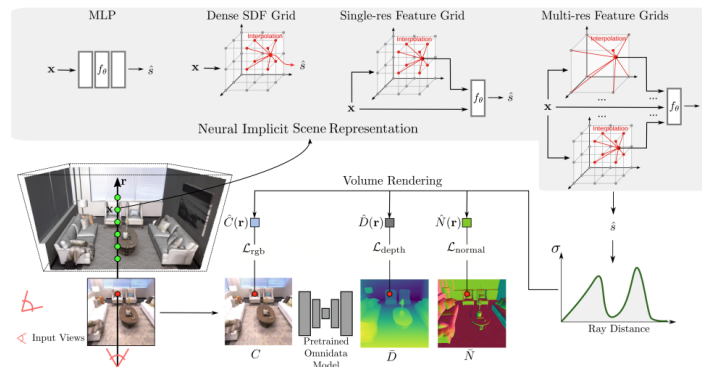


Figure 60: MonoSDF Model Overview

The Dense SDF Grid is a method where the space of interest is discretized into a three-dimensional grid. Each cell in this grid stores an SDF value, representing the shortest distance from the center of the cell to the nearest surface of the object, with the sign indicating whether the point is inside or outside the surface. To determine the SDF value at any arbitrary point in space, which may not necessarily be at the center of a grid cell, trilinear interpolation is used. This method interpolates the values based on the eight nearest grid points around the arbitrary point, providing a smooth estimate of the distance field.

An alternative to the dense grid is modeling the SDF using a single Multi-Layer Perceptron (MLP). The output of the MLP is a single value representing the signed distance from the input point to the nearest surface.

Hybrid approaches within the context of MonoSDF and similar neural surface reconstruction methods combine MLPs with feature grids to improve 3D surface modeling. These methods leverage the MLPs' capability to learn general characteristics and the details that can be captured by feature grids. In the approach with a Feature Grid and MLP Decoder, the MLP is used to calculate the Signed Distance Function (SDF). However, instead of working alone, the MLP also receives information from a feature grid, which is a three-dimensional grid storing data. In the approach with Multi-Resolution Feature Grids and MLP Decoder, multiple grids of different resolutions (cell sizes) are used to capture details at various scales. Like the simple grid, the MLP receives information from these grids, but here it uses data from grids of different resolutions simultaneously. This allows the MLP to capture details at different levels, offering greater accuracy and a more faithful representation of the scene, as it incorporates both general and detailed information from the various grids.

An MLP used for predicting colors is specifically designed to analyze each point in the scene. This network considers the exact position of each point in space and the direction from which the camera sees it. The network then combines this data with the geometric properties of the surface at that point, such as the surface normals, which indicate the surface's orientation. In this way, the MLP can predict the color that each point on the 3D surface will have, contributing to the creation of a more realistic and visually convincing final image.

Moreover, MonoSDF uses volume rendering to convert the 3D representations into 2D images, a process crucial for comparing the model's output with real images, thereby guiding optimization. The process begins with casting rays from the camera's viewpoint through each pixel.

Points are sampled along these rays, and the Signed Distance Function (SDF) and color values are evaluated. The SDF values are converted into densities using a predefined function, helping to manage surfaces crossing the zero level-set. Finally, the color is accumulated along the ray, considering the computed densities and transparencies, allowing for the rendering of realistic images.

As previously mentioned, MonoSDF exploits monocular geometric cues to address challenges in reconstructing complex real-world scenes, especially in areas with sparse or textureless regions. Depth maps are generated using a pretrained model like Omnidata, which has been trained on a wide variety of images to reliably predict depth from a single view. Meanwhile, normal maps are derived through a process of estimating the surface normals in the 3D space, which complement the depth maps by providing detailed information about the surface geometry that depth alone cannot, such as the tilt and curvature of surfaces.

Finally, the optimization process in MonoSDF is designed to continually improve the model's output. This is achieved through a series of loss functions that ensure both geometric accuracy and visual fidelity. The RGB Reconstruction Loss aims to reduce the difference between the colors of the rendered images and those of the actual photographs. By adjusting the model parameters to minimize these discrepancies, the model learns to produce visually similar outputs to the input images, enhancing the realism of the reconstructed scenes. The Eikonal Loss ensures that the gradients of the SDF maintain a norm close to one, which is crucial for a smooth and realistic surface representation. This regularization helps prevent sharp discontinuities or unrealistic smoothness in the rendered surfaces, promoting a more natural and physically plausible surface model. The Depth Consistency Loss aligns the depth values predicted by the model with those from the depth maps, dynamically correcting scale and shift issues per batch. This is vital, as each image may have its own scale and depth perception, especially when dealing with diverse datasets or varying camera setups. Finally, the Normal Consistency Loss ensures that the normals rendered by the model match those provided by the normal maps. Accurate alignment of normals is crucial for correctly rendering the interaction of light with surfaces, which affects the visual quality of shadows, highlights, and textures. (Yu, Peng, Niemeyer, Sattler, & Geiger, 2022)

7. Object Scanning Process

Before starting the scanning process, the camera underwent an initial calibration using the Intel RealSense Viewer v2.54.2. This tool facilitates self-calibration by allowing the camera to adjust its internal parameters automatically, compensating for environmental changes and ensuring the reliability of 3D reconstructions. This feature addresses common issues such as temperature fluctuations and minor physical displacements, ensuring the camera operates optimally over time. Additionally, maintaining correct intrinsics is critical, as they define the geometric and optical characteristics of the camera, which are critical for accurate depth measurement and 3D reconstruction

To achieve optimal scanning results, several parameters were defined. The frame rate was set to 5 frames per second (fps) for both depth and RGB data streams. This frame rate was chosen to balance the need for sufficient data capture with the practical considerations of processing time and dataset size. Also, depth and RGB data were captured at a resolution of 640 x 480 pixels. This resolution was selected to maintain a manageable dataset size while ensuring adequate detail for accurate 3D reconstruction. The chosen configuration aimed to generate fewer frames, thus creating smaller datasets. This approach reduces the processing time required for data analysis and 3D model

generation, making the process more efficient without compromising the quality of the final output. Both the frame rate and resolution were set to ensure synchronization between color and depth data. (RealSense, 2018)

The scanning process involved the vertical movement of the camera, both downward and upward, to fully capture the environment, with the movement being performed in a right-handed direction. These movements were carried out smoothly to avoid abrupt changes that could affect the quality of the data, while maintaining an overlap of over 40% between the movements. The final dataset included 833 color and depth frames.

The scanning method of a space is crucial for the success of camera alignment. If the captured images do not have sufficient overlap or common features, there is a high likelihood of alignment errors and geometric issues in the scene representation.

A notable example of this challenge arose from a test scan conducted in a corridor at the University of West Attica. The scanning was done separately for each side of the corridor. During the camera's turn after completing the scan of the first side, the movement was counterclockwise instead of clockwise. This choice resulted in capturing fewer nearby objects that could have aided in subsequent processing. As a result, the connection between the two sides of the corridor relied mainly on shared elements like the floor and ceiling. This limited overlap led to poor alignment and significant deviation in the final point cloud composition (Figure 62). Therefore, it is important to ensure that the scan directions allow for the best possible overlap and capture of important objects.

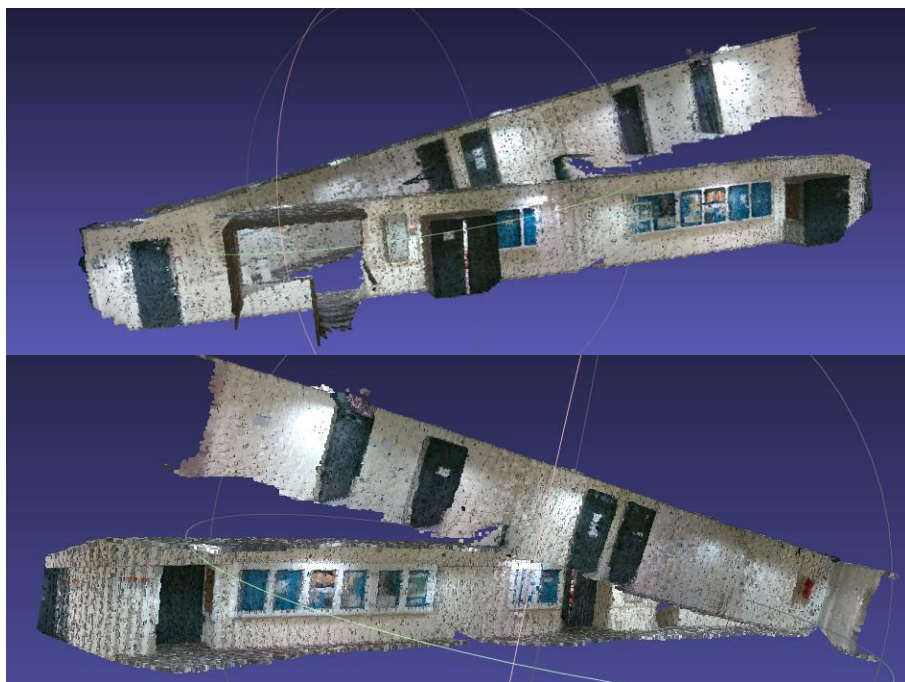


Figure 611: Misalign merged point cloud

7.1. Sunlight Interference Issue

During the scanning process, an issue was encountered with the RGB sensor of the Intel RealSense D455 camera displaying a pink hue in the lower right part of the image when exposed to sunlight (Figure 62). This is a known behaviour for the D455 camera in extremely bright scenes, caused

by a saturation issue. It occurs when the sensor is set to short exposure times under high light intensity, leading to color distortion. (Garcia, 2020)



Figure 62: Saturation issue in RGB images in sunlight exposure

To mitigate this issue, it is advisable to refer to the Intel white paper on Optical Filters for Intel RealSense Cameras. One effective solution is to use a Neutral Density (ND) filter, often referred to as "sunglasses" for cameras. ND filters work by uniformly reducing the amount of light entering the camera lens without affecting the color of the light. This reduction in light intensity allows the camera to operate with longer exposure times, thus avoiding the saturation that causes the pink hue. (John, Sweetser, & Grunnet-Jepsen, n.d.)

Additionally, incorporating a polarizing filter can further enhance image quality by reducing reflections and glare from bright surfaces, such as water or metal. Polarizers help in controlling the polarization of light, which can significantly diminish unwanted reflections and improve color saturation in images. By combining an ND filter with a polarizer, the likelihood of encountering color distortions, like the pink hue observed, can be greatly minimized, leading to clearer and more accurate images.

Due to the lack of the necessary filters and the camera's difficulty in adequately capturing and utilizing the pattern under strong sunlight, the scanning was conducted indoors. The time of day was chosen when the sunlight was not strong, and there was no direct exposure to the object being scanned, to avoid issues with image quality and to ensure the most reliable recording possible.

8. Pose Extraction and Camera Alignment

Open3D offers a powerful data structure for image manipulation, supporting a variety of functions. Extracting a point cloud from an RGBD image involves several steps, each of which is crucial for accurately converting color and depth information into a 3D representation.

This chapter discusses the process of aligning multiple point clouds in a global coordinate system and extracting the corresponding 4x4 transformation matrices (poses) for each frame, which are necessary for the further processing of the data during the 3D reconstruction process.

To achieve this goal, Open3D's Multiway Registration using the pose graph process was applied. This method involves constructing a pose graph and optimizing it through multiple registration algorithms to ensure accurate and uniform alignment of all points. The input typically consists of point cloud geometries, while the output is a series of rigid transformations (4*4 homogenous matrices) that align the point clouds within the global space.

Initially, using a coarse point-to-plane ICP with a larger matching distance, we first improve the rough alignment based on odometry by using a minimum point-to-plane distance between the source and target cloud points. To further improve this initial alignment, we apply a finer point-to-plane ICP using a smaller matching distance to further optimize the alignment to achieve higher accuracy.

Due to the complexity of the scanning scene in this specific application, Colored ICP is also integrated, which uses both geometric and photometric information to further refine the alignment. This step leverages the color information from the RGBD images to enhance the accuracy of the registration, particularly in regions where geometric features alone may not provide sufficient information for accurate alignment.

The combination of point-to-plane ICP and color ICP significantly enhances the overall accuracy and robustness of the alignment process. By incorporating photometric information alongside geometric information, Colored ICP extends the capabilities of traditional ICP, making it well-suited for a wide range of 3D reconstruction and scene understanding tasks.

A very important step involves the transformation of the exported poses from the camera's coordinate system to the SDFStudio coordinate system for their correct placement. The D455 uses a left-handed coordinate system, where the +X axis points to the right, the +Y axis points downward, and the +Z axis extends forward from the camera. In contrast, SDFStudio follows the right-handed coordinate system used in OpenGL/Blender, where the +X axis remains to the right, the +Y axis points upward, and the +Z axis points backward. This inversion is achieved using a transformation matrix (Figure 64) that changes the signs of the Y and Z axes by combining the two inverse transformation matrices (Figure 63).

Figure 63: Transformation matrices for rotation of Y (left) and Z (right) axes

The transformation matrix used refers to that reverses the directions of the Y and Z axes so that the D455's coordinate system aligns with the one used in SDFStudio. After applying this transformation, the camera's data adopts the same orientation as the data used by SDFStudio, ensuring correct interpretation and processing. This transformation is applied only to the camera poses, ensuring that the positional and orientational data is fully compatible with the coordinate system used in SDFStudio. This allows for accurate representation of the camera's movement and orientation within the 3D space. (GateVidyalay, 2020)

Figure 64: Transformation matrix for rotation of Y and Z axes



Figure 68: Misalignment or noisy areas of the final combined point cloud

9. Parameter Selection and Preparation for Reconstruction

SDFStudio, for reproducing reconstruction results with RGBD data, suggests training the NeuS model with the following defined parameters, as presented in the table below:

Table 1: Example parameter values used in the 3D reconstruction process with the NeuS model

Model Parameter	Example Value	Default value
pipeline.model.sdf-field.use-grid-feature	TRUE	FALSE
pipeline.model.sdf-field.hidden-dim	256	256
pipeline.model.sdf-field.num-layers	2	8
pipeline.model.sdf-field.num-layers-color	2	4
pipeline.model.sdf-field.use-appearance-embedding	TRUE	FALSE
pipeline.model.sdf-field.geometric-init	TRUE	TRUE
pipeline.model.sdf-field.inside-outside	TRUE	TRUE
pipeline.model.sdf-field.bias	0.8	0.8
pipeline.model.sdf-field.beta-init	0.3	0.1
pipeline.datamanager.train-num-images-to-sample-from	-1	500
trainer.steps-per-eval-image	5,000	0.0
pipeline.model.background-model	none	none
pipeline.model.sensor-depth-l1-loss-mult	0.1	0.0
pipeline.model.sensor-depth-freespace-loss-mult	10	0.0
pipeline.model.sensor-depth-sdf-loss-mult	6,000	0.0
pipeline.model.mono-normal-loss-mult	0.05	0.0
pipeline.datamanager.train-num-rays-per-batch	1,024	1,024
include_sensor_depth	TRUE	FALSE

From the parameters mentioned above, some will remain fixed, according to the recommendations of the SDFStudio team. These concern the number of images used during training (all images = -1), the Inside-Outside parameter, which is suggested to be kept as "True" for indoor scans, the inclusion of the generated camera depth maps in the final model, the reconstruction of the background, and the mono normal loss.

The parameters to be used for evaluating the reconstructions include "*pipeline.model.sdf-field.use-grid-feature*," which determines whether the model will use a grid feature to represent the

3D space. The grid feature essentially divides the space into small cubes (grid), and each cube stores information about the part of the scene it represents (Figure 69). Application of grid features is advantageous if any scene involves many intricate details since the model can easily arrange these details in a network. In the example, this feature is enabled (TRUE) which means a grid-based representation can be used, but the default status is FALSE meaning a grid will not be used by default in a model. (Wang, Bleja, & Agapito, 2022)

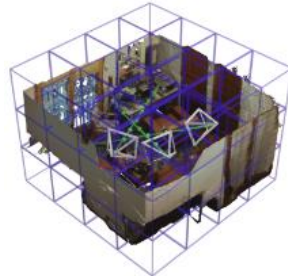


Figure 69: 3D scene representation divided into grid-like cubes

The *pipeline.model.sdf-field.hidden-dim* parameter indicates the size of the hidden layers in the neural network architecture. Neural networks consist of multiple layers and each layer contains a number of neurons that process information. The hidden dimension refers to the number of neurons in these layers. A larger hidden dimension allows the model to learn more complex patterns and details, but it also increases the computational resources required for training and execution. (Dutta, 2024)

Depending on the setting on the number of layers, a certain number of layers are used to process the Signed Distance Field (SDF), which is the technique that the particular model applies to model the shapes of objects in 3D space. When there are many layers, something complex is achieved through the shapes and details that the model can recognize. However, expanding the number of layers in a network increases the complexity of the model, this results in slowing down the training time and the time required to make predictions. (Krish, 2019)

The color layer depth determines how many layers will be used to calculate the scene's color information. In addition to mimicking the shapes and geometry of objects, the model is supposed to determine how they should be painted and textured. Even more layers here allow the model to extract detailed and complex color patterns and thus enhance the realism of the represented scenes, but impose more computational burden

The SDF field parameter is "*pipeline.model.sdf-field.use-appearance-embedding*", which, when set to true, indicates that the model has an "appearance embedding" - something similar to memory that enhances the model's understanding of how objects appear under different lighting and viewing conditions. The fact that appearance embedding is used allows for flexibility in object illumination and can successfully produce an output that is reasonable even from different angles or under different conditions. (Chen, Wu, Li, Li, & Liu, 2024)

The "*pipeline.model.sdf-field.geometric-init*" parameter specifies whether the model starts with a geometric principle on which it is based. A good geometric initialization means that the model can start closer to the correct solution and therefore learn better. Geometric initialization helps the model in that it introduces the rudimentary ideas of geometry and scene structures before the data. (Esposito, et al., 2024)

The term bias in a neural network refers to an enhancement term, which aids the model to fine tune it. This parameter specifies the level of the intrinsic bias within the training process. bias enables the model of making more appropriate adjustments within a situation to make more accurate predictions if the input parameters don't match the prior trained patterns precisely. (Effect of Bias in Neural Network, 2018)

The parameter of *"pipeline.model.sdf-field.beta-init"* defines the starting value of "beta" which defines the boundaries in the model's prediction. Appropriate initialization of beta is critical to achieving the best performance in the model and especially in discriminating objects within a scene and empty space areas. Beta controls how sensitive the model is to changes in the space it models. (Carnegie Mellon University, 2022)

The number of steps defined in the evaluation image determines how often the model will inspect the images in order to monitor its progress. One important thing for the model is to be able to frequently evaluate its performance and adjust learning accordingly. But if the model evaluates too often then training can take a long time, and if evaluations are performed too infrequently then the model's performance can be significantly affected.

To regulate the weight of the depth predictions from sensors, there is a control of the L1 loss parameter for such. L1 loss measures the distance between the model's predictions and the actual depth values. Adjusting the weight of this loss controls the model's focus on the accuracy of depth predictions from sensor data. A higher weight prioritizes depth accuracy during training.

The SDF loss parameter of sensor depth determines how much the model should attempt to minimize the error between the model's predictions on signed distance field and the actual depth values from the sensor. When modifying this parameter, the model is able to improve the alignment of the perceived object shapes with their distances from the camera thus improving on depth and shape estimations. (Alake, 2023)

Sensor depth free space loss parameter that determines the degree of emphasis the model places on free space, the regions of the scene where there are no objects. Modifying this loss enables the model to distinguish between the objects and the background which is very important in building the 3D scene representation. (Wang, Wang, & Agapito, 2023)

The parameter for the number of rays per batch defines how many rays going from camera through the scene. Rays are applied to obtain the view of the scene and to gain data regarding the objects and their location. Increasing the number of rays per batch provides the model with more data to learn from at each training step but requires more computational power.

9.1. Comparison of Training Results and Reconstruction Quality of MonoSDF and UniSurf for the Selection of Number of Iterations

The choice of Neus-facto was made to achieve optimal results, following the proposed model of SDFStudio. The other two models were selected due to their different methodologies, allowing for a comparison both between the models using the same parameters and for each model individually with varying parameter changes. Besides the recommended parameters, each model/method includes an additional number of parameters with different values depending on the requirements of each application.

An important parameter that affected the training time for MonoSDF and UniSurf was the number of iterations. Specifically, for MonoSDF and UniSurf, the number of iterations was 200,000 and 100,000 respectively, while for Neus-facto, it was only 20,000. The number of iterations refers to how many times the model updates its parameters during training. In the context of machine learning, an iteration typically occurs when the model processes a batch of training data and adjusts its internal parameters based on the resulting error (loss). (Epochs, Batch Size, Iterations - How are They Important to Training AI and Deep Learning Models, 2024)

The results of each model were compared both with the initial number of iterations and with 50,000 iterations. The comparison was based on the final 3D reconstruction, training time, PSNR (Peak Signal-to-Noise Ratio), the accuracy of depth and color predictions, as well as parameters such as Eikonal Loss, Freespace Loss, and SDF Loss. These parameters play a crucial role in the quality of the reconstruction. The data was collected through wandb with the aim of evaluating whether increasing the number of iterations provides a significant improvement in reconstruction quality or if the model has already been sufficiently trained with fewer iterations.

The learning rate is one of the most important parameters in training neural networks. It determines the size of the adjustments the model makes to its parameters, such as weights, to reduce the loss. A high learning rate can speed up learning but may cause instability. On the other hand, a low learning rate allows for more precise adjustments but increases training time. Often, a learning rate decay is applied, where it starts high for fast learning and gradually decreases for more precise adjustments. (geeksforggeeks, 2023)

The Peak Signal-to-Noise Ratio (PSNR) is a metric used to evaluate the quality of reconstructed images produced by a neural network. It is calculated as being inversely proportional to the logarithm of the Mean Square Error (MSE) between the real image (ground truth) and the generated high-resolution (HR) image. This means that a lower MSE corresponds to a higher PSNR, which indicates better quality in the reconstruction. A high PSNR value signifies a closer resemblance of the reconstructed image to the original, reflecting the effectiveness of the reconstruction algorithm. Acceptable PSNR values typically range from 30 to 50 dB for high-quality images, while values below 20 dB are considered poor quality. In this context, the PSNR value is computed across all images in the dataset, including both RGB images and depth images, ensuring a comprehensive evaluation of the reconstruction quality across different modalities. (lesalnieks, 2022)

9.1.1. MonoSDF num iteration comparison

Specifically, MonoSDF, in addition to the values for time, PSNR, and learning rate, the Eikonal Loss was compared between 200,000 and 50,000 iterations to evaluate the accuracy of surface normalization. Additionally, the Monocular Depth Consistency (depth loss) was compared to assess the stability of depth predictions. Color accuracy was examined through the RGB Loss to evaluate the quality of the color representation in the reconstructions, while PSNR was used to assess the overall image quality produced by the model after each number of iterations. (Yu, Chen, Antic, Peng, & Bhattacharyya, Supervision, 2022)

Table 2: MonoSDF training metrics presented for 200,000 and 50,000 iterations

Iterations number	200,000	50,000
Duration	6h 39m	1h 17m
Learning Rate	0.000049	0.00028
Eikonal Loss	0.0013	0.0017

PNSR	17.98	18.92
RGB Loss	0.030	0.036

According to the values yielded by each training and Table 2, combined with the final Meshes (Figure 70/Figure 1), it can be concluded that the training time for 200,000 iterations is significantly longer, reaching 6 hours and 39 minutes, compared to just 1 hour and 17 minutes for 50,000 iterations.

The learning rate in the 200,000-iteration model is much lower, at 0.000049, compared to 0.00028 in the 50,000-iteration model. A lower learning rate towards the end of training allows the model to optimize its parameters more carefully.

The Eikonal Loss is lower in the 200,000-iteration model, at 0.0013 compared to 0.0017, which suggests better surface smoothness and accuracy in the surface normalizations. This loss directly affects the quality of geometry and surface in the 3D reconstruction, so the lower value in the 200,000-iteration model indicates a better overall geometric representation. Surprisingly, the 50,000-iteration model has a higher PSNR, at 18.92 compared to 17.98 in the 200,000-iteration model. A higher PSNR typically indicates better image quality, suggesting that the 50,000-iteration model may produce cleaner or less noisy images.

The RGB Loss is lower in the 200,000-iteration model (0.030 versus 0.036 for the 50,000-iteration model), indicating better color fidelity in the generated images. This means that the 200,000-iteration model performs better at capturing the color details of the scene, which is important for evaluating the realism of the final 3D reconstructions.

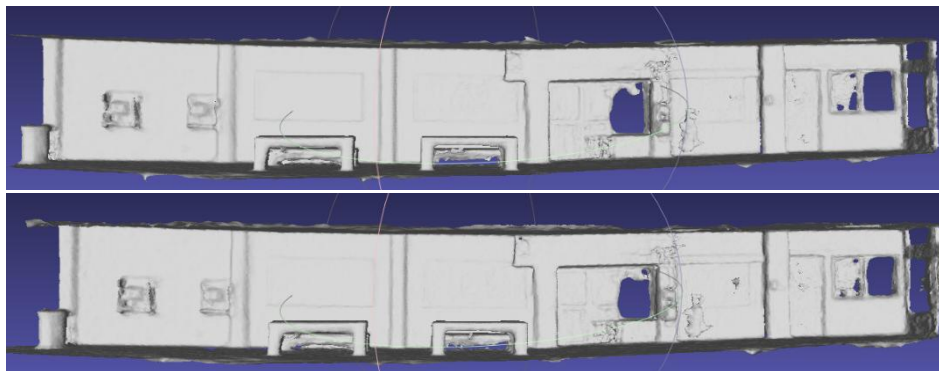


Figure 70: MonoSDF meshes after 200,000 iter. (up) & Mesh after 50,000 iter. (down)

Initially, it should be noted that all meshes have undergone perimeter noise removal to ensure the area of interest appears clearer and more directly. The geometry of the top mesh appears cleaner and overall less noisy. In contrast, the bottom mesh, with fewer iterations, shows more noise. Specifically, the top mesh exhibits less noise in open areas, while there is no significant differentiation in the geometry along the edges and corners. Regarding shape consistency, the differences in walls, windows, and other elements are minimal, especially in terms of edge sharpness.

Based on the above data and the analysis of the results, it is concluded that despite the improved performance of the training with 200,000 iterations, particularly in terms of surface and color accuracy, the difference in geometry quality and color rendering is not significant enough to justify the substantially longer training time. In fact, the PSNR value is higher for the 50,000 iterations, indicating that the resulting image is cleaner or has less noise. Given the importance of time for

conducting the tests, it was decided that the MonoSDF tests would be performed with 50,000 iterations.

9.1.2. UniSurf num iteration comparison

For UniSurf, in contrast to MonoSDF, iterations of 100,000 and 50,000 were compared based on Smoothness Loss, which controls the smoothness of surfaces. At the same time, performance in free spaces was compared through Freespace Loss to assess accuracy in predicting empty spaces and managing the geometric features of the scene. Additionally, RGB Loss was evaluated to examine the quality of the model's color rendering. (Yu, Chen, Antic, Peng, & Bhattacharyya, Supervision, 2022)

Table 3: UniSurf training metrics presented for 200,000 and 50,000 iterations

Iterations number	200,000	50,000
Duration	2h 9m	1h 27m
Learning Rate	0.00038	0.00047
Smoothness Loss	0.00061	0.00062
Freespace Loss	0.000064	0.000056
PNSR	13.69	23.18
RGB Loss	0.034	0.036

According to the values yielded by each training in Table 3 combined with the final Meshes (Figure 71) the training time for 200,000 iterations is longer, with a duration of 2 hours and 9 minutes, compared to 1 hour and 27 minutes for 50,000 iterations, which is expected, as more iterations require more time for optimization. The learning rate is slightly lower in the 50,000-iteration model (0.00047 vs. 0.00038). The Smoothness Loss is almost identical in both models, with values of 0.00061 and 0.00062, showing that both models perform similarly in generating smooth surfaces. The Freespace Loss is slightly better in the 50,000-iteration model (0.000056) compared to the 200,000-iteration model (0.000064).

The PSNR (Peak Signal-to-Noise Ratio) presents the biggest difference. The 50,000-iteration model has a higher PSNR (23.18) compared to the 200,000-iteration model (13.69), suggesting that the 50,000-iteration model produces sharper and less noisy images, possibly due to overfitting in the 200,000-iteration model. The RGB Loss is similar, with 0.034 for the 200,000-iteration model and 0.036 for the 50,000-iteration model, indicating a slight improvement in color reproduction in the former.

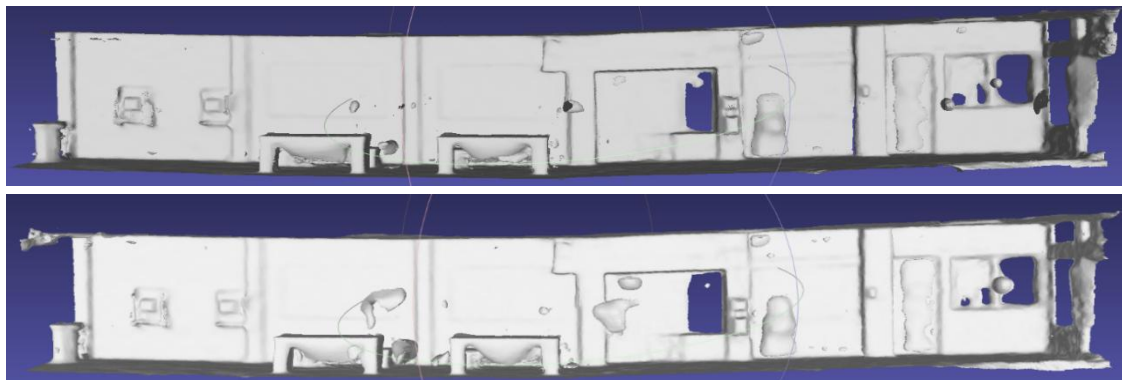


Figure 71: UniSurf meshes after 200,000 iter. (up) & Mesh after 50,000 iter. (down)

Regarding surface details, although the Freespace Loss had a better value in the 50,000-iteration model, the meshes show that this model presents more noise compared to the 200,000-iteration model. Additionally, the 200,000-iteration mesh displays fewer visible artifacts, especially in complex areas like benches. However, in certain surfaces, such as the phone booths and the corner of the wall on the right side of the scan, there are slight distortions or less accurate reconstructions. In terms of edge sharpness, the 200,000-iteration mesh captures more precise details, with sharper edges along the folds of the wall due to the columns.

In conclusion, each mesh has its advantages and disadvantages, with small differences overall. Since the differences are not particularly significant and because this thesis focuses on comparing the models as well as how the parameters affect the result, the iteration count will be maintained at 50,000 for this case, as well.

9.2. Selection of Parameter Values for Comparative Model Analysis

The selection of the parameters to be modified during training was made with the aim of examining the impact of each parameter on the final training process and the resulting mesh. In the Table 4, the parameter values that deviate from the default values are highlighted in red. These values were chosen to present significant deviations from the recommended ones, to make their effect on the training and the result more apparent.

Table 4: Parameter values for default settings, example cases and selected values for comparison in the training process

	Default values	Test values													
pipeline.model.sdf-field.use-grid-feature	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
pipeline.model.sdf-field.hidden-dim	256	256	256	64	256	256	256	256	256	256	256	256	256	256	256
pipeline.model.sdf-field.num-layers	8	2	2	2	8	2	2	2	2	2	2	2	2	2	2
pipeline.model.sdf-field.num-layers-color	4	2	2	2	2	8	2	2	2	2	2	2	2	2	2
pipeline.model.sdf-field.use-appearance-embedding	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
pipeline.model.sdf-field.geometric-init	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
pipeline.model.sdf-field.bias	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,05	0,8	0,8	0,8	0,8	0,8	0,8
pipeline.model.sdf-field.beta-init	0,1	0,3	0,3	0,3	0,3	0,3	0,3	0,3	0,3	0,8	0,3	0,3	0,3	0,3	0,3
trainer.steps-per-eval-image	500	5000	5000	5000	5000	5000	5000	5000	5000	5000	25	5000	5000	5000	5000
pipeline.model.sensor-depth-l1-loss-mult	0,0	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,8	0,1	0,1	0,1
pipeline.model.sensor-depth-freespace-loss-mult	0,0	10,0	10,0	10,0	10,0	10,0	10,0	10,0	10,0	10,0	10,0	10,0	0,0	10,0	10,0
pipeline.model.sensor-depth-sdf-loss-mult	0,0	6000,0	6000,0	6000,0	6000,0	6000,0	6000,0	6000,0	6000,0	6000,0	6000,0	6000,0	6000,0	0,0	6000,0
pipeline.datamanager.train-num-rays-per-batch	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	1024	2048

Specifically, by changing values such as hidden-dim from 256 to 64 or steps-per-eval-image from 500 to 25, the goal is to assess how these parameters influence training time, geometry accuracy, and the overall quality of the mesh. Parameters like beta-init, sensor-depth-freespace-loss-mult, and others were also selected to be tested through variations ranging from small to drastic, aiming to highlight their interactions with the model. Using both the default and modified values will allow for a comparative analysis that will show how each parameter affects the model's performance in 3D scene reconstruction, depth consistency, and surface smoothness. The tests were conducted on a computer with 16 physical and 24 logical cores, an NVIDIA GeForce RTX 3070 Ti graphics card, 62 GB of RAM, and 2 GB of swap memory (of which 1.3 GB was used and 711 MB was available), while available memory was 50 GB. These factors set the limits for training, especially for parameters that impact memory and storage, influencing the selection of parameter values for the models.

10. Experiments

This chapter, present the results from training the three models using 417 images of color and depth data. Of these, all were used for training, while 14 were used for evaluation. First, the results of all models for each modified parameter will be presented. Then, we will focus on each model individually and analyze how it is affected by the adjustment of each parameter, and finally an evaluation of the best mesh for each model in terms of ground truth will be applied. It is important to note that for the first part of the comparison the learning rate graph will only be presented once, as the flow of values remains essentially constant across all models and parameter tests and will only be discussed if the final value of the learning rate changes.

The evaluation and comparison of results for all models across different parameter adjustments will focus on the following criteria:

- Eval Images/img
- Eval Images/sensor_depth
- Duration
- PSNRS
- Learning Rate
- Sensor_l1_loss
- Sensor_freespace_loss
- Sensor_sdf_loss
- RGB_loss

For PSNR and RGB loss, the evaluation values (Eval Loss Dict/rgb_loss and Eval Metrics Dict/psnr) will be used, as they are considered the best metrics for assessing generalization and determining how well a model performs on new data. The “*Eval Loss Dict/rgb_loss*” reflects the loss incurred when reconstructing RGB images during model evaluation using unseen data (test or validation set), providing insight into the model's ability to generalize to new images. However, the so called “*Eval Metrics Dict/psnr*” is often used as the primary and comprehensive measure for comparison with other models or settings. It provides a much enhanced measure using many pictures giving a better estimate of the loss function as a measure of the model at reconstructing images. For example, while judging two models in terms of PSNR for an entire scene, the scores signify the relative degree of the average image quality of models. Based on these criteria, the performance of the models will be analyzed both at an overall level and for each modified parameter.

It is important to mention that all meshes have been roughly cleaned to give a preliminary view of how well each model differentiates details and reconstructs the scene without generating noise in free areas or challenging spots. All models can be further optimized to improve clarity by removing noise that doesn't belong to object regions.

10.1. Models Comparison and Results for Each Parameter Adjustment

Keeping the parameter values as proposed by the SDFStudio team, the three models Neus-Facto, MonoSDF and UniSurf show different behaviors and performances during the 3D scene reconstruction process. The training time differs significantly between them. Neus-Facto completes the process in only 12 minutes and 42 seconds, which makes it the fastest. MonoSDF takes much longer, about 1 hour, 17 minutes and 5 seconds, while UniSurf requires 1 hour, 27 minutes and 16 seconds, making it the slowest. However, it should be noted that the number of iterations for the Neus-Facto is 20,000, while for the other two models it is 50,000.

Table 5: Training metrics for suggested parameter values across Neus-Facto, MonoSDF, and UniSurf models

Parameter: Suggested values			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	12m 42s	1h 17m 5s	1h 27m 16s
PNSR	16.23	18.92	13.64
Learning Rate	0.000025	0.00028	0.00047
Sensor_l1_loss	0.004559	0.005400	0.022068
Sensor_freespace_loss	0.000064	0.000111	0.000056
Sensor_sdf_loss	0.007961	0.009285	0.010082
RGB_loss	0.108155	0.079607	0.149631

According to Keeping the parameter values as proposed by the SDFStudio team, the three models Neus-Facto, MonoSDF and UniSurf show different behaviors and performances during the 3D scene reconstruction process. The training time differs significantly between them. Neus-Facto completes the process in only 12 minutes and 42 seconds, which makes it the fastest. MonoSDF takes much longer, about 1 hour, 17 minutes and 5 seconds, while UniSurf requires 1 hour, 27 minutes and 16 seconds, making it the slowest. However, it should be noted that the number of iterations for the Neus-Facto is 20,000, while for the other two models it is 50,000.

Table 5 regarding the losses, there is a clear distinction between the models. The L1 sensor loss for Neus-Facto and MonoSDF is almost identical, approximately 0.004559 and 0.005400, respectively. In contrast, UniSurf L1 sensor loss amounts to a much higher value of 0.022068, implying that its depth predictions are more far from reality, which corroborates with its lower PSNR. On the other hand, UniSurf has the least sensor freespace loss of 0.000056, while MonoSDF has the highest at 0.000111. Despite this, the SDF sensor loss is higher for UniSurf at 0.010082, followed by MonoSDF and Neus-Facto. This then reveals the weakness of UniSurf for correct surface geometry representation in a scene.

The RGB loss, with the highest value for UniSurf at 0.149631, indicates that the model struggles to align synthetic images with real ones. MonoSDF has a corresponding RGB loss of 0.079607, whereas Neus-Facto had a loss of 0.108155. Obviously, MonoSDF is in the lead. From the point of view of image quality, regarding PSNR (Figure 72), MonoSDF can be highlighted for its high PSNR. This can be interpreted to mean that MonoSDF provides high-quality images with low noise. For Neus-Facto, the PSNR is 16.23, which is still quite a decent number, reflecting good image quality. The lowest value of PSNR is for UniSurf among all three models, which indicates that their reconstructions have more noise and lower quality.

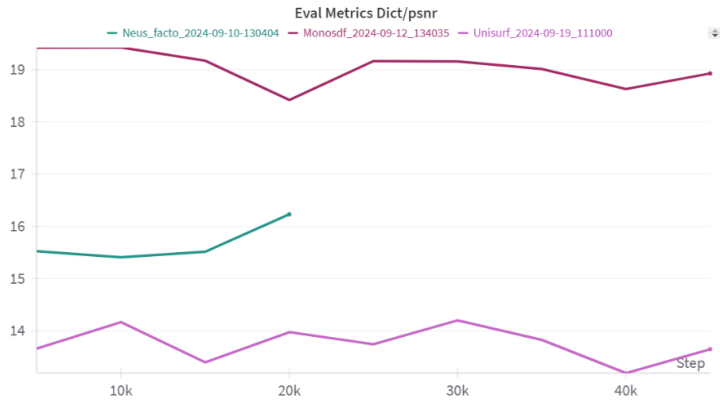


Figure 72: PSNR Evaluation Metrics Across Models (Suggested Parameter Values)

Learning rate (Figure 73) plays a crucial role in how these models adapt during training. Neus-Facto converges to the lowest learning rate, after a rapid decrease, as shown in the graphs. After approximately 20,000 steps, the learning rate stabilizes, indicating a cautious approach to parameter adjustment. This rapid decrease suggests that Neus-Facto aims to stabilize early in training, avoiding overfitting or drastic changes as training progresses. In contrast, MonoSDF converges to a higher learning rate, which decreases more gradually over time. This gradual decrease allows MonoSDF to continue learning at a measured pace throughout training, which may explain why it achieves the highest PSNR. It continues refining details for a longer period, enabling it to produce higher-quality reconstructions. UniSurf, with the highest final learning rate of 0.000047, maintains a more steady and consistent learning rate throughout training. However, this relatively high and stable rate may lead to less precise parameter updates, contributing to the lower PSNR and less accurate reconstructions.

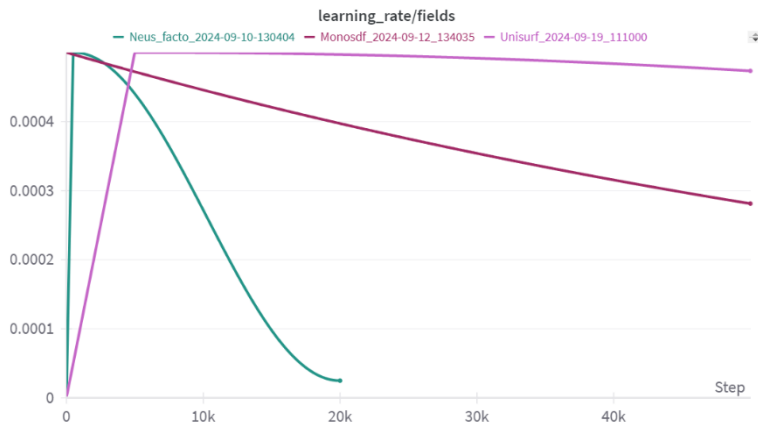


Figure 73: Learning Rate Progression Across Models (Suggested Parameter Values)

The quality assessment of the depth sensor and image (Figure 74) shows that both Neus-Facto and MonoSDF have much sharper and more detailed depth maps and images, while UniSurf creates a much darker and less detailed depth map, and that can be one of the reasons for its higher sensor

losses. Visually, MonoSDF and Neus-Facto perform better than UniSurf in generating sharper and finer detailed images.

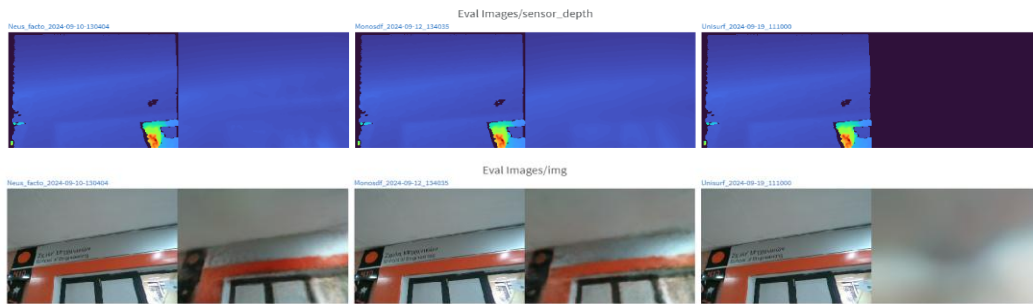


Figure 74: Sensor Depth and Image Evaluation Results (Suggested Parameter Values)

According to the meshes (Figure 75) MonoSDF stands out as the best model in terms of image quality and reconstruction accuracy, although it requires more training time. This is evident from its precise representation of the scene, particularly in challenging areas such as built-in benches, corner folds in the walls, and objects like doors and windows. The model manages to capture these details due to its gradually decreasing learning rate, which allows it to keep improving over time. Neus-Facto offers a faster alternative with decent performance, thanks to its rapidly decreasing learning rate, which stabilizes the model early on. While it doesn't reach the same level of detail as MonoSDF, it provides a good balance between speed and performance. In contrast, UniSurf struggles with both accuracy and image quality due to its relatively high and steady learning rate, which may hinder the model's effective convergence. Although its reconstruction is less detailed and noisier, it performs better in certain clearer areas, such as the telephone booths and regions with sparse information, like the right sides of the two columns located on the right side of the scene. Finally, noise is observed in the empty areas across all models, with the most significant amount appearing in UniSurf.



Figure 75: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) using suggested parameter values.

With the parameter `pipeline.model.sdf-field.use-grid-feature` set to `False`, significant performance changes are observed in the three models Neus-Facto, MonoSDF, and UniSurf, as shown

in Table 6: Training metrics with pipeline.model.sdf-field.use-grid-feature set to False. These changes affect various aspects of the models' behavior and accuracy, highlighting the impact of this parameter on the quality of the 3D reconstructions and overall performance.

Table 6: Training metrics with pipeline.model.sdf-field.use-grid-feature set to False

Parameter: pipeline.model.sdf-field.use-grid-feature False			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	10m 22s	1h 17m 27s	12m 42s
PNSR	18.30	17.74	15.71
Learning Rate	0.000025	0.00028	0.00047
Sensor_l1_loss	0.003225	0.003524	0.022065
Sensor_freespace_loss	0.000112	0.000251	0.000030
Sensor_sdf_loss	0.009417	0.009345	0.011510
RGB_loss	0.085408	0.090455	0.121937

Initially, the training time differs, with Neus-Facto remaining the fastest and UniSurf having a similar completion time, while MonoSDF continues to require more time. The increased speed in both Neus-Facto and UniSurf compared to previous results suggests that not using the grid feature to represent the 3D space can accelerate training, particularly in UniSurf.

According to the PSNR values, Neus-Facto takes the lead this time, with MonoSDF following closely at 17.74. This difference indicates that Neus-Facto manages to improve image quality when the grid-feature parameter is disabled, while UniSurf continues to struggle with the quality of its results. The sensor L1 loss remains comparable between Neus-Facto and MonoSDF, while UniSurf once again lags in accurately predicting depth values.

The sensor freespace loss is low across all three models, with MonoSDF and Neus-Facto showing slightly higher values compared to UniSurf, which has the lowest. For sensor SDF loss, UniSurf reports the highest value, indicating more difficulties in surface modeling compared to Neus-Facto and MonoSDF. The RGB loss remains significantly higher for UniSurf, while Neus-Facto and MonoSDF have lower values. In terms of depth map and image reconstruction evaluation (Figure 76), Neus-Facto and MonoSDF produce more reliable depth maps and RGB images, though they exhibit more blurriness and reduced accuracy. On the other hand, while UniSurf still struggles with depth map generation, in this case, it produces better results in the RGB images.

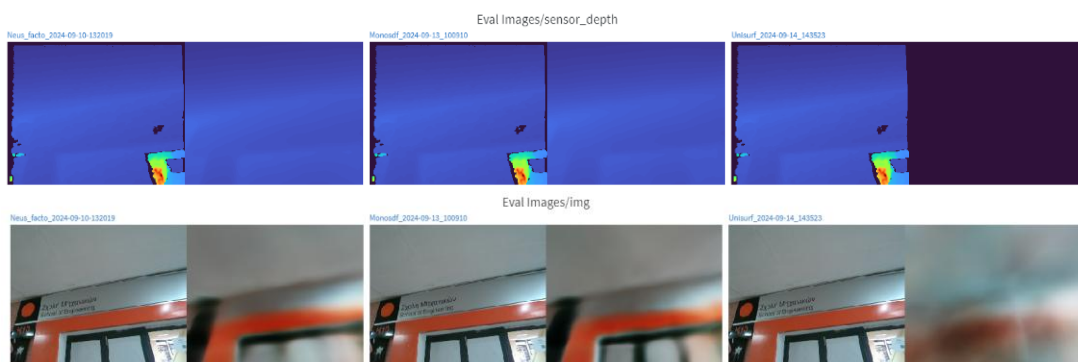


Figure 76: Sensor Depth and Image Evaluation Results with pipeline.model.sdf-field.use-grid-feature set to False

Disabling the grid feature has interesting effects on the final reconstruction of the scene (Figure 77). Specifically, it appears to negatively impact all models, as although the noise in the empty areas has been almost entirely reduced, this comes at the cost of significant detail loss in the reconstructions. One positive outcome is the clearer representation around the benches, but the meshes overall show less sharpness and a more generalized depiction of the scene's geometry. Instead of the detailed and clear surfaces observed previously, disabling the grid feature results in a smoother, but less accurate representation of the scene.

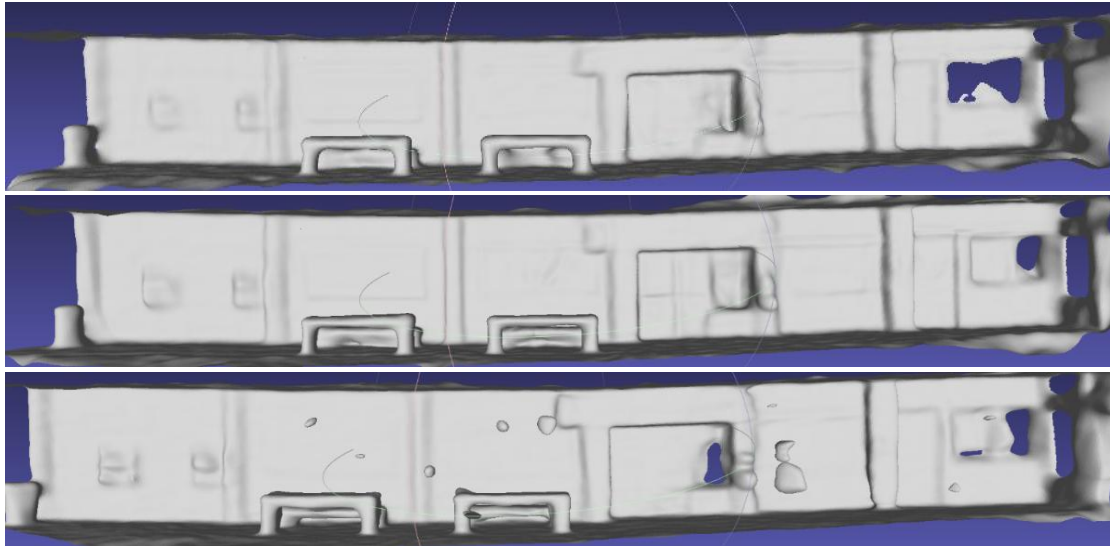


Figure 77: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with `pipeline.model.sdf-field.use-grid-feature` set to `False`

By setting the number of neurons per layer to 64 instead of the initial 256, initial changes are observed in the performance of the three models Neus-Facto, MonoSDF and UniSurf both in terms of training time and reconstruction quality. This alteration in the hidden layer dimension impacts the computational complexity, consequently affecting the training time and the accuracy of the results (Table 7). Notably, the training duration is significantly reduced across all models compared to previous settings. As expected, the reduction in hidden layer size appears to accelerate the training process.

Table 7: Training metrics for models with `pipeline.model.sdf-field.hidden-dim` set to 64

Parameter: pipeline.model.sdf-field.hidden-dim 64			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	9m 47s	41m 5s	44m 1s
PNSR	18.37	18.55	13.45
Learning Rate	0.000025	0.00028	0.00047
Sensor_l1_loss	0.004980	0.006291	0.022566
Sensor_freespace_loss	0.000045	0.000031	0.000052
Sensor_sdf_loss	0.007499	0.007195	0.009873
RGB_loss	0.086908	0.084299	0.156266

Image quality, as measured by PSNR, Neus-Facto and MonoSDF show fairly good values compared to UniSurf, which continues to lag behind. The learning rate follows the same trend as in previous experiments, along with the other performance indicators.

The visual comparison of depth maps and reconstructed images shows similar results to those where the parameters were kept as initially proposed (Figure 78). The reconstructed images from Neus-Facto and MonoSDF retain more detail and clarity, with MonoSDF producing sharper images, while UniSurf remains blurry and of much lower quality.

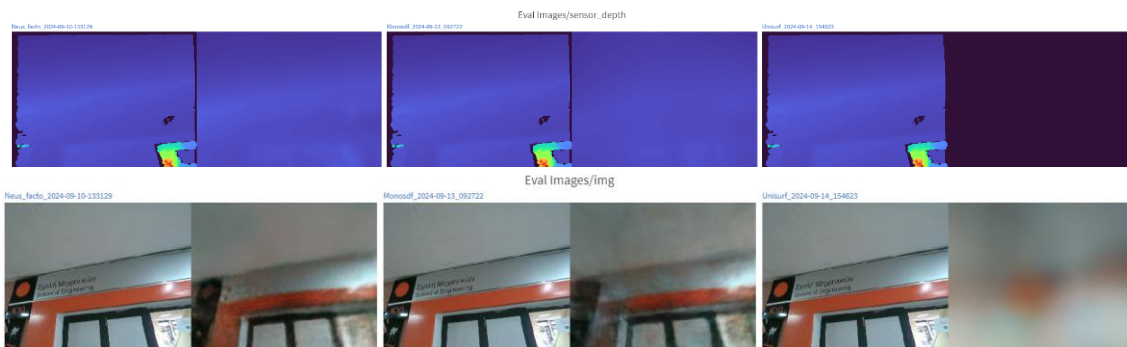


Figure 78: Sensor Depth and Image Evaluation Results with `pipeline.model.sdf-field.hidden-dim` set to 64

In the 3D reconstruction, all models exhibit considerable scattered noise on almost all surfaces of the scene (Figure 79). However, UniSurf defines the geometry of the object on the walls more effectively, despite the values discussed earlier. It is important to note that in the area on the right side of the object, where light incidence was stronger and the other two models struggled to accurately represent the geometry, UniSurf performed significantly better.

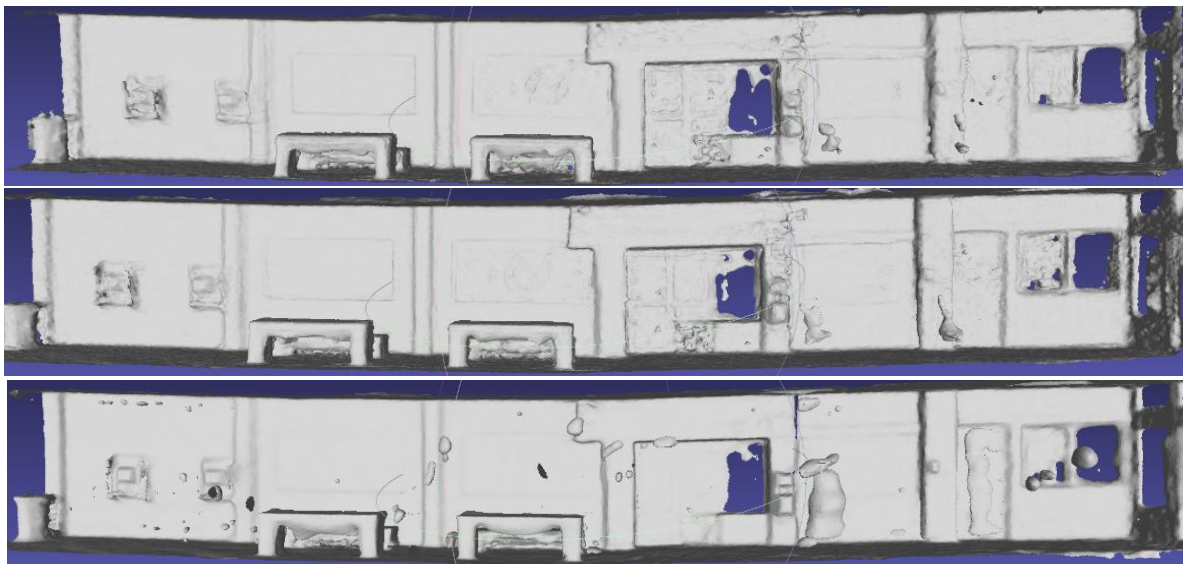


Figure 79: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with `pipeline.model.sdf-field.hidden-dim` set to 64

With the number of layers increased to 8 from 2, it's important to note that UniSurf encountered a memory issue (CUDA out of memory), indicating it was unable to complete the training due to system resource constraints.

Table 8: Training metrics for models with pipeline.model.sdf-field.num-layers set to 8

Parameter: pipeline.model.sdf-field.num-layers 8			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	23m 4s	2h 25m 17s	CUDA out of memory
PNSR	17.19	18.59	
Learning Rate	0.000025	0.000281	
Sensor_l1_loss	0.006033	0.005068	
Sensor_freespace_loss	0.000095	0.000049	
Sensor_sdf_loss	0.008570	0.008855	
RGB_loss	0.098429	0.081279	

According to Table 8 the training duration nearly doubled for the two models that completed training. In terms of image quality, MonoSDF maintains its lead with PSNR, though the trend in the graph shows a decline in both models during training (Figure 80). The sensor L1 loss remains low for both models, as does the sensor freespace loss. Both models exhibit similar performance in surface modelling and color rendering, according to the sensor SDF loss and RGB loss values, respectively.

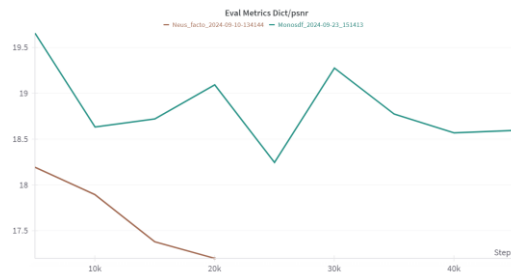


Figure 80: PSNR Evaluation Metrics Across Models with pipeline.model.sdf-field.num-layers set to 8

The depth maps and reconstructed images (Figure 81) show that Neus-Facto produces slightly more accurate depth maps, as well as less blurry RGB images with better precision.

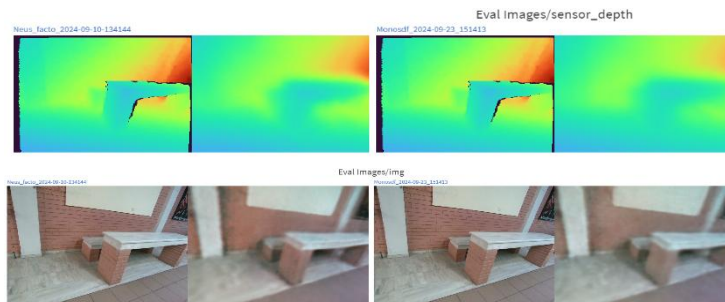


Figure 81: Sensor Depth and Image Evaluation Results with pipeline.model.sdf-field.num-layers set to 8

In the 3D reconstruction, both Neus-Facto and MonoSDF produce well-structured surfaces (Figure 82). However, MonoSDF delivers clearer surfaces with better detail and smoothness, while Neus-Facto shows slightly lower detail.

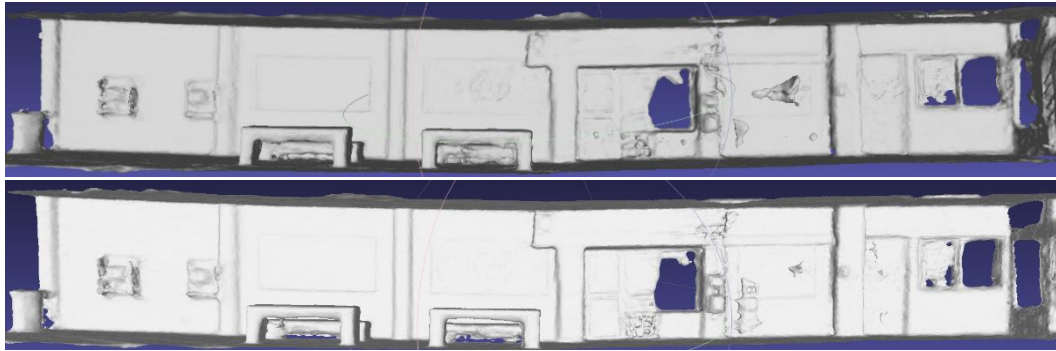


Figure 82: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline.model.sdf-field.num-layers set to 8

With the increase in the number of hidden layers from 2 to 8, the performance changes in the Neus-Facto, MonoSDF, and UniSurf models do not show significant differences, except for the “sensor freespace loss” value in MonoSDF, as shown in Table 9. This value is nearly half compared to the corresponding value when the parameter was set to 2.

Table 9: Training metrics for models with pipeline.model.sdf-field.num-layers-color set to 8

Parameter: pipeline.model.sdf-field.num-layers-color 8			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	15m 41s	1h 30m 44s	1h 25m 57s
PNSR	16.35	18.00	14.71
Learning Rate	0.000025	0.00028	0.00047
Sensor_l1_loss	0.004599	0.005965	0.023176
Sensor_freespace_loss	0.000053	0.000059	0.000053
Sensor_sdf_loss	0.009171	0.009152	0.010658
RGB_loss	0.103656	0.082842	0.131047

Although increasing the number of color layers was theoretically expected to improve the detail and accuracy of the color reconstructions, the reconstructed images (Figure 83) for each model did not show the anticipated improvements. Specifically, having more hidden layers in the color network was supposed to allow the neural network to better "understand" the nuances of lighting, color shading, and material properties in the reconstructed images. This includes more accurately capturing subtle changes in color that depend on different viewing angles or scene lighting.

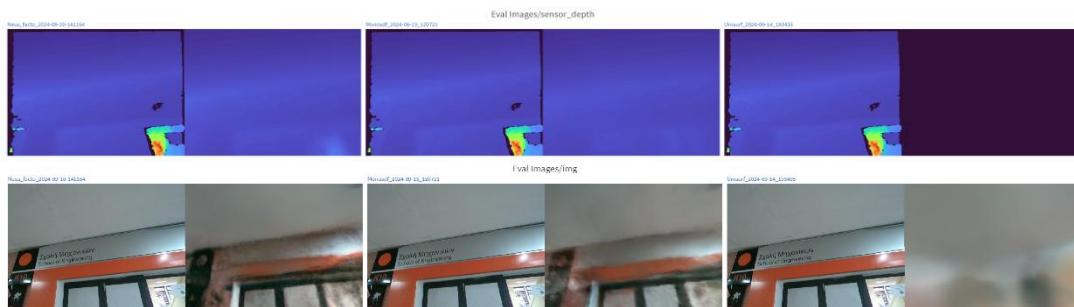


Figure 83: Sensor Depth and Image Evaluation Results with pipeline.model.sdf-field.num-layers-color set to 8

The results of MonoSDF and UniSurf are a little bit lower in quality in the case of the 3D reconstruction compared to the previous tests (Figure 84). Because, in a theoretical way, increasing the number of color layers could improve model performance by allowing more realistic, more detailed, visually accurate color representation capturing fine changes in surface appearance, it was not reflected in these results. Contrary to expectations, the models did not present any significant improvements, probable due either to overtraining or difficulties in learning from these complex patterns. Increased aggregate gain by the number of color layers did nothing to significantly enhance the level of detail and quality, often giving rise to more noise, especially in more complex areas of a scene.

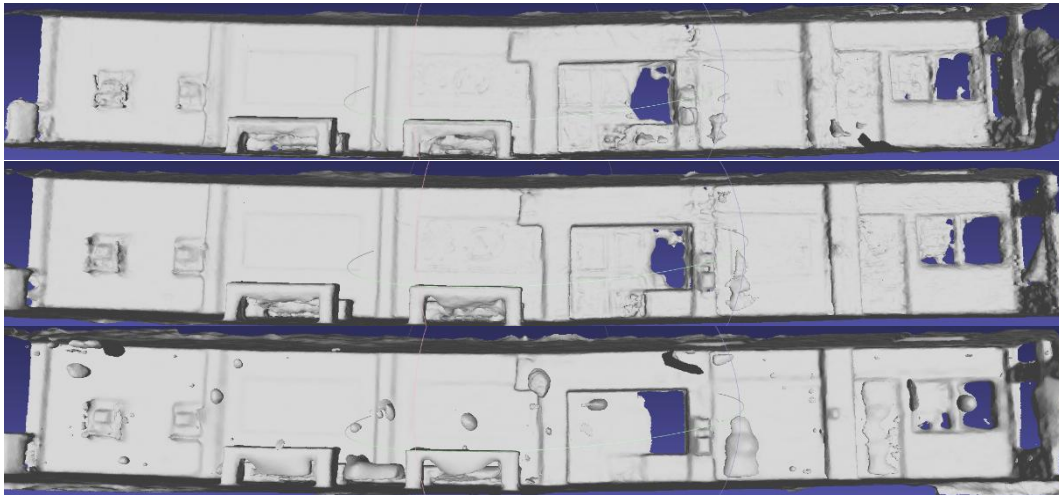


Figure 84: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with `pipeline.model.sdf-field.num-layers-color` set to 8

By disabling the appearance embedding parameter, the model relies primarily on spatial information, making it incapable of accurately capturing color variations caused by different viewing angles or lighting conditions. As with the other training metrics, the final PSNR values for all models remain almost unchanged (Table 10), with MonoSDF exceeding a value of 20, demonstrating its superior performance.

Table 10: Training metrics for models with `pipeline.model.sdf-field.use-appearance-embedding` set to False

Parameter: pipeline.model.sdf-field.use-appearance-embedding False			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	12m 22s	1h 7m 38s	1h 4m 51s
PSNR	17.44	21.74	14.29
Learning Rate	0.000025	0.00028	0.00047
Sensor_l1_loss	0.005658	0.006866	0.022067
Sensor_freespace_loss	0.000084	0.000086	0.000070
Sensor_sdf_loss	0.007077	0.007635	0.008215
RGB_loss	0.084078	0.052206	0.138686

It is also noteworthy that the PSNR values stay consistent throughout training, without significant fluctuations, indicating a stable and predictable learning process across the models (Figure

85). Depth and color reconstructions remain consistent with the results obtained from the recommended parameters, with UniSurf still failing to produce clear results.

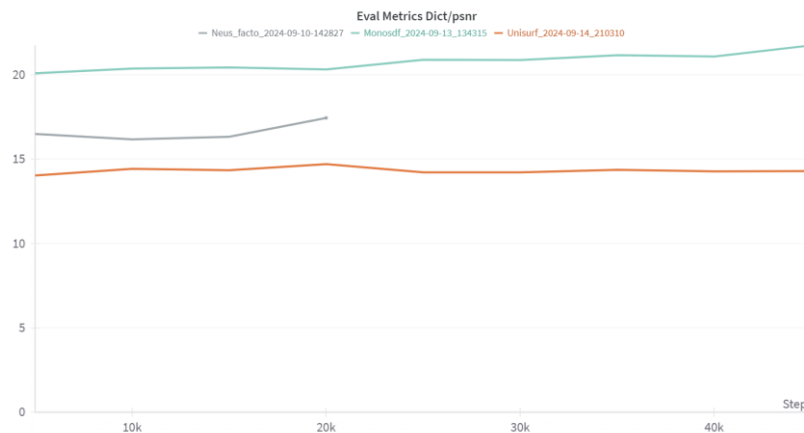


Figure 85: PSNR Evaluation Metrics Across Models with `pipeline.model.sdf-field.use-appearance-embedding` set to `False`

In the 3D reconstructions, MonoSDF remains the top-performing model, delivering cleaner and more detailed surfaces. Neus-Facto shows more noise and less accuracy in certain areas, while UniSurf, despite reduced noise, struggles to capture details and exhibits issues in many regions (Figure 86). Overall, MonoSDF provides better geometry and clearer results, while UniSurf maintains reduced noise in areas where the other models face difficulties.

Disabling appearance embedding negatively affects the quality of color reconstruction and detail in all models, with MonoSDF being the least affected and retaining its superiority in both quality and geometry.

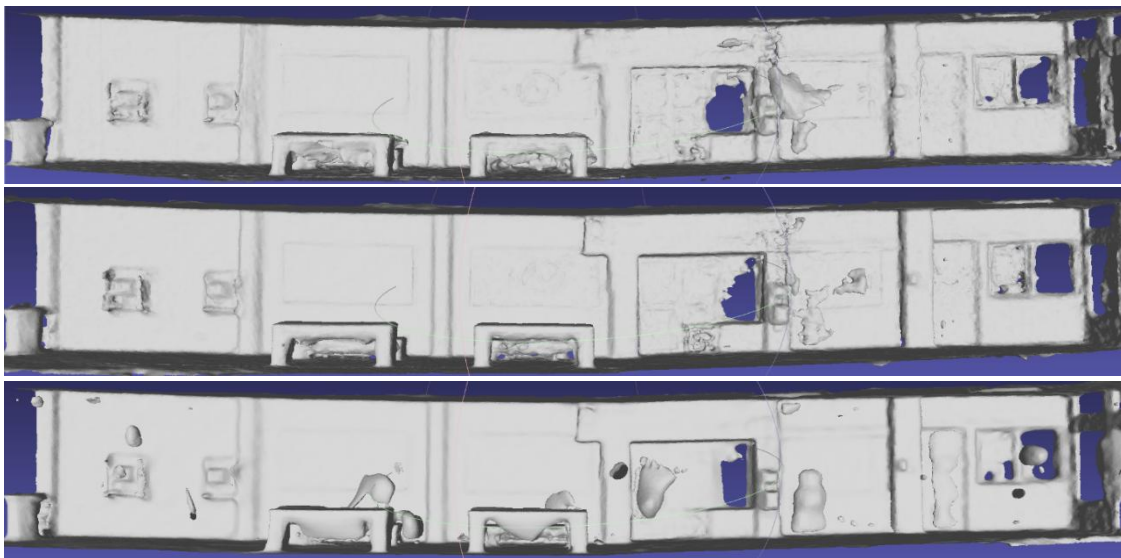


Figure 86: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with `pipeline.model.sdf-field.use-appearance-embedding` set to `False`

The geometric initialization parameter helps the model start with an initial geometric approximation, improving performance in the early stages of training and ensuring better

convergence in complex geometries. When it is disabled (set to False), the models rely more on random initial values, which can lead to difficulties in representing fine details.

A significantly reduced RGB loss value is observed for UniSurf (Table 11), while the reconstructed depth and color images show results like previous trials. The 3D reconstructions display significant noise in UniSurf, particularly in areas where geometric initialization had previously produced better results. Noise is also present in empty spaces, further deteriorating reconstruction quality.

Table 11: Training metrics for models with `pipeline.model.sdf-field.geometric-init` set to False

Parameter: <code>pipeline.model.sdf-field.geometric-init</code> False			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	12m 20s	1h 22m 39s	1h 4m 58s
PNSR	17.81	18.82	14.29
Learning Rate	0.000025	0.00028	0.00047
Sensor_l1_loss	0.003910	0.003344	0.023506
Sensor_freespace_loss	0.000063	0.000072	0.000100
Sensor_sdf_loss	0.008228	0.008220	0.009263
RGB_loss	0.087998	0.071867	0.000473

However, it is noteworthy that the 3D reconstructions of the other two models show better results compared to when geometry initialization was enabled (Figure 87). The improvement is evident both in reducing noise in complex object areas and in minimizing noise in empty regions. Once again, the mesh produced by MonoSDF stands out with the best results, offering superior detail and clarity in the reconstruction.



Figure 87: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with `pipeline.model.sdf-field.geometric-init` set to False

The decrease of the bias parameter from 0.8 to 0.05 decreased SDF loss in all three models (Table 12). This controls the initial bias of the model for the computed distance from surfaces. With a low bias, the model tends to more aggressively approach surfaces. This behavior might contribute

toward improving performance with respect to SDF loss but also increases the chances that some areas will produce errors or noise.

Table 12: Training metrics for models with pipeline.model.sdf-field.bias set to 0.05

Parameter: pipeline.model.sdf-field.bias 0.05			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	12m 23s	1h 10m 6s	1h 5m 11s
PNSR	18.38	19.12	12.98
Learning Rate	0.000025	0.00028	0.00047
Sensor_l1_loss	0.006976	0.006126	0.022078
Sensor_freespace_loss	0.000066	0.000056	0.000094
Sensor_sdf_loss	0.005277	0.008455	0.007574
RGB_loss	0.088918	0.077923	0.165009

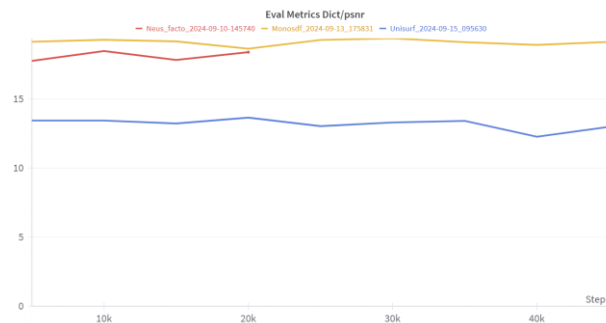


Figure 88: PSNR Evaluation Metrics Across Models with pipeline.model.sdf-field.bias set to 0.05

While the PSNR values remain in a steady state, it also indicates good overall learning across models (Figure 88). The reconstructions of depth and color do not exhibit major changes from previous trials, thereby indicating stability in results. In some cases, the 3D reconstructions may even seem to have surface outlines that are a little sharper owing to the adjustment in the bias parameter. Despite this, reduced accuracy with increased noise is observed in the 3D reconstructions across all models (Figure 89).



Figure 89: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with `pipeline.model.sdf-field.bias` set to 0.05

The beta initialization parameter controls the initial value of beta in the surface smoothing mechanism used by the model to converge surfaces based on the calculated distance from them (SDF).

Table 13: Training metrics for models with `pipeline.model.sdf-field.beta-init` set to 0.8

Parameter: <code>pipeline.model.sdf-field.beta-init</code> 0.8			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	12m 11s	1h 14m 32s	1h 4m 57s
PNSR	18.51	19.049	13.90
Learning Rate	0.000025	0.00028	0.00047
Sensor_l1_loss	0.003129	0.005422	0.022069
Sensor_freespace_loss	0.000098	0.000099	0.000066
Sensor_sdf_loss	0.007734	0.009323	0.008645
RGB_loss	0.083095	0.080265	0.148282

By increasing the value from 0.3 to 0.8, the model tends to enforce stricter surface smoothing, resulting in clearer and more accurate surfaces. This adjustment helps reduce noise and enhances detail in surface areas, as beta controls how "sharp" or "smooth" the reconstructed surfaces are. With a higher beta-init value, the model becomes more efficient at producing well-defined surfaces and maintaining precise geometric representation. The results of the reconstruction confirm this improvement. The reconstructed depth and color images of Neus-Facto are the most accurate so far among all models (Figure 90). In the color images, even small details such as the letters on a sign are clearly visible, demonstrating a significant enhancement in the quality of the visual representation.

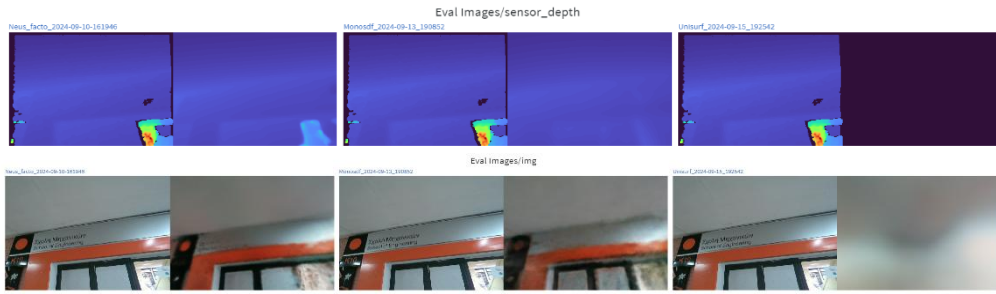


Figure 90: Sensor Depth and Image Evaluation Results with `pipeline.model.sdf-field.beta-init` set to 0.8

The 3D reconstruction of Neus-Facto also shows excellent results (Figure 91), with minimal noise and well-defined areas. Although it loses some accuracy and information in regions with changing depth, the overall outcome is impressive, providing an outstanding representation with very few issues.



Figure 91: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with `pipeline.model.sdf-field.beta-init` set to 0.8

By changing the image evaluation step parameter from 5000 to 25, the model evaluates its performance much more frequently, every 25 steps instead of every 5000. This adjustment provides quicker feedback during training, allowing for faster detection of potential issues such as overfitting or underfitting. It also allows for more detailed monitoring of the model's progress, although this increases computational cost, significantly slowing down the overall training speed across all three models (Table 14).

Table 14: Training metrics for models with `trainer.steps-per-eval-image` set to 25

Parameter: <code>trainer.steps-per-eval-image</code> 25			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	55m 36s	6h 36m 30s	6h 6m 12s

PNSR	15.93	18.78	14.04
Learning Rate	0.000025	0.00028	0.00047
Sensor_l1_loss	0.004573	0.005588	0.022066
Sensor_freespace_loss	0.000066	0.000148	0.000053
Sensor_sdf_loss	0.007881	0.008639	0.010623
RGB_loss	0.110600	0.077244	0.144039

Despite the increased monitoring, this change did not significantly improve the results for the reconstructed depth and color images, nor did it enhance the final meshes of the models in a satisfactory or meaningful way. Therefore, while more frequent evaluation provides greater control, it did not lead to the expected improvements in the quality of the final reconstructions (Figure 92).

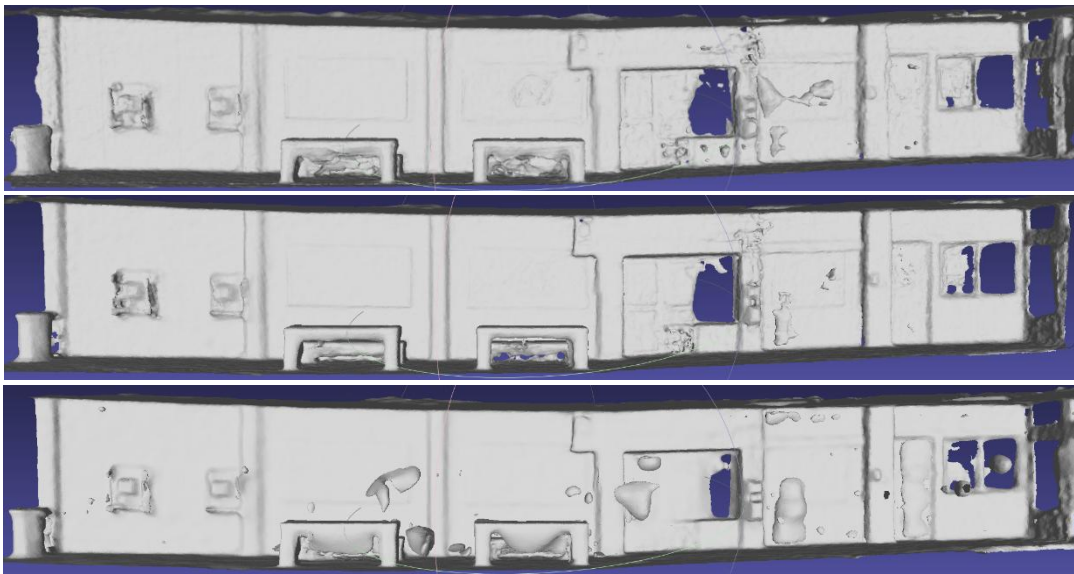


Figure 92: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with `trainer.steps-per-eval-image` set to 25

Increasing the depth l1 loss parameter from 0.1 to 0.8 significantly affects the model's performance. This change emphasizes depth accuracy by increasing the weight of L1 loss for depth (Table 15), which leads to more precise depth reconstructions (Figure 94).

Table 15: Training metrics for models with pipeline. `model.sensor-depth-l1-loss-mult` set to 0.8

Parameter: pipeline. model.sensor-depth-l1-loss-mult 0.8			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	12m 13s	1h 26m 15s	1h 5m 14s
PNSR	18.33	18.22	17.00
Learning Rate	0.000025	0.00028	0.00047
Sensor_l1_loss	0.022290	0.025789	0.055985
Sensor_freespace_loss	0.000074	0.000186	0.000037
Sensor_sdf_loss	0.007171	0.006597	0.009859
RGB_loss	0.084911	0.084470	0.090278

As a result, UniSurf, which typically starts with lower PSNR values, experiences a noticeable improvement after the 25,000th iteration (Figure 93).

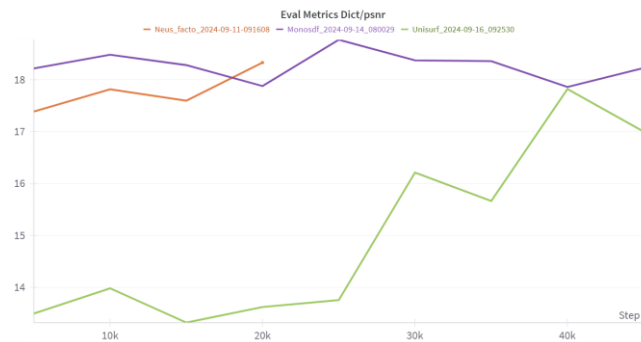


Figure 93: PSNR Evaluation Metrics Across Models with pipeline. model. sensor-depth-l1-loss-mult set to 0.8

Both Neus-Facto and MonoSDF exhibit superior depth and color reconstructions, with minimal noise in the final 3D reconstructions (Figure 95).



Figure 94: Sensor Depth and Image Evaluation Results with pipeline. model. sensor-depth-l1-loss-mult set to 0.8

The final mesh from Neus-Facto captures the scene's geometry more accurately, despite some minor discrepancies. This increased focus on depth accuracy results in clearer surfaces and a more detailed overall reconstruction.



Figure 95: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with pipeline. model. sensor-depth-l1-loss-mult set to 0.8

With the depth freespace loss parameter set to 0 instead of 10, huge changes in model performance are expected, particularly in how the models handle free space. The Sensor Freespace Loss has been eliminated, meaning that the model no longer considers free space information during training.

Table 16: Training metrics for models with pipeline.model.sensor-depth-freespace-loss-mult 0

Parameter: pipeline.model.sensor-depth-freespace-loss-mult 0			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	12m 23s	1h 16m 28s	1h 5m 26s
PNSR	15.44	18.86	13.37
Learning Rate	0.000025	0.00028	0.00047
Sensor_l1_loss	0.004730	0.005571	0.022100
Sensor_freespace_loss	0	0	0
Sensor_sdf_loss	0.007803	0.009499	0.002546
RGB_loss	0.114227	0.080006	0.157978

Despite the absence of freespace loss for all models, the final PSNR values remain at similar levels to previous tests, with a stable trend throughout training (Figure 96).

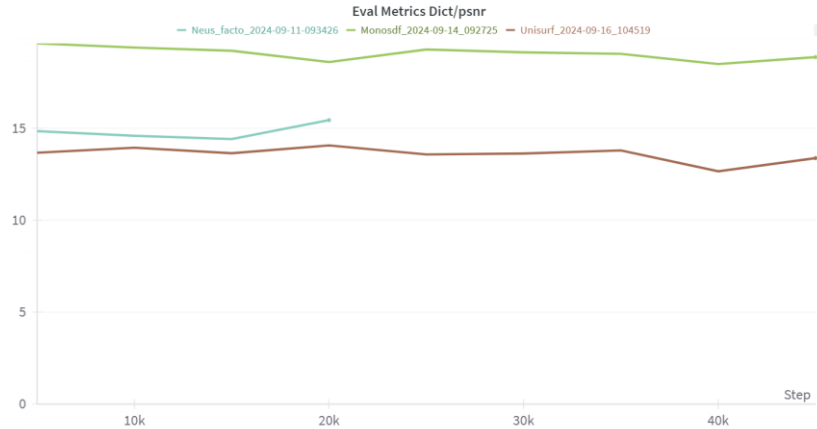


Figure 96: PSNR Evaluation Metrics Across Models with `pipeline.model.sensor-depth-freespace-loss-mult 0`

The reconstructions for Neus-Facto and MonoSDF are still satisfactorily good, although some regions exhibit more noise. In contrast, UniSurf shows significant noise in the 3D reconstruction, especially in areas that should typically be empty (Figure 97). The zero value for this parameter leads to less accurate reconstructions in free space areas, particularly for UniSurf. While models like MonoSDF and Neus-Facto maintain satisfactory performance, UniSurf struggles more without the freespace loss guiding its reconstructions.



Figure 97: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with `pipeline.model.sensor-depth-freespace-loss-mult 0`

With the depth SDF loss parameter set to 0 instead of 6000, we observe some noteworthy changes in model performance. This parameter controls how important the SDF loss is for depth accuracy, so reducing it to zero means the model no longer accounts for distance error during training, potentially leading to less accurate surface representation.

Table 17: Training metrics for models with pipeline.model.sensor-depth-sdf-loss-mult set to 0

Parameter: pipeline.model.sensor-depth-sdf-loss-mult 0			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	12m 20s	1h 21m 25s	1h 5m 33s
PNSR	19.19	18.75	19.00
Learning Rate	0.000025	0.00028	0.00047
Sensor_l1_loss	0.003898	0.004943	0.003641
Sensor_freespace_loss	0.000068	0.000070	0.000009
Sensor_sdf_loss	0	0	0
RGB_loss	0.077764	0.081114	0.075109

For the first time in these parameter trials, all models show similar PSNR values, with UniSurf exhibiting significant improvements in depth and color reconstructions (Figure 98), even surpassing Neus-Facto and MonoSDF in some areas.

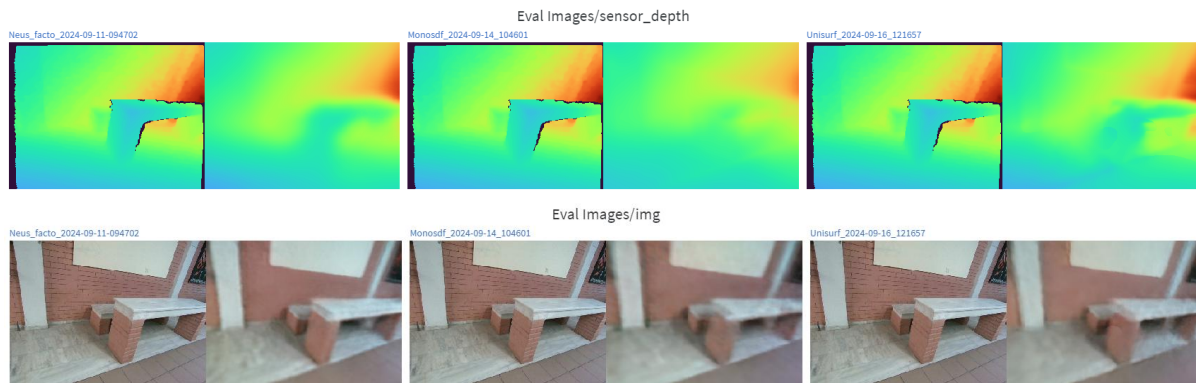


Figure 98: Sensor Depth and Image Evaluation Results with pipeline.model.sensor-depth-sdf-loss-mult set to 0

Despite these improved results in 2D depth and color images, this does not translate to better quality in the meshes, as all models produced extremely low accuracy meshes. While UniSurf performs well with 2D data, it fails to represent the scene adequately in 3D reconstructions, resulting in inaccurate and noisy outputs (Figure 99). The reduced mesh accuracy is directly related to the deactivation of the sensor-depth-sdf-loss-mult parameter. This parameter helps each model emphasize the correct representation of surfaces via SDF distances, improving the estimation of the distance between points and surfaces.

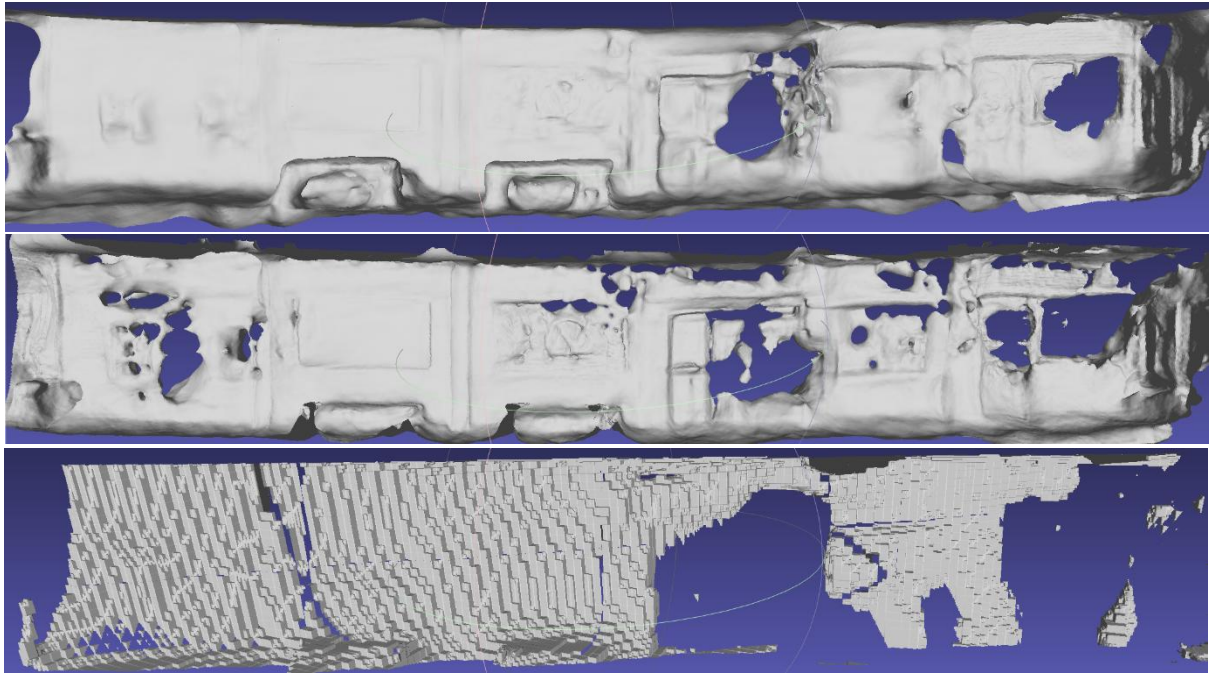


Figure 99: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with `pipeline.model.sensor-depth-sdf-loss-mult` set to 0

When the rays number per batch parameter is increased from 1024 to 2048, we expect a longer training time due to the increased number of rays processed in each batch. This change requires more computational resources, which leads to slower completion of training (Table 18).

Table 18: Training metrics for models with `pipeline.datamanager.train-num-rays-per-batch` set to 2048

Parameter: pipeline.datamanager.train-num-rays-per-batch			
2048			
Model	Neus-Facto	MonoSDF	UniSurf
Duration	21m 35s	2h 5m 33s	1h 58m 16s
PNSR	15.72	18.24	13.537
Learning Rate	0.000025	0.00028	0.00047
Sensor_l1_loss	0.004199	0.005552	0.023152
Sensor_freespace_loss	0.000100	0.000041	0.000069
Sensor_sdf_loss	0.007406	0.007728	0.008874
RGB_loss	0.114496	0.087055	0.155125

In the results, the increased number of rays does not cause significant changes in other metrics, such as PSNR or sensor losses, while the reconstructed depth and color images, as well as the meshes, remain consistent with previous trials. Once again, MonoSDF stands out by representing the scene's geometry with greater accuracy, providing a clearer and more precise reconstruction compared to the other models (Figure 100).



Figure 100: 3D reconstruction results for Neus-Facto (top), MonoSDF (middle), and UniSurf (bottom) with `pipeline.datamanager.train-num-rays-per-batch` set to 2048

10.2. Individual Model Evaluation for Each Parameter Adjustment

The evaluation of each model's results based on parameter adjustments will be carried out using multiple criteria. First, the training completion time will be considered, as it is a key factor in the efficiency of each model. Next, the final PSNR value will be assessed. Additionally, the accuracy of optical depth and color image reproduction will be examined, along with the performance in the final 3D scene reconstruction (mesh reconstruction). These criteria will allow for a comprehensive assessment of each model's performance and help identify the parameters that most influence result quality.

Regarding the Neus-Facto model, which was trained for 20,000 iterations, the average completion time was around 15 minutes, as recorded in the table. However, exceptions were observed when specific parameter changes affected the training process. These changes primarily included the evaluation frequency and the number of hidden layers in the neural network, which resulted in longer training times.

Table 19: Training Time and PSNR for Neus-Facto Across Different Parameter Configurations

Test ID	Parameter	Duration	PNSR
2024-09-10_130404	Suggested values	12m 42s	16,23
2024-09-10_132019	<code>pipeline.model.sdf-field.use-grid-feature</code> False	10m 22s	18,30
2024-09-10_133129	<code>pipeline.model.sdf-field.hidden-dim</code> 64	9m 47s	18,37
2024-09-10_134144	<code>pipeline.model.sdf-field.num-layers</code> 8	23m 4s	17,19
2024-09-10_141154	<code>pipeline.model.sdf-field.num-layers-color</code> 8	15m 41s	16,35
2024-09-10_142827	<code>pipeline.model.sdf-field.use-appearance-embedding</code> False	12m 22s	17,44
2024-09-10_144139	<code>pipeline.model.sdf-field.geometric-init</code> False	12m 20s	17,81
2024-09-10_145740	<code>pipeline.model.sdf-field.bias</code> 0,05	12m 23s	18,38
2024-09-10_161946	<code>pipeline.model.sdf-field.beta-init</code> 0,8	12m 11s	15,93
2024-09-10_152340	<code>trainer.steps-per-eval-image</code> 25	55m 36s	18,51
2024-09-11_091608	<code>pipeline.model.sensor-depth-l1-loss-mult</code> 0,8	12m 13s	18,33
2024-09-11_093426	<code>pipeline.model.sensor-depth-freespace-loss-mult</code> 0	12m 23s	15,44
2024-09-11_094702	<code>pipeline.model.sensor-depth-sdf-loss-mult</code> 0	12m 20s	19,19
2024-09-17_121649	<code>pipeline.datamanager.train-num-rays-per-batch</code> 2048	21m 35s	15,72

Regarding result accuracy, PSNR values ranged between 15 and 18, with some fluctuations depending on the parameterization, indicating that the geometric accuracy of the reconstruction is somewhat limited. Nevertheless, the overall performance of depth and color image reconstruction shows a satisfactory approximation to the original data, with certain parameter adjustments significantly improving the results.

Specifically, depth image reconstruction (Figure 101) was positively affected by specific configurations. First, disabling the Grid Feature seems to help the model simplify the learning process and focus on the essential information of the scene, avoiding unnecessary details that could introduce noise. Additionally, increasing the beta-init value provided the model with a more stable initial variance, allowing it to explore more potential solutions for accurate depth reconstruction. Finally, increasing the depth-l1-loss value improved accuracy in depth reproduction by penalizing incorrect predictions more strongly, resulting in cleaner and smoother surfaces in the reconstructions.

The process of reproducing color images (Figure 102) was also significantly influenced by specific parameter adjustments. Increasing the depth-l1-loss helped the model maintain stability and improve the clarity of the color reconstructions. On the other hand, setting the depth-sdf-loss to zero allowed the model to eliminate unnecessary geometric constraints, which in some cases may have negatively impacted the accurate color representation. Additionally, disabling geometric initialization gave the network the freedom to discover more complex geometric forms, which improved the overall quality of the color reproductions, as the model was not restricted by predefined geometric assumptions.

However, the bias parameter produced the worst results, both for the depth and color images. In the depth reconstruction, there were no discernible details of the object, and the image was significantly degraded. Similarly, for color images, the bias setting led to unclear representations. Additionally, while the Grid Feature provided a smooth and reliable depth map, it struggled with the color image reproduction, introducing a significant amount of noise in the final output, leading to a less accurate color reconstruction. This highlights the importance of balancing certain parameters for achieving the best results in both depth and color reconstructions.

Regarding the final meshes from the experiments (Figure 103), it is observed that the most optimal mesh for this algorithm is achieved by changing the depth L1 loss parameter from 0.3 to 0.8. The L1 loss parameter for depth predictions from sensor data controls the weight given to the accuracy of the depth predictions. L1 loss measures the distance between the model's predictions and the actual depth values. Adjusting the weight of this loss determines how much the model focuses on the accuracy of the depth predictions during training. A higher weight prioritizes the accuracy of depth predictions.

The final result shows significantly less noise across the entire surface of the object compared to other configurations, along with greater geometric accuracy in complex surfaces. Additionally, it demonstrates better prediction of depth changes, improving the representation of details and smaller objects.

Eval Images/sensor_depth

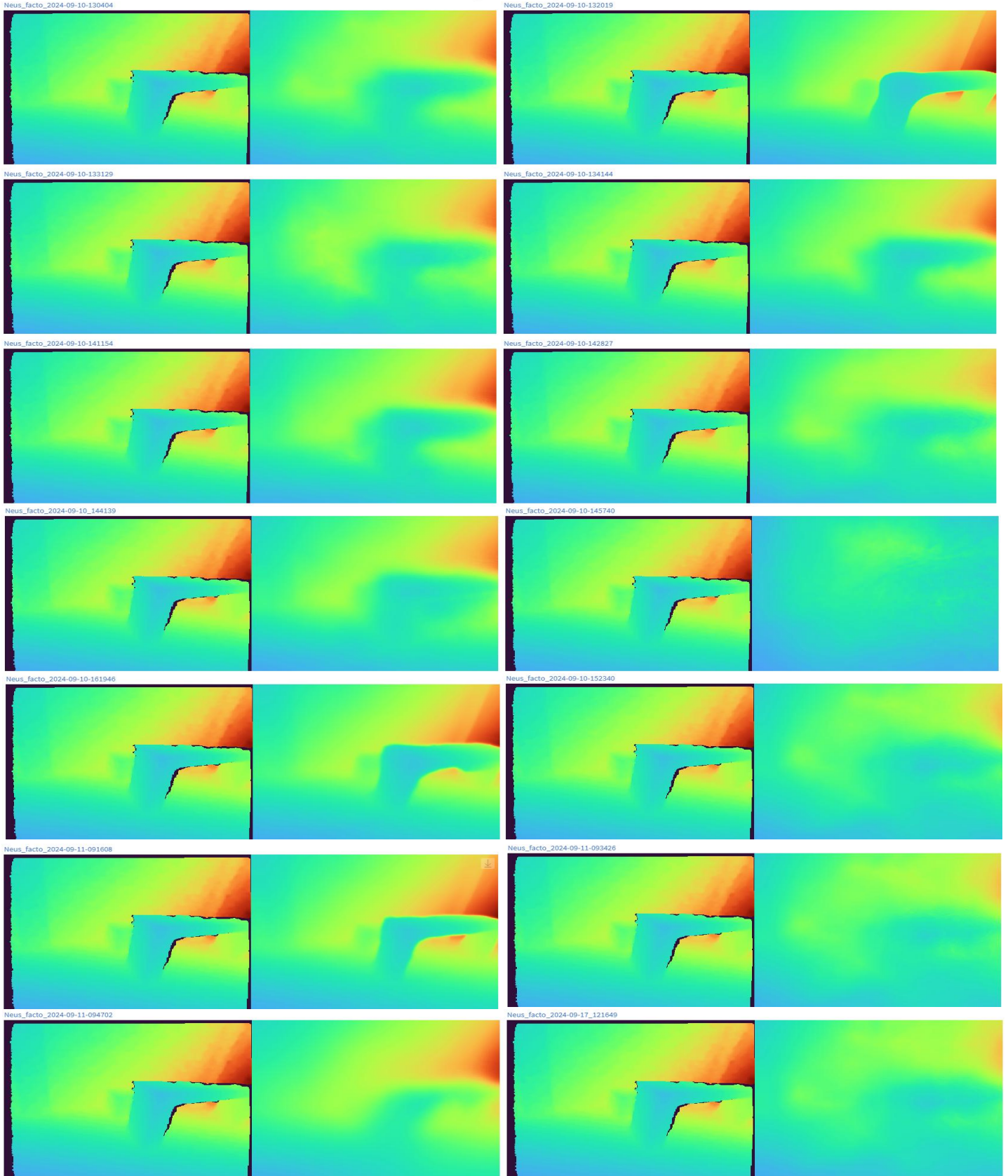


Figure 101: Neus-facto depth reconstruction of each test

Eval Images/img

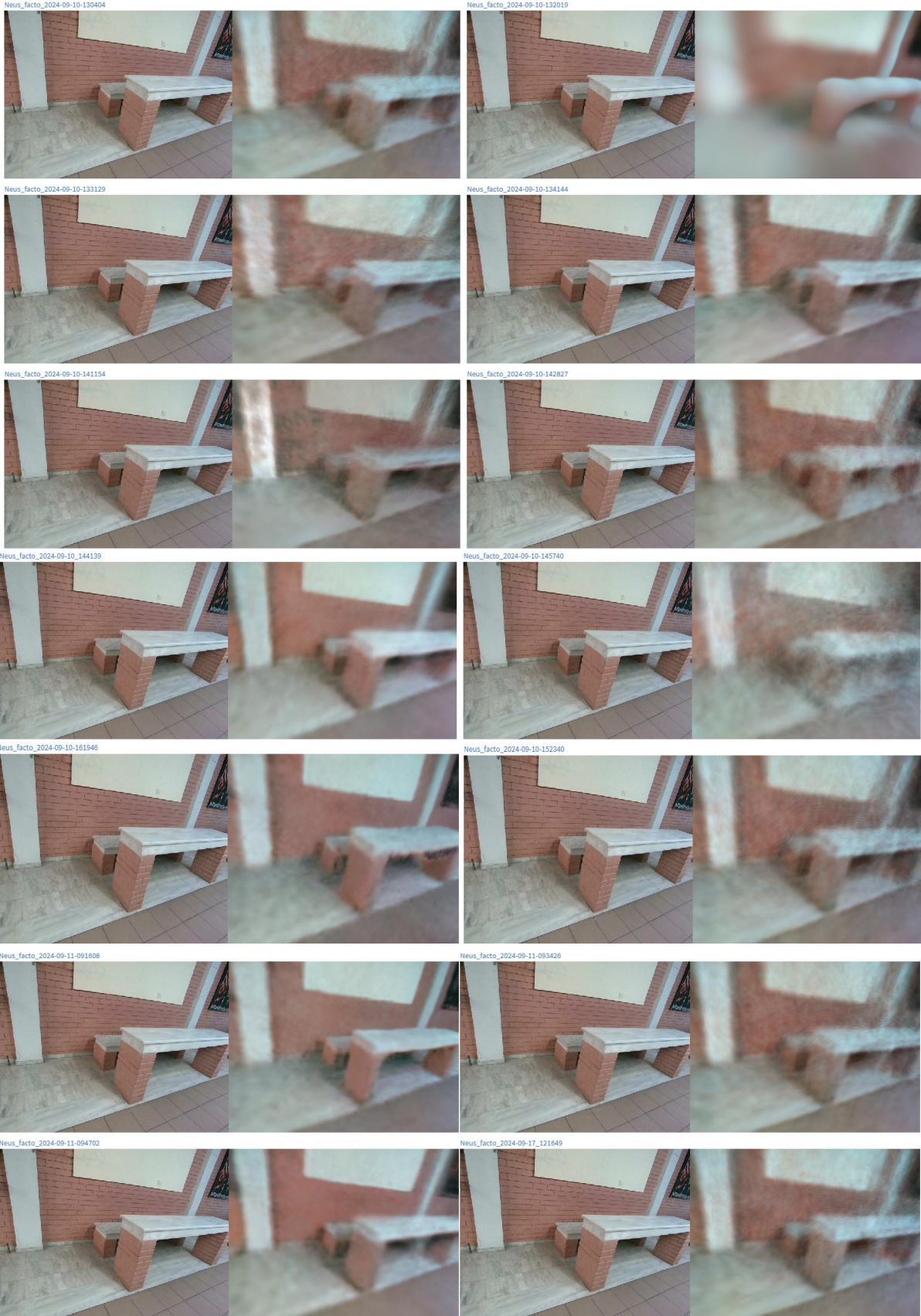


Figure 102: Neus-facto RGB reconstruction of each test



Figure 103: Neus-facto 3D reconstruction of each test

Regarding the MonoSDF model, which was trained for 50,000 iterations, the average completion time ranged from 41 minutes to over 6 hours, as recorded in the table. The completion time was influenced by specific parameter adjustments, such as the depth-l1-loss value and the bias parameter. These changes had a noticeable effect on both the training duration and the overall accuracy of the results.

Table 20: Training Time and PSNR for MonoSDF Across Different Parameter Configurations

Test ID	Parameter	Duration	PNSR
2024-09-12_134035	Suggested values	1h 17m 5s	18,92
2024-09-13_100910	pipeline.model.sdf-field.use-grid-feature False	1h 17m 27s	17,74
2024-09-13_092722	pipeline.model.sdf-field.hidden-dim 64	41m 5s	18,55
2024-09-23_151413	pipeline.model.sdf-field.num-layers 8	2h 25m 17s	18,59
2024-09-13_120721	pipeline.model.sdf-field.num-layers-color 8	1h 30m 44s	18,00
2024-09-13_134315	pipeline.model.sdf-field.use-appearance-embedding False	1h 7m 38s	21,74
2024-09-13_145415	pipeline.model.sdf-field.geometric-init False	1h 22m 39s	18,82
2024-09-13_175831	pipeline.model.sdf-field.bias 0,05	1h 10m 6s	19,12
2024-09-13_190852	pipeline.model.sdf-field.beta-init 0,8	1h 14m 32s	19,04
2024-09-13_202605	trainer.steps-per-eval-image 25	6h 36m 30s	18,78
2024-09-14_080029	pipeline.model.sensor-depth-l1-loss-mult 0,8	1h 26m 15s	18,22
2024-09-14_092725	pipeline.model.sensor-depth-freespace-loss-mult 0	1h 16m 28s	18,86
2024-09-14_104601	pipeline.model.sensor-depth-sdf-loss-mult 0	1h 21m 25s	18,75
2024-09-17_090950	pipeline.datamanager.train-num-rays-per-batch 2048	2h 5m 33s	18,24

In terms of result accuracy, PSNR values fluctuated between 18 and 21, suggesting varying degrees of fidelity in the geometric reconstructions. Despite these fluctuations, the depth image reconstructions (Figure 104) achieved remarkable improvements with certain parameter adjustments. Specifically, increasing the depth-l1-loss value significantly enhanced the accuracy of the depth maps by penalizing incorrect depth predictions more strongly. This resulted in smoother surfaces and clearer object boundaries. For color image reconstructions (Figure 105), the depth-l1-loss parameter once again played a crucial role, helping the model to maintain clarity and reduce noise in the color outputs. However, when the bias parameter was reduced from 0.8 to 0.05, the overall performance in both depth and color reconstructions saw improvements, indicating that a lower bias helps the model to focus on finer details without introducing unnecessary artifacts. On the contrary, using a higher bias tended to degrade the quality of the images, leading to less accurate representations and increased noise, especially in the color outputs.

Overall, the adjustments made to the depth-l1-loss and bias parameters proved critical in improving the quality of the depth and color reconstructions. The performance of the model shows that careful tuning of these parameters allows for more accurate and noise-free reconstructions, highlighting the importance of parameter optimization for achieving the best results.

Regarding the depth l1 loss parameter set to 0.8, it was observed that this setting led to the optimal reconstruction of the scenes, as was also the case with NeusFacto (Figure 103). The depth l1 loss controls the alignment between the estimated depth produced by the model and the depth data that is either provided or predicted by the depth models. This setting helped achieve greater accuracy in the reconstructed surfaces, especially in areas of the scene where data was sparse or insufficient. This is likely because the model enhances the accuracy of its depth predictions, thus improving the overall geometry of the reconstruction. Concerning the meshes for the built benches, the adjustment of the num layers parameter had a significant impact on the accuracy of the geometry. Increasing the network layers allowed the model to learn more complex relationships and surface details, leading to a more accurate representation of the structures. The 8 layers provided the model with greater

capacity to capture geometric details of constructed objects, such as built benches, which include geometric intricacies and curves that require higher resolution and complexity in learning. The comparison of the meshes shows that increasing the layers allowed MonoSDF to maintain accuracy in geometry, while the optimization with depth l1 loss enabled accurate representation of areas with insufficient data, which would have been difficult with other parameter settings. This indicates that MonoSDF is particularly effective when used with the appropriate settings for reconstructing detailed and geometrically complex scenes.

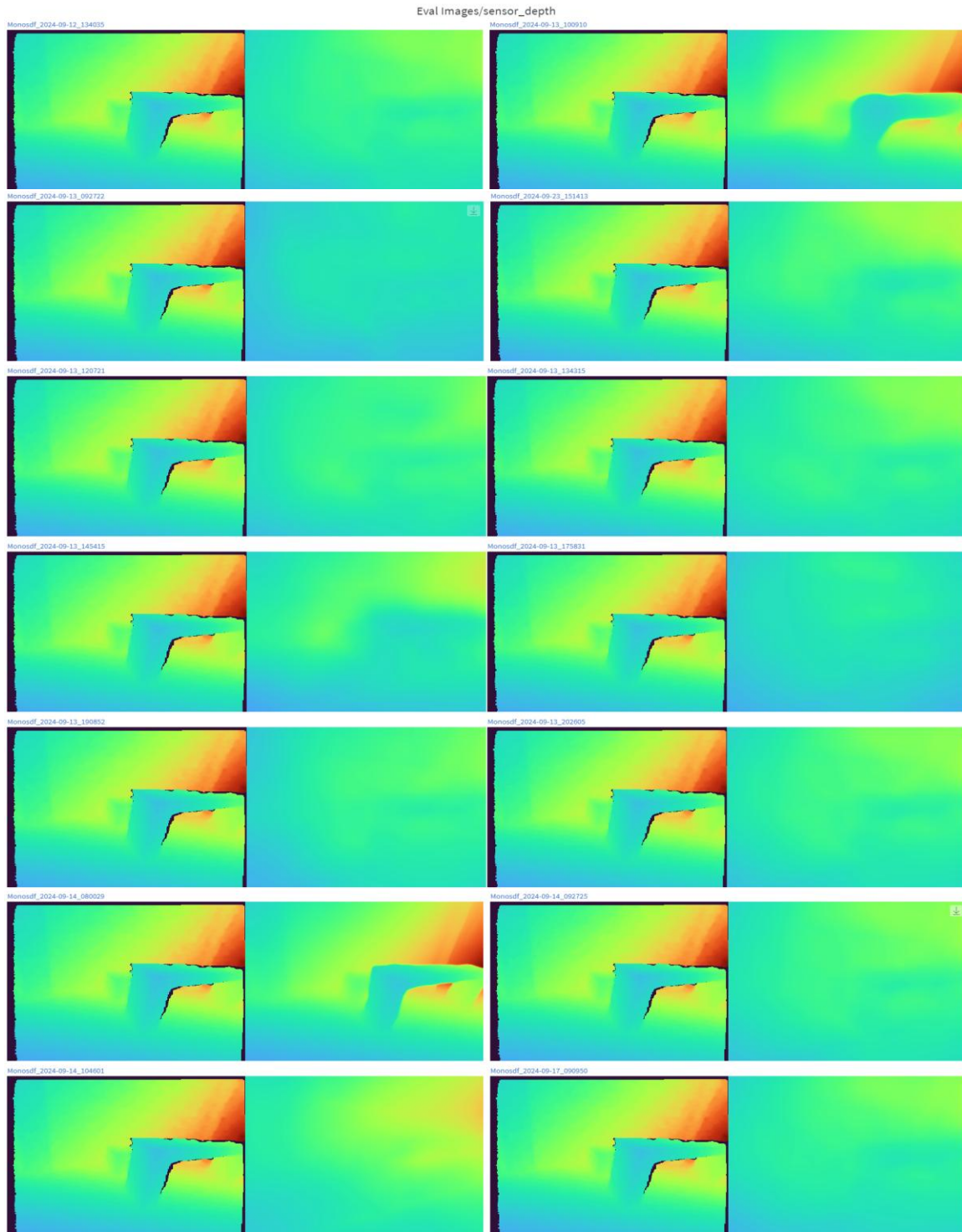


Figure 104: MonoSDF depth reconstruction of each test

Eval Images/img

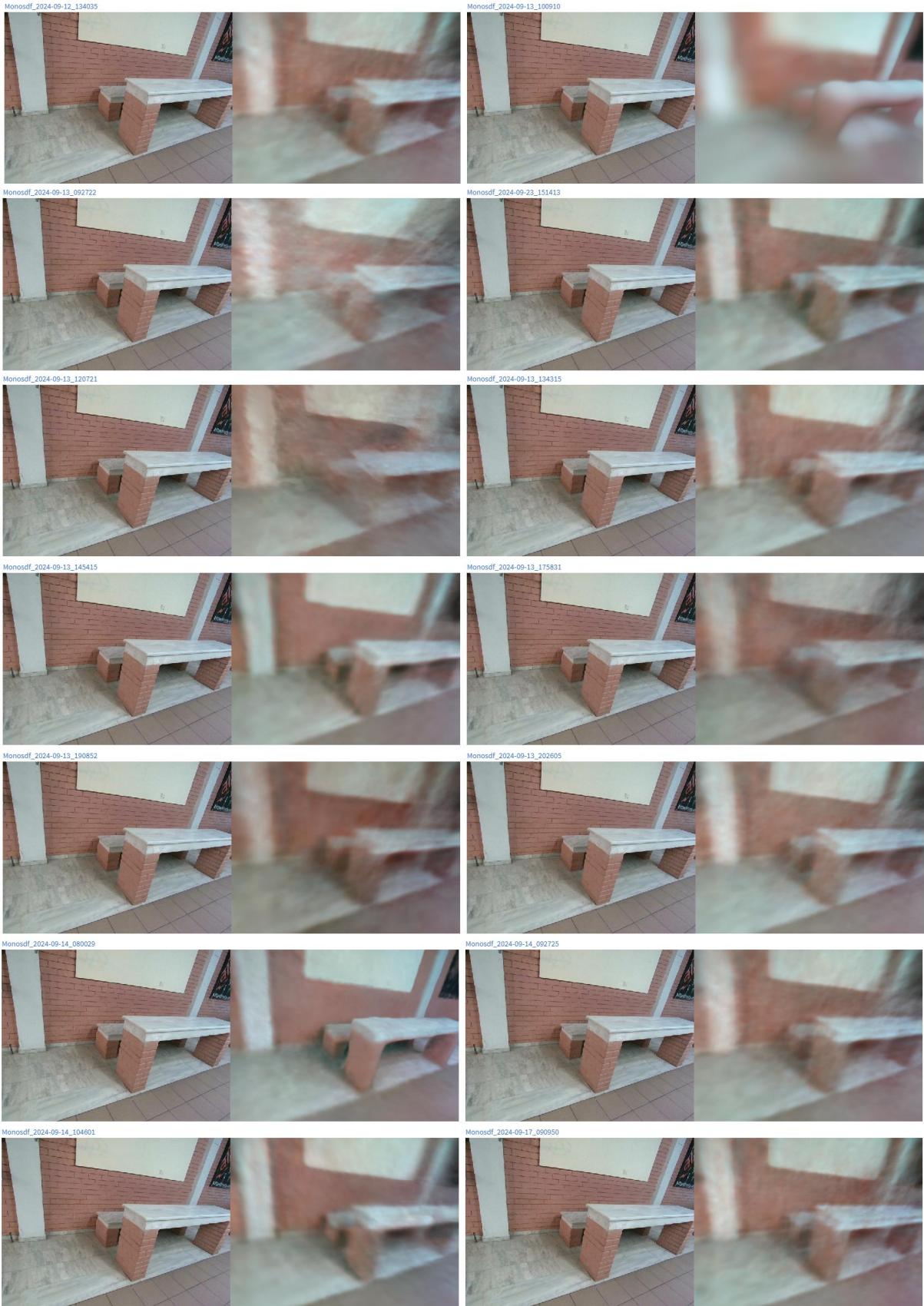


Figure 105: MonoSDF RGB reconstruction of each test

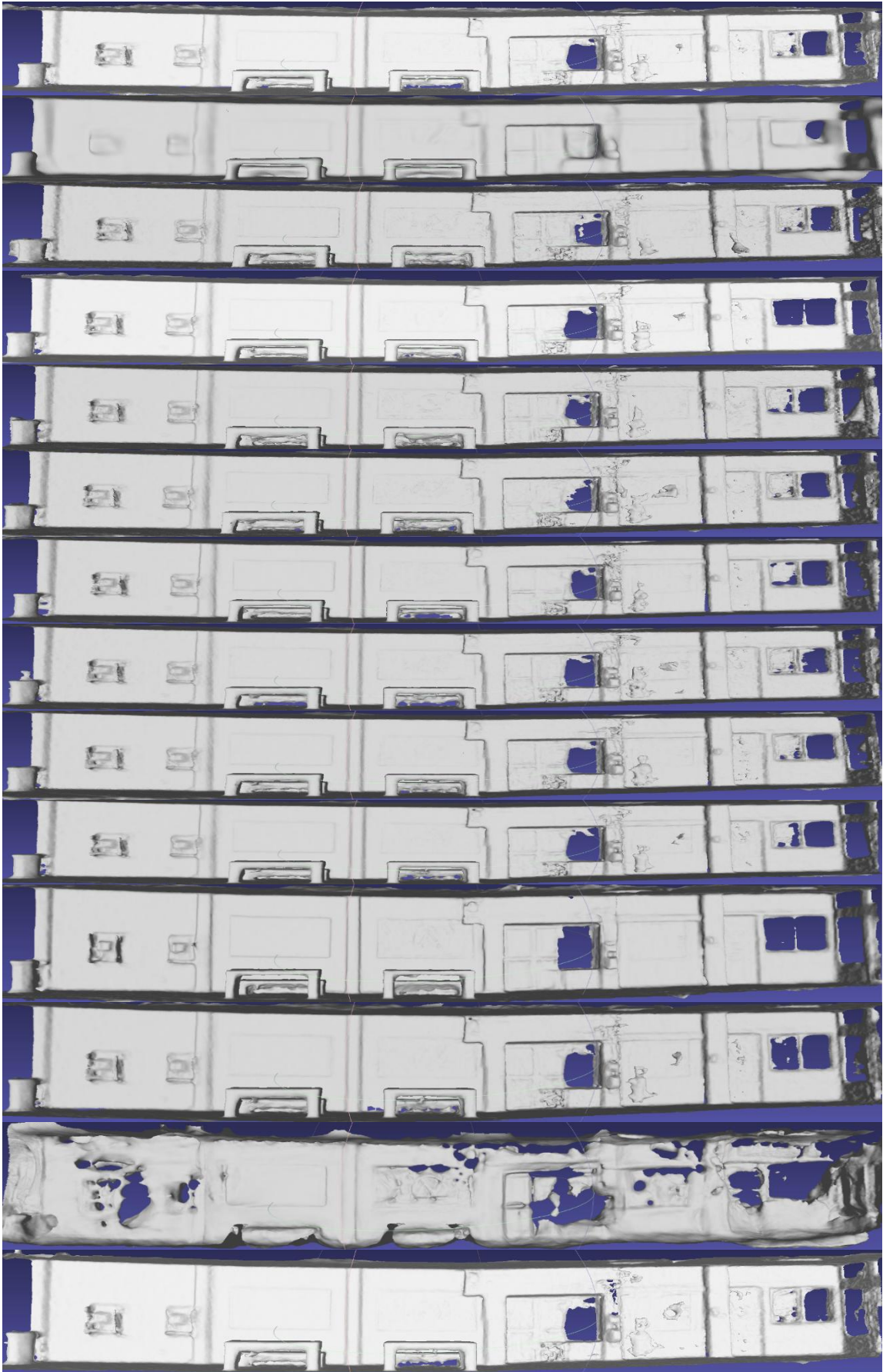


Figure 106: MonoSDF 3D reconstruction of each test

The UniSurf model trained across different iterations with different sets of parameters (Table 21) and the results present significant problems, especially within depth map reconstruction. According to times recorded in the table, the completion times varied between roughly 1 hour to over 6 hours depending on the parameters in question. First, one observation was related to how inconsistent the depth image results were, where most reconstructions had a monochromatic look and provided little to no valuable information. This issue became more evident during the manipulation of some of the important parameters around depth, such as depth-sdf-loss.

Table 21: Training Time and PSNR for UniSurf Across Different Parameter Configurations

Test ID	Parameter	Duration	PSNR
2024-09-19_111000	Suggested values	1h 27m 16s	13,64
2024-09-14_143523	pipeline.model.sdf-field.use-grid-feature False	54m 8s	15,71
2024-09-14_154623	pipeline.model.sdf-field.hidden-dim 64	44m 1s	13,45
-	pipeline.model.sdf-field.num-layers 8	-	-
2024-09-14_193435	pipeline.model.sdf-field.num-layers-color 8	1h 25m 57s	14,71
2024-09-14_210310	pipeline.model.sdf-field.use-appearance-embedding False	1h 4m 51s	14,29
2024-09-14_221104	pipeline.model.sdf-field.geometric-init False	1h 4m 58s	14,29
2024-09-15_095630	pipeline.model.sdf-field.bias 0,05	1h 5m 11s	12,98
2024-09-15_192542	pipeline.model.sdf-field.beta-init 0,8	1h 4m 57s	13,90
2024-09-15_203106	trainer.steps-per-eval-image 25	6h 6m 12s	14,04
2024-09-16_092530	pipeline.model.sensor-depth-l1-loss-mult 0,8	1h 5m 14s	17,00
2024-09-16_104519	pipeline.model.sensor-depth-freespace-loss-mult 0	1h 5m 26s	13,37
2024-09-16_121657	pipeline.model.sensor-depth-sdf-loss-mult 0	1h 5m 33s	19,00
2024-09-16_152023	pipeline.datamanager.train-num-rays-per-batch 2048	1h 58m 16s	13,53

PSNR values ranged between 13 and 19.00, but these variations did not lead to any significant improvement in geometric accuracy. Most reconstructions, particularly the depth maps, lacked structural information and failed to approximate the original data from the camera. The only notable enhancement occurred when the depth-sdf-loss parameter was set to zero, reduced from 6000. In this setup, both depth and RGB images were slightly better, but still not adequate for meaningful reconstructions. The depth image reconstructions (Figure 107) consistently lacked detailed structure, especially in the monochromatic outputs. Removing the depth-sdf-loss constraint helped relax the limitations on depth exploration, but the reconstructions still did not achieve a satisfactory level of accuracy. Further adjustments, such as changes to the bias and grid feature parameters, worsened the results, yielding noisy or incomplete images. Similarly, the RGB image reconstructions (Figure 108) followed a similar pattern. The deactivation of depth-sdf-loss improved the clarity to a small degree, but the overall results remained blurry, and no distinct structures were identifiable. The failure to reproduce color information accurately highlights the limitations in the model's current configuration.

The UniSurf model, even with various configurations, largely failed to produce reconstructions (Figure 109) that closely resembled the original camera data. The most notable improvement occurred when depth-sdf-loss was set to zero, suggesting that excessive geometric constraints may have prevented the model from focusing on critical data. However, despite this adjustment, the overall quality remained unsatisfactory, indicating the need for further parameter tuning to achieve better depth and color reconstruction. Also, as mentioned earlier, struggles to achieve highly accurate reconstruction in certain scenarios. However, one of the most successful modifications that improved the accuracy was changing the hidden dimension parameter from 256 to 64. This adjustment helped the model focus better during training, leading to more precise geometric reconstructions, while also reducing the training complexity, making the process more efficient. By using both volume and surface rendering techniques, UNISURF achieves continuous scene representation without needing object

masks. This adjustment made the model lighter and its geometry representation more accurate and stable.

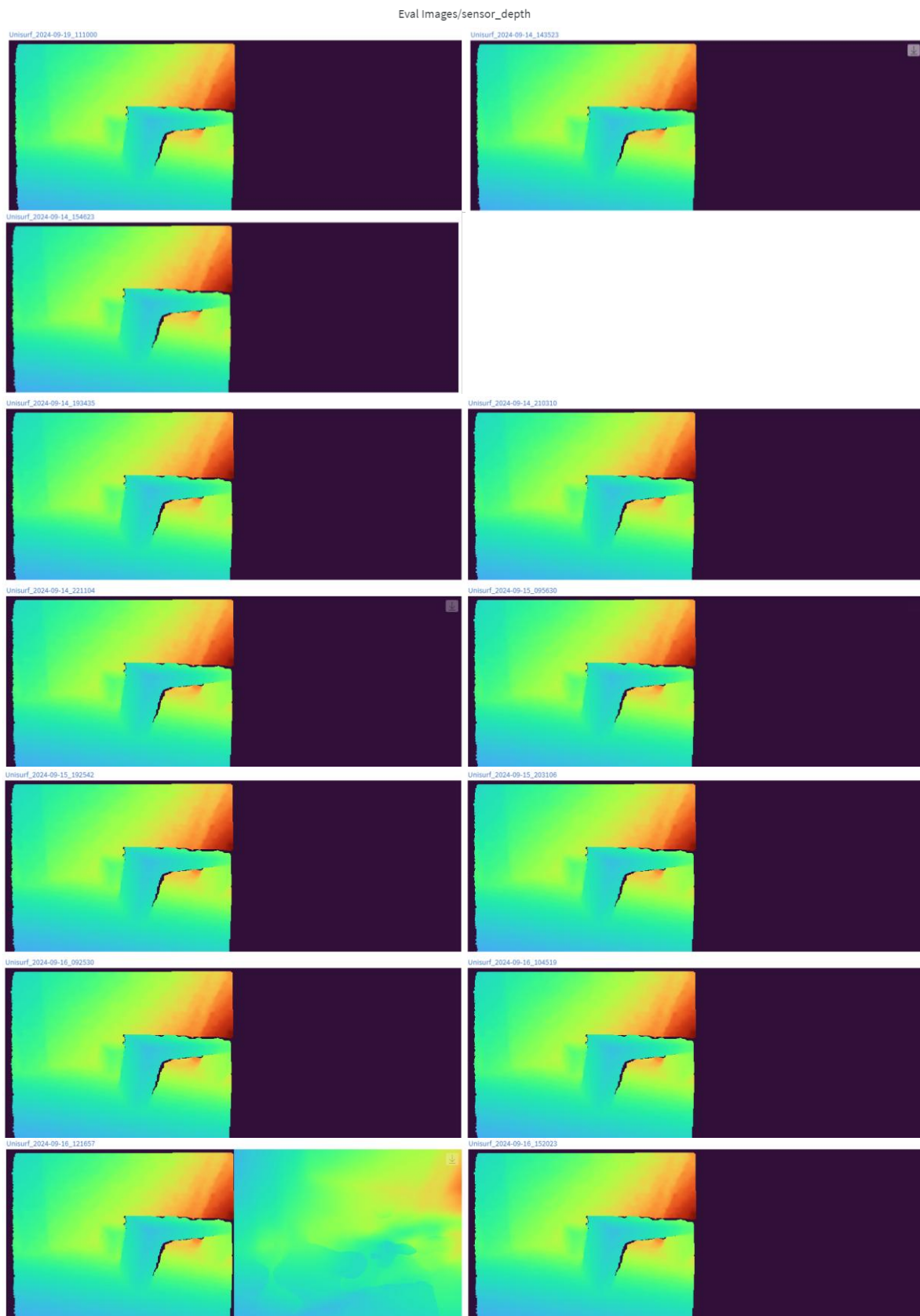


Figure 107: UniSurf Depth reconstruction of each test

Eval Images/img



Figure 108: UniSurf RGB reconstruction of each test

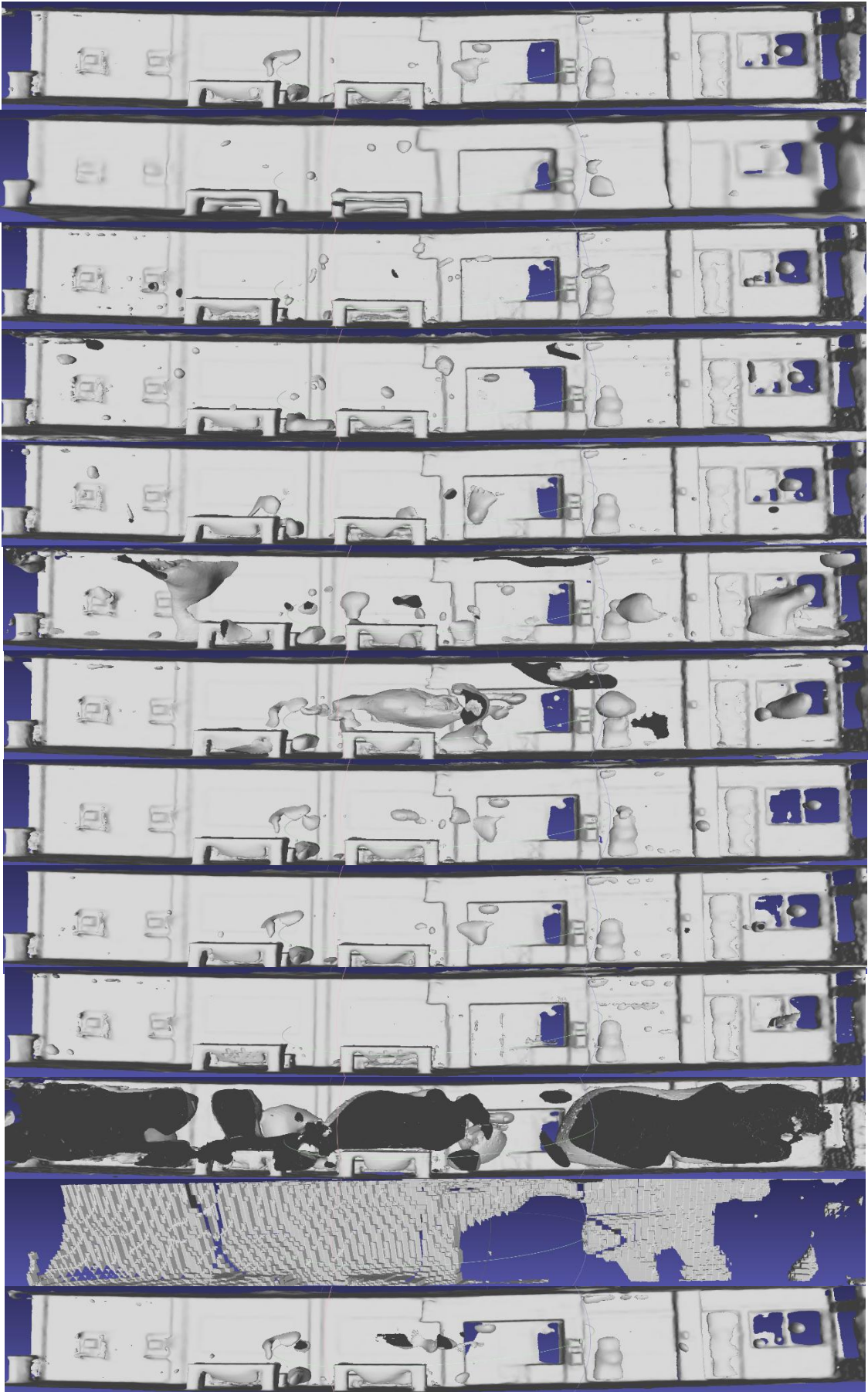


Figure 109: UniSurf 3D reconstruction of each test

10.3. Comparison of Ground Truth with the Best Reconstructed Mesh of each Algorithm

This section presents a comparison between the point clouds of a 3D scene generated employing NeuSFacto, MonoSDF and UniSurf with the ground truth surface captured by a terrestrial laser scanner (Figure 110). The comparison focuses on alignment, absolute distance metrics and performance indicators such as accuracy, recall and F-score for narrow distance thresholds (0.25 cm, 1 cm, 2 cm, 3 cm). Precision measures the quality of the reconstruction, indicating how accurately the reconstructed points align with the ground truth which serves as the reference cloud. Recall evaluates how comprehensively the reconstructed points represent the ground truth, treating the reconstructed points as the reference cloud. These metrics are combined into the F-score, which is calculated as their harmonic mean. The comparison was conducted using Cloud Compare, following manual alignment and ICP registration to address differences in scale and coordinate systems.

It is important to note that the ground truth point cloud includes areas with missing or extra information, such as the presence of chairs on the left side. These discrepancies should be taken into account when comparing it with the point clouds generated by the three models, as they are reliable comparable parts of the ground truth.

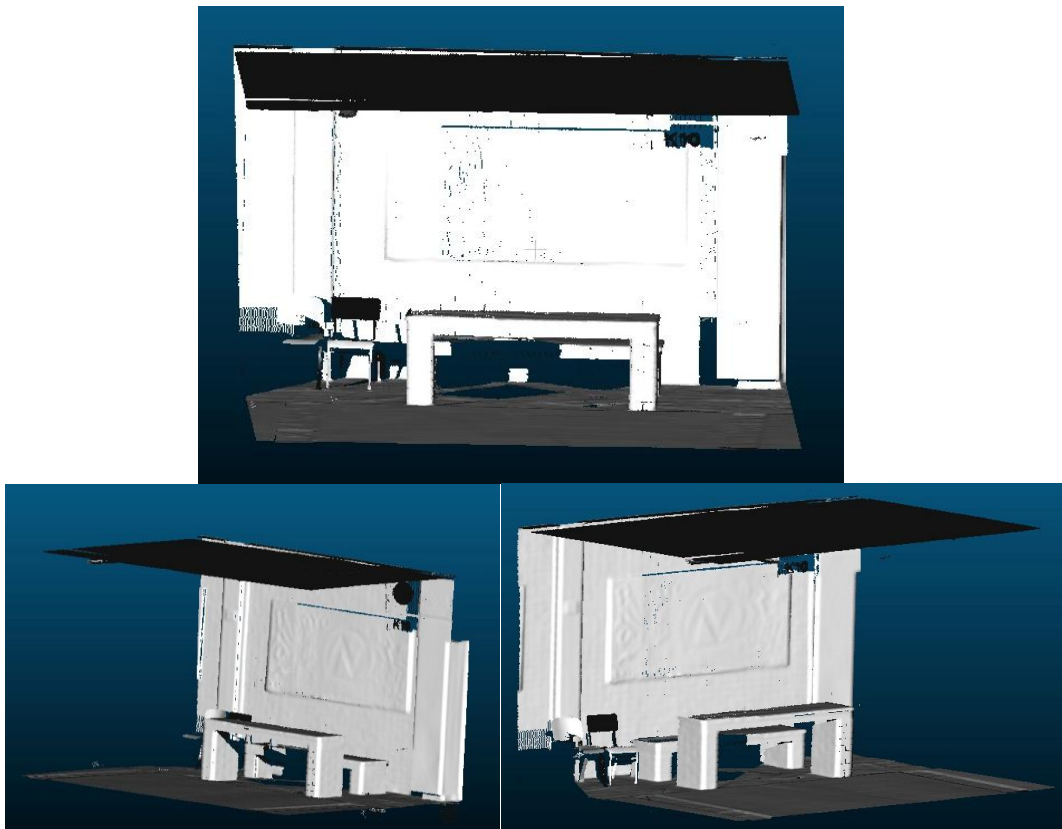


Figure 110: Ground truth (from Terrestrial Laser Scanner) as reference

The maximum comparison distance was set to 10 cm, based on the sensor's specified (2% for 4 m) and the capture distance, with a slightly extended tolerance to accommodate variations.

10.3.1. Neu-facto

The discrepancies between the NeuSFacto point cloud (Figure 111) and the ground truth are defined by a color scale.

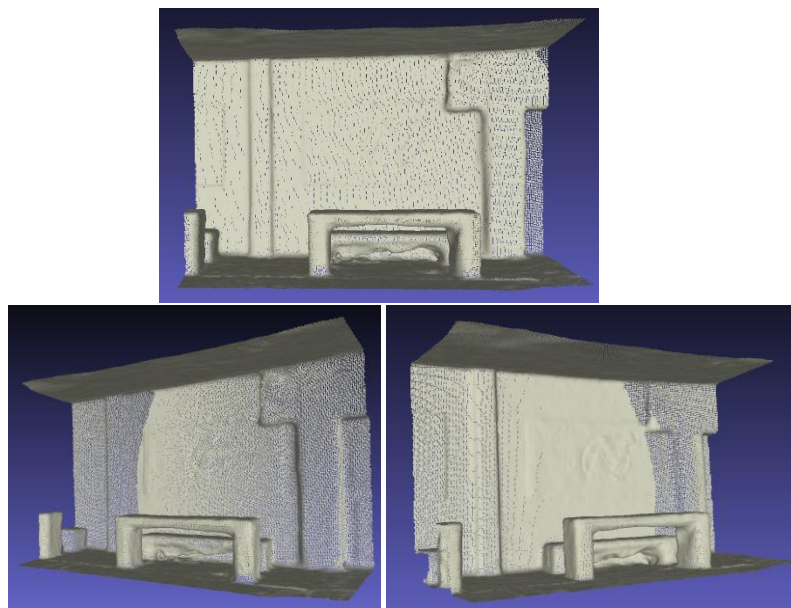


Figure 111: NeuSFacto point cloud

The blue and green regions are closely aligned with the ground truth, defining deviation distances of 0 - 5 cm, while red highlights regions with larger deviations of 10 cm (Figure 112). NeuSFacto shows very good alignment over most of its surface. Strong deviations are observed in areas where there is no information in the ground truth or there are objects that do not exist in the depth camera scan and in areas where the model's reflection did not perform with high accuracy (edges) or there was a limitation in the information data (bottom of the bench).

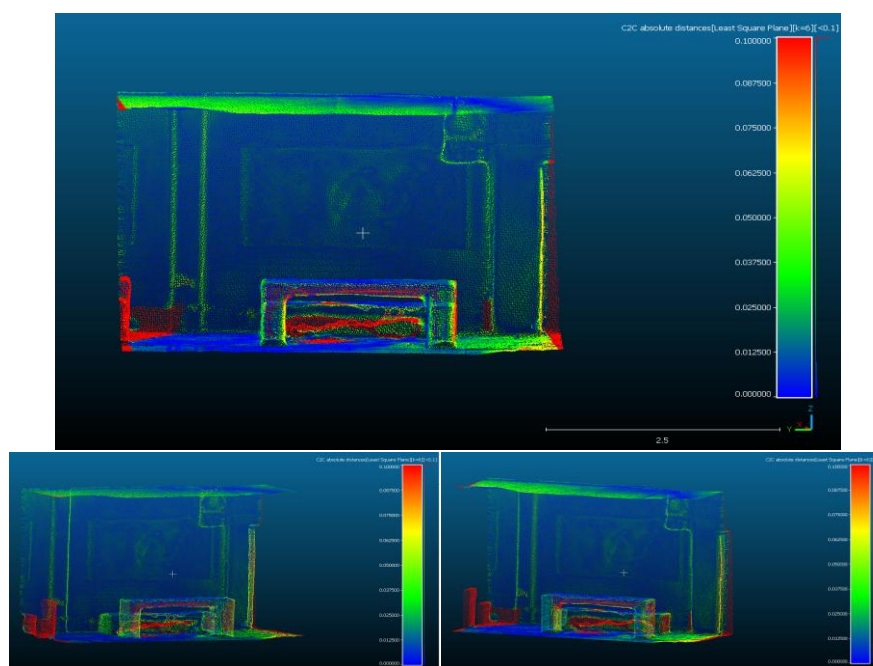


Figure 112: Absolute distances between NeuSFacto's pcd and GT's pcd

At the tighter tolerance of 0.25 cm, NeuSFacto’s F-score is relatively low, 9.505, highlighting its difficulty in capturing fine details such as edges and small features. However, as the distance threshold increases to 3 cm, the F-score improves to 58.387 (Figure 110), indicating that NeuSFacto effectively captures wider structural features, even if detailed accuracy remains difficult.

	0.25cm	1cm	2cm	3cm
Precision	13.498	43.521	62.121	72.822
Recall	7.3351	25.78	40.532	48.729
F-Score	9.505	32.38	49.056	58.387

Figure 113: Precision, Recall and F-score values for different distance thresholds

The histogram of point distances (Figure 114) illustrates the percentage of NeuSFacto's reconstruction points falling within specific distance ranges from the ground truth. A significant fraction of the points lies within 0.25 to 1 cm, indicating a fairly close alignment with the reference model for most of the scene. As the distance from ground truth increases, the percentage of points decreases, indicating that the deviations of NeuSFacto points from ground truth are less than the high precision alignments.

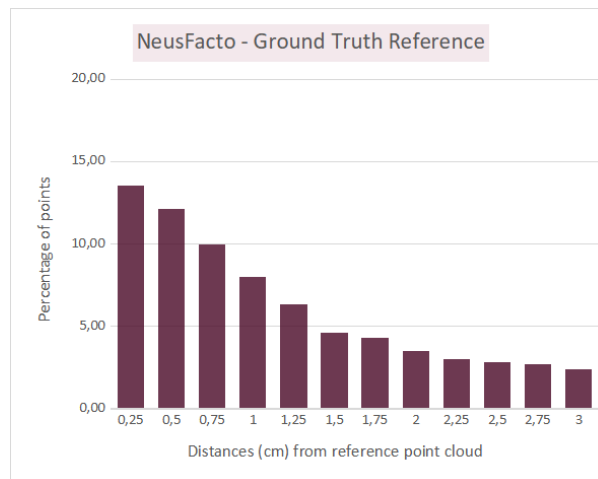


Figure 114: Percentage of NeuSFacto points distances from ground truth

NeuSFacto shows high performance in the structure of the stage. It is significantly aligned with the ground truth, as shown by the improvement in precision, recall and F-score metrics at higher tolerances. However, it exhibits challenges in complex regions where the deviations are increased.

10.3.2. MonoSDF

The distance maps show correspondingly good alignment across large surfaces, effectively capturing the general shape of the scene from MonoSDF point cloud (Figure 115).

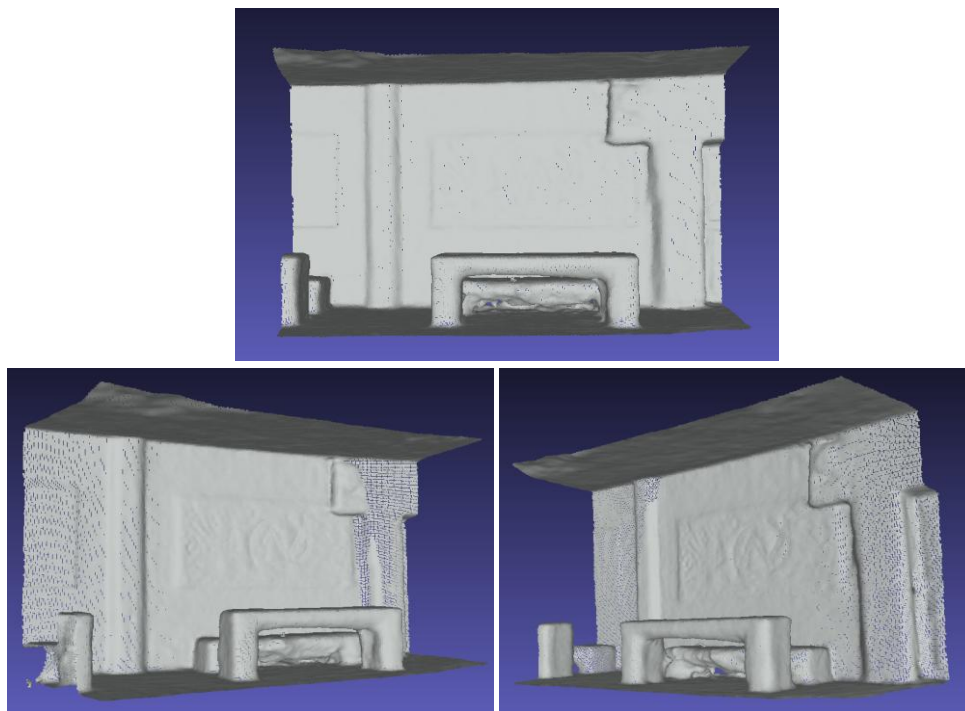


Figure 115: MonoSDF point cloud

However, like NeuSFacto, deviations increase where there is no information in the ground truth or there are objects that do not exist in the depth camera scan (Figure 116).

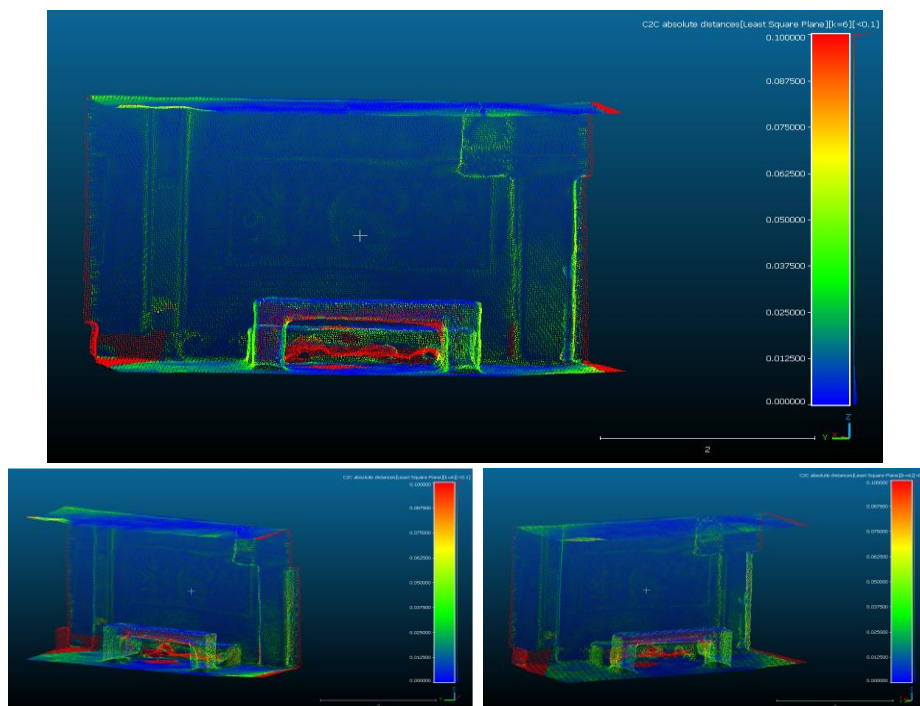


Figure 116: Absolute distances between MonoSDF's pcd and GT's pcd

The comparison at different distance thresholds (0.25 cm, 1 cm, 2 cm and 3 cm) demonstrates the performance of MonoSDF in terms of broad structural alignment. Similarly to NeuSFacto, at a narrow tolerance limit of 0.25 cm, the accuracy, recall and F-score values of MonoSDF are relatively low, with an F-score of 11.543 (Figure 117). However, as the distance threshold increases to 3 cm, the F-score improves significantly to 63.16, reflecting the ability of MonoSDF to effectively capture wider scene features.

	0.25cm	1cm	2cm	3cm
Precision	15.752	50.287	69.484	78.207
Recall	9.1085	31.487	46.209	52.97
F-Score	11.543	38.726	55.505	63.16

Figure 117: Precision, Recall and F-score values for different distance thresholds

According to the histogram (Figure 118) of the distances between the cloud and ground truth, there is a high concentration of points in the range 0.25 to 1 cm with some percentages exceeding 15%, suggesting better alignment with the reference model for most of the scene, compared to NeuSFacto. While as the distance increases, the percentage of points decreases.

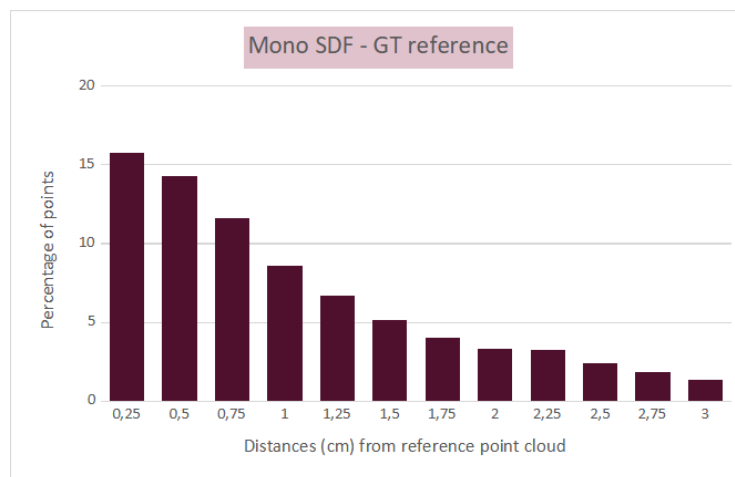


Figure 118: Percentage of MonoSDF points distances from ground truth

10.3.3. UniSurf

The distance visualization maps capture well the general structure of the point cloud and in the case of the UniSurf model's point cloud (Figure 119) with intricate details and edges showing strong divergences, especially in areas of high complexity and limited information data.

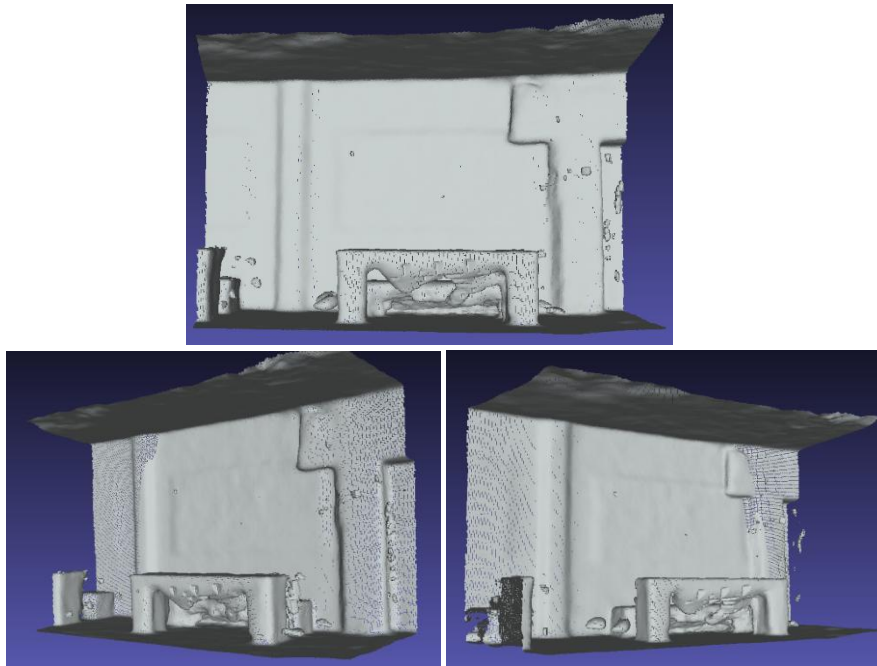


Figure 119: UniSurf point cloud

There is also a significant amount of noise at various points in the scene (Figure 120).

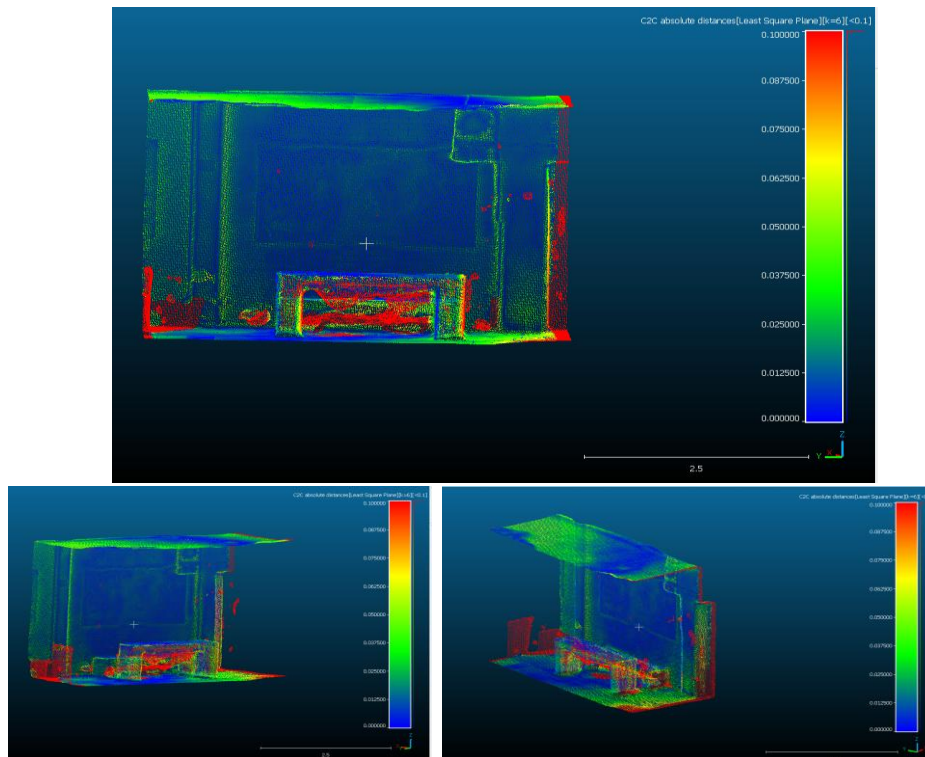


Figure 120: Absolute distances between UniSurf's pcd and GT's pcd

Evaluation at various distance thresholds (0.25 cm, 1 cm, 2 cm and 3 cm) provides further information on the accuracy of the UniSurf reconstruction, whereby it indicates lower accuracy (Figure 121) than the previous two models. As for the distance threshold at 3 cm, the F-score reached 55.308. This value highlights UniSurf’s ability to effectively capture wider structural elements, although finer details remain difficult to achieve with high fidelity.

	0.25cm	1cm	2cm	3cm
Precision	9.3865	33.424	52.022	64.907
Recall	6.0001	23.002	36.788	48.182
F-Score	7.3206	27.25	43.098	55.308

Figure 121: Precision, Recall and F-score values for different distance thresholds

The distance histogram (Figure 122) shows that a relatively significant percentage of UniSurf reconstruction points fall within the 0.25 to 1 cm range, however, as the distance threshold increases and fewer points fall within each subsequent range, no significant pointing is observed. This distribution pattern demonstrates a relatively good alignment of the UniSurf-generated scene with the ground truth, with some deviations.

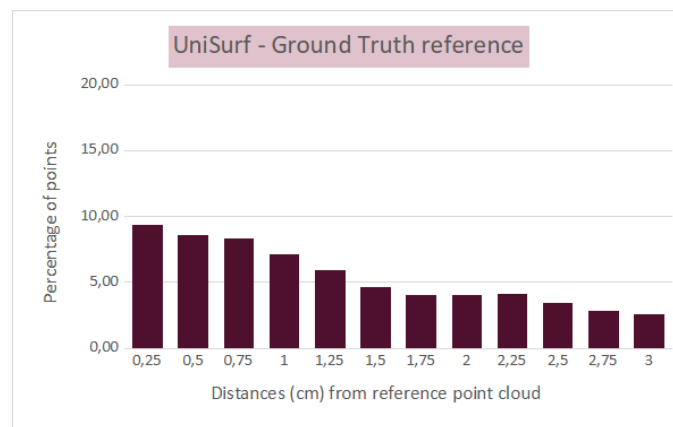


Figure 122: Percentage of UniSurf points distances from ground truth

Finally, a summary Table 22 is presented with the evaluations of all methods, with the best performances for each distance threshold highlighted in bold. Based on this table, it is evident that MonoSDF achieves the best results for each metric.

Table 22: Evaluations of all methods for each distance threshold

		0,25 cm	1 cm	2 cm	3 cm
Neus-Facto	Precision	13,498	43,521	62,121	72,822
	Recall	7,3351	25,78	40,532	48,729
	F-score	9,505	32,38	49,056	58,387
MonoSDF	Precision	15,752	50,287	69,484	78,207
	Recall	9,1085	31,487	46,209	52,97
	F-score	11,543	38,726	55,505	63,16
UniSurf	Precision	9,3865	33,424	52,022	64,907
	Recall	6,0001	23,002	36,788	48,182
	F-score	7,3206	27,25	43,098	55,308

11. Conclusions

Important conclusions about the reliability and efficiency of each model are made based on training results and performance as well as the factors affecting the final quality of reconstructed scenes. Each model performs exhibited distinct characteristics. Neus-Facto had the shortest training time, at 12 minutes and 42 seconds. However, even with its decent reconstruction accuracy, it exhibited some inconsistencies for distinct parts of the final mesh and presence of noise. The PSNR for Neus-Facto was 16.23, indicating moderate image quality for the novel views (lower than MonoSDF). Meanwhile, MonoSDF surpassed other methods in both PSNR (18.92) and structural performance. In contrast, MonoSDF required a much longer training time, exceeding one hour. On the other hand, UniSurf, despite having lower noise levels, presented a much lower quality for the reconstruction (PSNR = 13.64) and was by far the slowest model, needing 1 hour and 27 minutes of training which means worse representations using more time.

It should be noted that using the same data for both training and validation may not yield reliable results. For better outcomes, the validation dataset should differ from the training dataset as much as possible. By separating the data into two distinct sets, the evaluation of models would be more accurate, preventing overfitting. Model performance was heavily influenced by the adjustment of parameters. For instance, decreasing the “*hidden dim*” parameter from 256 to 64 improved all models, with UniSurf benefiting the most without a noticeable loss in reconstruction quality. MonoSDF exhibited faster training times with only minor reductions in quality. For some models, increasing layers up to 8 improved the fit. While MonoSDF maintained its accuracy, its training time increased significantly. Neus-Facto adapted well to the changes, while UniSurf reported memory problems (CUDA out of memory), showing limitations in handling more complex architectures.

A comparison of RGB and L1 loss functions shows that MonoSDF had the lowest values, indicating the greatest accuracy in pairing sensor data with real image data. Neus-Facto had slightly higher loss values, but still close to the original data. UniSurf had the highest L1 loss values, indicating a greater deviation from sensor data and suggesting it was the least reliable in terms of reconstruction accuracy.

Each model demonstrated unique strengths. Neus-Facto’s fast training time makes it ideal for applications where quick reconstructions are required, and absolute accuracy is not the main concern. MonoSDF, with its remarkable geometric and photorealistic reconstruction, is the best choice for cases requiring high detail and fidelity. Although less efficient in terms of time and accuracy, UniSurf proves useful for reconstructing scenes with incomplete or noisy data, making it ideal for environments where full data coverage is not possible.

Overall, the current evaluation shows that the model that provides the highest accuracy in 3D reconstruction is MonoSDF, albeit with a prolonged training time. Neus-Facto, on the other hand, outperforms other methods for efficient model learning with minor accuracy compared to MonoSDF, but with excellent integration time. Although UniSurf is less accurate than the other algorithms, it is useful in cases of employing geometric reconstruction from noisy data.

Finally, based on the results from distance comparisons of Point clouds and ground truth, MonoSDF demonstrates, once again, the best overall performance in scene reconstruction. It achieves higher accuracy, recall and F-scores at all distance thresholds compared to the next best and recommended model, NeuSFacto. For instance, at the 3 cm threshold, MonoSDF yields an F-score of 63.16, while NeuSFacto reaches 58.387. This higher F-score at close and moderate tolerances suggests that MonoSDF has a better balance between detail accuracy and structural alignment. Furthermore,

the histogram of distances indicates that MonoSDF has a higher concentration of points within the 0.25 to 1 cm range, highlighting better alignment with the ground truth at closer distances.

12. Future Works

The research conducted in this thesis focused on studying and evaluating the capabilities of selected models including Neus-Facto, UniSurf and MonoSDF, within SDFStudio for reconstructing 3D scenes from depth camera data. However, several opportunities for further investigation and improvement remain. One potential direction involves the optimization of existing neural network architectures to exploit depth maps from scratch, without the need to optimize the extracted data. This could reduce processing times and improve accuracy.

Furthermore, the implementation and testing of various algorithms available in both NeRFstudio and SDFStudio could, through experimentation of additional options and parameter adjustments, further improve the quality of the reconstruction.

Another practical limitation is the time required to train and reconstruct scenes. Developing methods for real-time 3D reconstruction, particularly using depth camera data, is a significant challenge.

Finally, exploring modern methods techniques such as Gaussian Splatting could provide faster reconstruction times with high accuracy. Studies have shown that Gaussian Splatting can achieve realistic reconstruction of complex scenes efficiently, providing an opportunity for comparisons and potential integration with existing models in SDFStudio.

Bibliography

- Alake, R. (2023, November 24). *Loss Functions in Machine Learning Explained*. Retrieved from datacamp: <https://www.datacamp.com/tutorial/loss-function-in-machine-learning>
- Alexa, M. (2003, 05). Differential coordinates for local mesh morphing. *The visual Computer*, 19, 105-114. doi:10.1007/s00371-002-0180-0
- Anders, G.-J., John, S., Tri, K., Sergey, D., Dave, T., Ofir, M., . . . Evgeni, R. (n.d.). *Intel® RealSense™ Self-Calibration for D400 Series Depth Cameras*.
- Backpropagation in Neural Network*. (2024, Jul 09). Retrieved from geeksforgeeks: <https://www.geeksforgeeks.org/backpropagation-in-neural-network/>
- Bálint, C., Valasek, G., & Gergo, L. (2023, 03). Operations on Signed Distance Function Estimates. *Computer-Aided Design and Applications*, 1154-1174. doi:10.14733/cadaps.2023.1154-1174
- Barron, J., Mildenhall, B., Verbin, D., Srinivasan, P., & Hedman, P. (2011). *Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields*. doi:10.48550/arXiv.2111.12077
- Basso, F., Menegatti, E., & Pretto, A. (2017, 01). Robust Intrinsic and Extrinsic Calibration of RGB-D Cameras. *IEEE Transactions on Robotics*, 34. doi:10.1109/TRO.2018.2853742
- Bourke, P. (1997, June). Implicit surfaces. Retrieved from <https://paulbourke.net/geometry/impliciturf/>
- Carnegie Mellon University. (2022, 05 08). 3. VolSDF. Retrieved from <https://www.andrew.cmu.edu/course/16-889/projects/rohansax/proj4/>
- Chen, S., Wu, B., Li, H., Li, Z., & Liu, Y. (2024, 05). Asteroid-NeRF: A deep-learning method for 3D surface reconstruction of asteroids. *Astronomy and Astrophysics* 687, 687, 687. doi:10.1051/0004-6361/202450053
- Choi, S., Zhou, Q.-Y., & Koltun, V. (2015). *Robust Reconstruction of Indoor Scenes*. doi:10.1109/CVPR.2015.7299195
- Chris, B. (2023, October 6). *Intro to Signed Distance Fields*. Retrieved from RustAdventure: <https://www.rustadventure.dev/introduction-to-signed-distance-fields>
- Darwish, W., Li, W., Tang, S., Li, Y., & Chen, W. (2019, 05). AN RGB-D DATA PROCESSING FRAMEWORK BASED ON ENVIRONMENT CONSTRAINTS FOR MAPPING INDOOR ENVIRONMENTS. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 263-270. doi:10.5194/isprs-annals-IV-2-W5-263-2019
- Depth cameras and RGB-D camera SLAM*. (2020, 11 06). Retrieved from kudan: <https://www.kudan.io/blog/depth-cameras-and-rgb-d-slam/>
- DEPTH SENSING TECHNOLOGIES OVERVIEW*. (n.d.). Retrieved from FRAMOS: <https://www.framos.com/en/products-solutions/3d-depth-sensing/depth-sensing-technologies#:~:text=Working%20Principles%20Stereo%20Vision&text=The%20technology%20determines%20quantifiable%20depth,at%20a%20slightly%20different%20angle.>
- DrMax. (2022, November 4). *Computer Vision: Stereo 3D Vision*. Retrieved from baeldung: <https://www.baeldung.com/cs/stereo-vision-3d>

- Dutta, S. (2024, June 5). *Understanding the Number of Hidden Layers in Neural Networks: A Comprehensive Guide*. Retrieved from Medium: https://medium.com/@sanjay_dutta/understanding-the-number-of-hidden-layers-in-neural-networks-a-comprehensive-guide-0c3bc8a5dc5d
- Epochs, Batch Size, Iterations - How are They Important to Training AI and Deep Learning Models*. (2024, August 09). Retrieved from sabrepc: <https://www.sabrepc.com/blog/Deep-Learning-and-AI/Epochs-Batch-Size-Iterations>
- Esposito, S., Xu, Q., Kania, K., Hewitt, C., Mariotti, O., Petikam, L., . . . Aodha, O. (2024). *GeoGen: Geometry-Aware Generative Modeling via Signed Distance Functions*. doi:10.48550/arXiv.2406.04254
- Garcia, J. (2020). *Intel® RealSense™ Depth Camera D455 shows reddish color in RGB image when there is bright sunlight*. Retrieved from Intel RealSense : <https://support.intelrealsense.com/hc/en-us/articles/360058987613-Intel-RealSense-Depth-Camera-D455-shows-reddish-color-in-rgb-image-when-there-is-bright-sunlight>
- GateVidyalay. (2020). *3D Transformations in Computer Graphics*-. Retrieved from https://www.gatevidyalay.com/tag/specular-reflection-in-computer-graphics/?utm_content=cmp-true
- geeksforgeeks. (2018, September 25). *Effect of Bias in Neural Network*. Retrieved from <https://www.geeksforgeeks.org/effect-of-bias-in-neural-network/>
- geeksforgeeks. (2023, July 31). *Impact of learning rate on a model*. Retrieved from <https://www.geeksforgeeks.org/impact-of-learning-rate-on-a-model/>
- Grisetti, G., Kümmerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based SLAM. *IEEE Transactions on Intelligent Transportation Systems Magazine*, 31-43.
- Gupta, M., Yin, Q., & Nayar, S. (2013, 12). Structured Light in Sunlight. *Proceedings of the IEEE International Conference on Computer Vision*, 545-552. doi:10.1109/ICCV.2013.73
- Henri, P. G. (2024). *The Levenberg-Marquardt algorithm for*. Department of Civil and Environmental Engineering.
- Henry, P., Krainin, M., Ren, X., & Fox, F. (2014). RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments. In O. Khatib, V. Kumar, & G. Sukhatme (Eds.), *Experimental Robotics: The 12th International Symposium on Experimental Robotics* (pp. 477-491). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-28572-1_33
- Ho, N. (n.d.). *Finding optimal rotation and translation between corresponding 3D points*. Retrieved from https://nghiaho.com/?page_id=671
- How do you choose the optimal baseline and resolution for a stereo vision camera?* (2024, 7 17). Retrieved from LinkedIn: <https://www.linkedin.com/advice/1/how-do-you-choose-optimal-baseline-resolution-stereo>
- Ibrahim, M., Liu, Q., Khan, R., Yang, J., Adeli, E., & Yang, Y. (2020, 09). Depth map artifacts reduction: a review. *IET Image Processing*, 14. doi:10.1049/iet-ipr.2019.1622

- lesalnieks, J. (2022, June 7). *Full-Reference Quality Metrics: VMAF, PSNR and SSIM*. Retrieved from TestDevLab: <https://www.testdevlab.com/blog/full-reference-quality-metrics-vmf-psnr-and-ssim>
- IntelRealSense. (2019, July 15). *Beginner's guide to depth (Updated)*. Retrieved from Intel real sense: <https://www.intelrealsense.com/beginners-guide-to-depth/>
- Jaesik, Park, Q.-Y., Zhou, V., & Koltun. (2017). Colored Point Cloud Registration Revisited. *Conference: 2017 IEEE International Conference on Computer Vision*, (pp. 143-152). doi:10.1109/ICCV.2017.25
- John, Sweetser, A., & Grunnet-Jepsen. (n.d.). *Optical Filters for Intel® RealSense™ Depth Cameras D400*. Retrieved from Intel RealSense: <https://dev.intelrealsense.com/docs/optical-filters-for-intel-realsense-depth-cameras-d400>
- Kate, Y. (2023, December 23). A Comprehensive Overview of Gaussian Splatting. *Towards data Science*.
- Kazhdan, M., Bolitho, M., & Hoppe., H. (2006). Poisson Surface Reconstruction. *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. 256, pp. 61-70. Switzerland: Eurographics Association. Retrieved from <http://dl.acm.org/citation.cfm?id=1281957.1281965>
- Kim, J.-K., Byung-Seo, P., Woosuk, K., Jung-Tak, P., So, I. L., & Young-Ho, S. (2022). *Robust Estimation and Optimized Transmission of 3D Feature Points for Computer Vision on Mobile Communication Network*. doi:<https://doi.org/10.3390/s22218563>
- Kočevár, T., & Gabrijelčič, H. (2017). *Modelling and Visualisation of the Optical Properties of Cloth*. doi:10.5772/67736
- Krish, N. (2019). *How to choose number of hidden layers and nodes in Neural Network*. Retrieved from <https://www.youtube.com/watch?v=Bc2dWI3vnE0>
- Kumar, P. (2022, 19 May). *What are RGBD cameras? Why RGBD cameras are preferred in some embedded vision applications?* Retrieved from e-con systems: <https://www.e-consystems.com/blog/camera/technology/what-are-rgbd-cameras-why-rgbd-cameras-are-preferred-in-some-embedded-vision-applications/#:~:text=Adding%20RGB%20data%20to%20a,measuring%20the%20depth%20to%20them.>
- Langmann, B., Hartmann, K., & Loffeld, O. (2012, 01). Depth camera technology comparison and performance evaluation. *ICPRAM 2012 - Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, 2, 438-444.
- Li, L. (n.d.). *Time-of-Flight Camera - An Introduction*. Retrieved from Mouser Electronics: <https://gr.mouser.com/applications/time-of-flight-robotics/>
- Low, K.-L. (2004). Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration.
- Ma, Y., Chuang, L., Bo, W., Zhang, S., Li, M., & Song, P. (2018, 10). Weighted Total Least Squares for the Visual Localization of a Planetary Rover. *Photogrammetric Engineering & Remote Sensing*, 605-618. doi:10.14358/PERS.84.10.605

- Matrix Transformations*. (n.d.). Retrieved from Developer Unigine: https://developer.unigine.com/en/docs/latest/code/fundamentals/matrix_transformations/index?rlang=cpp
- Mildenhall, B., Srinivasan, P., Tancik, M., Barron, J., Ramamoorthi, R., & Ng, R. (2022, 01). NeRF: representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, *65*, 99-106. doi:10.1145/3503250
- Oechsle, M., Peng, S., & Geiger, A. (2021). *UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction*.
- Park, J., Zhou, Q.-Y., & Koltun, V. (2017). Colored Point Cloud Registration Revisited., (pp. 143-152). doi:10.1109/ICCV.2017.25
- Pfingsthorn, M. (2014). *Generalized Simultaneous Localization and Mapping (SLAM) on Graphs with Multimodal Probabilities and Hyperedges*. doi:10.13140/RG.2.1.4662.3600
- RealSense. (2018). *RGB and Depth frames not synchronized #1548*. Retrieved from GitHub: <https://github.com/IntelRealSense/librealsense/issues/1548>
- RealSense. (n.d.). *D435i*. Retrieved from GitHub: <https://github.com/IntelRealSense/librealsense/blob/master/doc/d435i.md#sensor-origin-and-coordinate-system>
- RealSense, I. (2023). *Intel RealSense Product Family D400 Series*. Intel RealSense. Retrieved from <https://www.intelrealsense.com/wp-content/uploads/2020/06/Intel-RealSense-D400-Series-Datasheet-June-2020.pdf>
- Relja, A., & Andrew, Z. (2021). *NeRF in detail: Learning to sample for view synthesis*.
- Salvi, J., Pages, J., & Batlle, J. (2004, 04). Pattern codification strategies in structure light systems. *Pattern Recognition*, *37*, 827-749. doi:10.1016/j.patcog.2003.10.002
- Scharstein, D., & Szeliski, R. (2003, 07). High-accuracy stereo depth maps using structured light. *Comput. Vision Pattern Recognit.*, *1*, 1-195. doi:10.1109/CVPR.2003.1211354
- Seitz, S. M. (1999). *An Overview of Passive Vision Techniques*. Retrieved from <https://api.semanticscholar.org/CorpusID:14340500>
- Sejal, J. (2024). *Multilayer Perceptrons in Machine Learning: A Comprehensive Guide*. Retrieved from datacamp: <https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning>
- Sharma, K., Kim, S., & Singh, M. (2012, 08). An improved feature matching technique for stereo vision applications with the use of self-organizing map. *International Journal of Precision Engineering and Manufacturing*, *13*. doi:10.1007/s12541-012-0179-z
- Simsangcheol. (2023, 5 2). *Pose Graph Optimization*. Retrieved from Medium: <https://medium.com/@sim30217/pose-graph-optimization-30ce29e4d65f>
- Singh, A. (2013, Dec 5). *3 Beginner-Friendly Techniques to Extract Features from Image Data using Python*. Retrieved from Analytics Vidhya: <https://www.analyticsvidhya.com/blog/2019/08/3-techniques-extract-features-from-image-data-machine-learning-python/>
- Skann.ai. (2023, September 21). *Neural Radiance Fields*. Retrieved from Medium: <https://skannai.medium.com/neural-radiance-fields-a3d4206832e5>

- Steinberg, S., Ramamoorthi, R., Bitterli, B., d'Eon, E., Yan, L.-Q., & Pharr, M. (2023). *A Generalized Ray Formulation For Wave-Optics Rendering*.
- Stereo Vision for 3D Machine Vision Applications. (n.d.). Retrieved from clearview: <https://www.clearview-imaging.com/en/blog/stereo-vision-for-3d-machine-vision-applications>
- Sugihara, K. (1986). Three principles in stereo vision. *Advanced Robotics*, 391-400. doi:10.1163/156855386X00256
- Sun, X., Mei, X., Jiao, S., Zhou, M., & Wang, H. (2011). Stereo Matching with Reliable Disparity Propagation., (pp. 132-139). doi:10.1109/3DIMPVT.2011.24
- Swayam. (2023, Apr 17). *Exploring Different Encoding Techniques for Machine Learning Models*. Retrieved from Medium: <https://medium.com/@swayampatil7918/exploring-different-encoding-techniques-for-machine-learning-models-b762e3b546db>
- Tancik, M., Weber, E., Ng, E., Li, R., & Yi, B. (2023). *Nerfstudio*. Retrieved from Github: <https://github.com/nerfstudio-project/nerfstudio/>
- Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J. a., . . . Kanazawa, A. (2023). *Nerfstudio: A Modular Framework for Neural Radiance Field Development*. doi:10.48550/arXiv.2302.04264
- Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., . . . Kanazawa, A. (2022). *Nerfacto*. Retrieved from Nerfstudio: <https://docs.nerf.studio/nerfology/methods/nerfacto.html#pose-refinement>
- Tang, S., Zhu, Q., Chen, W., Darwish, W., Wu, B., Hu, H., & Chen, M. (2016, 09). Enhanced RGB-D Mapping Method for Detailed 3D Indoor and Outdoor Modeling. *Sensors*, 16, 1589. doi:10.3390/s16101589
- team, V. (2018, October 10). *What is a stereo vision camera?* Retrieved from e-consystems.com: <https://www.e-consystems.com/blog/camera/technology/what-is-a-stereo-vision-camera-2/>
- Tutorialspoint. (n.d.). *Hash Table Data structure*. Retrieved from https://www.tutorialspoint.com/data_structures_algorithms/hash_data_structure.htm
- Tychola, K., Tsimperidis, I., & Papakostas, G. (2022, 08). On 3D Reconstruction Using RGB-D Cameras. *Digital*, 2, 401-423. doi:10.3390/digital2030022
- Understanding the normal mapping process*. (n.d.). Retrieved from Adobe: <https://www.adobe.com/products/substance3d/discover/normal-mapping.html>
- Unity. (n.d.). *Normal map (Bump mapping)*. Retrieved from Unity documentation: <https://docs.unity3d.com/2018.3/Documentation/Manual/StandardShaderMaterialParameterNormalMap.html>
- Wang, H., Wang, J., & Agapito, L. (2023). Co-SLAM: Joint Coordinate and Sparse Parametric Encodings for Neural Real-Time SLAM. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 13293-13302). doi:10.1109/CVPR52729.2023.01277
- Wang, J., Bleja, T., & Agapito, L. (2022). GO-Surf: Neural Feature Grid Optimization for Fast, High-Fidelity RGB-D Surface Reconstruction. *2022 International Conference on 3D Vision (3DV)*, (pp. 433-442).

- Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., & Wang, W. (2021). NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *NeurIPS*.
- Weights & Biases Documentation*. (n.d.). Retrieved from Wights&BiasDocs: <https://docs.wandb.ai/>
- Wong, J. (2016, July 15). *Ray Marching and Signed Distance Functions*. Retrieved from ZeroWind: <https://jamie-wong.com/2016/07/15/ray-marching-signed-distance-functions/>
- Wu, L., Lee, J., Bhattad, A., Wang, Y.-X., & Forsyth, D. (2022). DIVER: Real-time and Accurate Neural Radiance Fields with Deterministic Integration for Volume Rendering. *Conference on Computer Vision and Pattern Recognition*, (pp. 16179-16188). doi:10.1109/CVPR52688.2022.01572
- Yang, H., Sun, Y., Sundaramoorthi, G., & Yezzi, A. (2023). *StEik: Stabilizing the Optimization of Neural Signed Distance Functions and Finer Shape Representation*. doi:10.48550/arXiv.2305.18414
- Yang, Z., Danqing, C., Jun, W., Mingyi, H., & Yubin, W. (2022). Calibration of RGB-D Camera Using Depth Correction Model. *Journal of Physics: Conference Series*, 2203. doi:10.1088/1742-6596/2203/1/012032
- YodaYoda. (2020). From depth map to point cloud. *Map for Robots*.
- Yu, Z., Chen, A., Antic, B., Peng, S., & Bhattacharyya, A. (2022). *SDFStudio: A Unified Framework for Surface Reconstruction*. Retrieved from <https://github.com/autonomousvision/sdfstudio>
- Yu, Z., Chen, A., Antic, B., Peng, S., & Bhattacharyya, A. (2022). *Supervision*. Retrieved from Github: <https://github.com/autonomousvision/sdfstudio/blob/master/docs/sdfstudio-methods.md#supervision>
- Yu, Z., Chen, A., Antic, B., Peng, S., Bhattacharyya, A., Niemeyer, M., . . . Geiger, A. (2022). *Methods*. Retrieved from Github: <https://github.com/autonomousvision/sdfstudio/blob/master/docs/sdfstudio-methods.md>
- Yu, Z., Peng, S., Niemeyer, M., Sattler, T., & Geiger, A. (2022). *MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction*. doi:10.48550/arXiv.2206.00665