DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Deep Learning Based 3D Reconstruction Using Images With Known Poses

Başak Melis Öcal

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Deep Learning Based 3D Reconstruction Using Images With Known Poses

Deep-Learning-basierte 3D-Rekonstruktion mittels Bildern mit Bekannten Posen

Author:	Başak Melis Öcal
Supervisor:	Prof. Dr. Stefan Leutenegger
Advisor:	Lukas Köstler
Submission Date:	March 15, 2022

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, March 15, 2022 Başak Melis Öcal

Acknowledgments

I would like to express my sincere gratitude and appreciations to my supervisor Prof. Dr. Stefan Leutenegger for his invaluable guidance and giving me the opportunity to write this thesis in his lab. I would also like to extend my gratitude to my advisor Lukas Köstler, who has continuously supported the thesis. His crucial advice along with the constructive comments were really valuable. During the course of my Master's thesis, other than providing me with their unparalleled support, they also offered me an opportunity to always take part in inspiring discussions with them.

I would also like to thank my friends for their genuine support during my thesis. Last but not least, I would like to thank my family, for their continuous encouragement and support for me.

Abstract

3D scene reconstruction from multi-view images is a vital task in 3D computer vision with many fields of application, varying from augmented reality to robotic navigation. While learning-based volumetric approaches have achieved promising results in this domain, their ability to preserve fine-scale details while reconstructing the overall scene, is still limited for a number of reasons. A major issue is the cubically growing memory footprints and compute requirements with the increasing resolution. The gap between the representations used for network supervision and final geometry evaluation is another problem. While the discretized TSDFs employed for supervision, efficiently encode the distance information, they fall short in representing 3D geometry with high-level details when compared to triangular meshes or depth maps used for evaluation. In this work, we extensively investigate the factors affecting the quality of reconstructions by focusing on mathematical as well as structural properties of the employed scene representations, and propose several approaches addressing each one of them. We then selectively combine the presented methods into a single approach targeting the neural detailed 3D scene reconstruction. By minimizing the distance between surface samples of the predicted meshes and back-projected ground-truth depth maps, we combine the voxel-based TSDF supervision with an additional surface geometry supervision. We introduce an alternative to the TSDF fusion algorithm which estimates the target TSDF values on-the-fly using depth maps. The proposed auxiliary loss functions enforce the network to output the correct sign for the distance values. We evaluate our approach extensively on the widely employed InteriorNet and ScanNet datasets. Our quantitative and qualitative evaluations show that the proposed approach is able to reconstruct fine-scale details together with the coarse structures from many complex scenes, and delivers superior performance on InteriorNet and similar performance on ScanNet datasets when compared to state-of-the-art.

Kurzfassung

Die Rekonstruktion von 3D-Szenen aus Bildern mit mehreren Ansichten ist eine wichtige Aufgabe im Bereich des 3D-Computersehens mit vielen Anwendungsbereichen, die von der erweiterten Realität bis zur Roboternavigation reichen. Während lernbasierte volumetrische Ansätze in diesem Bereich vielversprechende Ergebnisse erzielt haben, ist ihre Fähigkeit, feine Details zu erhalten und gleichzeitig die Gesamtszene zu rekonstruieren, aus einer Reihe von Gründen immer noch begrenzt. Ein Hauptproblem ist der mit zunehmender Auflösung kubisch wachsende Speicher- und Rechenbedarf. Ein weiteres Problem ist die Diskrepanz zwischen den für die Netzüberwachung und die endgültige Geometriebewertung verwendeten Darstellungen. Während die diskretisierten TSDFs, die für die Überwachung verwendet werden, die Entfernungsinformationen effizient kodieren, sind sie im Vergleich zu Dreiecksnetzen oder Tiefenkarten, die für die Auswertung verwendet werden, unzureichend in der Lage, die 3D-Geometrie mit hochgradigen Details darzustellen. In dieser Arbeit untersuchen wir ausführlich die Faktoren, die die Qualität der Rekonstruktionen beeinflussen, indem wir uns auf die mathematischen und strukturellen Eigenschaften der verwendeten Szenendarstellungen konzentrieren und mehrere Ansätze vorschlagen, die sich mit jedem einzelnen dieser Faktoren befassen. Anschließend kombinieren wir die vorgestellten Methoden selektiv zu einem einzigen Ansatz, der auf die neuronale detaillierte 3D-Szenenrekonstruktion abzielt. Durch die Minimierung des Abstands zwischen den Oberflächenmustern der vorhergesagten Netze und den rückprojizierten Tiefenkarten der Bodenwahrheit kombinieren wir die voxelbasierte TSDF-Überwachung mit einer zusätzlichen Überwachung der Oberflächengeometrie. Wir führen eine Alternative zum TSDF-Fusionsalgorithmus ein, die die TSDF-Zielwerte on-the-fly mit Hilfe von Tiefenkarten schätzt. Die vorgeschlagenen Hilfsverlustfunktionen zwingen das Netzwerk dazu, das richtige Vorzeichen für die Distanzwerte auszugeben. Wir evaluieren unseren Ansatz ausgiebig mit den weit verbreiteten InteriorNet- und ScanNet-Datensätzen. Unsere quantitativen und qualitativen Auswertungen zeigen, dass der vorgeschlagene Ansatz in der Lage ist, feine Details zusammen mit den groben Strukturen vieler komplexer Szenen zu rekonstruieren, und im Vergleich zum Stand der Technik eine überragende Leistung bei InteriorNet und eine ähnliche Leistung bei ScanNet-Datensätzen liefert.

Contents

1. Introduction

1.1. Problem Statement and Motivation

3D reconstruction of a scene, which is to recover the complete and accurate 3D geometry of an environment from the given visual input, is a vital task in computer vision and graphics that has been extensively investigated over the years. Owing to the difficulty in simultaneously representing different levels of details, it has become a long-standing challenge for prior work. However, driven by the recent developments in computer vision and deep learning, considerable advances have been also made towards 3D scene reconstruction, fostering various application areas including augmented reality [1], motion planning [2] and robotic navigation [3].

For many of these fields of application, reconstructing 3D scenes faithfully by preserving fine-scale details along with coarse structures, is of crucial importance. In robotic applications, for instance, to navigate safely and efficiently, 3D scene reconstruction needs to be accurate and coherent. While coarser reconstructions may be sufficient for navigating in fields with only larger obstacles, reconstructing fine-scale details are needed for scenarios requiring tricky maneuvers in between relatively smaller objects. Similarly, in augmented reality, intending to provide realistic high-level interactions with the surrounding environment, 3D reconstructions should be able to accurately represent details within a scene.

One of the mostly employed strategies towards dense 3D reconstruction is, using sensors such as LiDAR to obtain depth measurements, and then fusing these measurements into a 3D representation. Although, these sensors can provide extremely precise measurements in the absence of occlusions or reflecting areas, they are more expensive and less ubiquitous than RGB cameras, and thus not adopted by many of the state-of-the-art systems. With the rise of deep learning, learning-based multi-view stereo (MVS) approaches that rely on RGB images to predict per-view depth maps and then fuse the predicted depth maps to extract the final 3D surface, have dominated the field. These approaches often follow a common pipeline based on plane-sweep algorithm under the assumption of photo-consistency [4, 5, 6, 7, 8, 9, 10]. Although these approaches have achieved significant improvements, they require depth maps as intermediate representations and accuracy of the depth estimation falls behind the depth sensors.

As another line of MVS studies, voxel-based approaches, that directly regress Truncated Signed Distance Function (TSDF) volume to overcome the limitations of using intermediate representations, are developed. More recently, *Atlas* [11] and its follow-up work *NeuralRecon* [12] propose a voxel-based pipeline where extracted features from 2D Convolutional Neural Networks (CNNs) are back-projected into a global volume to be further processed by 3D networks to regress the final TSDF volume. Different from *Atlas*, *NeuralRecon* incorporates

3D sparse convolutions for efficiently processing the 3D volume and follows a coarse-to-fine hierarchy to refine the TSDF volume at each level.

Although voxel-based MVS approaches have shown significant improvements for the task of 3D scene reconstruction, their ability to represent fine-scale details is still limited for a number of reasons. A major issue is the cubically growing memory footprints and compute requirements, which limit the resolution of final surface meshes, and thus result in losing detailed structures. Another problem is using different representations for network supervision and final geometry evaluations. The discretized TSDF volume used for supervision can be disadvantageous for representing high-level 3D geometry when compared to triangular meshes or depth maps used for evaluation. Considering that the majority of the scenes in real-life scenarios consists of both fine-scale details and coarse structures, obtaining a solution for the problem of detailed 3D reconstruction in complex scenes is essential.

In this work, we intend to tackle the challenging task of voxel-based detailed 3D scene reconstruction by extensively investigating a number of factors affecting the quality of reconstructions and proposing several methods based on these observations to further improve the accuracy.

1.2. Proposed Methodology Overview and Contributions

To the best of our knowledge, this is the first work extensively analyzing the factors towards reconstructing detailed 3D scenes from a voxel volume and proposing several methods addressing each of these factors, with the purpose of enhancing the reconstruction accuracy as well as coherence. To this end, we exploit several mathematical along with structural properties of the employed scene representations, then propose and investigate the following:

- TSDF & Occupancy Supervision:
  - Our method involves both TSDF & occupancy supervision. Aiming to obtain more consistent ground-truth TSDFs, we propose generating the target TSDFs from the ground-truth 3D surface meshes by finding the distance to the closest triangle. We examine whether the TSDF gradient magnitudes can be used to obtain more consistent TSDF predictions via enforcing the Eikonal property [13]. We also propose using gradients directly for supervision as an auxiliary signal to TSDF and occupancy losses.
  - In a different direction, we investigate whether the supervisory signal is oversimplified or overcomplicated for the task at hand.
- Surface Geometry Supervision:
  - We additionally propose a surface geometry supervision strategy where we densely sample both the target and predicted surface meshes to minimize the distance between the surface samples. We benefit from the Cauchy function to mitigate the effect of outliers.

  - Several auxiliary loss functions and modifications to the original TSDF loss function are proposed to enforce the sign of the per-voxel TSDF predictions to be the same with the target.

- Depth-based Supervision:
  - To benefit from the actual sensor measurements instead of meshes from the Marching Cubes [14] algorithm, we also consider using depth maps for surface geometry supervision. To this end, we propose minimizing the distance between the surface samples of the predicted mesh and back-projected ground-truth depth maps. In the same direction, we propose estimating the target TSDF values on-the-fly using the depth maps to remove the dependency to the TSDF fusion [15] algorithm which may introduce inaccuracies to the TSDF supervision owing to the many approximations it involves.
  - We benefit from the depth maps while sparsifying our voxels to be processed by sparse convolutions.

- Resolution: The code basis of this work has a three-level coarse-to-fine hierarchy. We investigate the effect of having a finer resolution TSDF representation by inserting an additional fourth layer and employing a 3D network for neural upsampling separately.

Succinctly, we deliver the following **contributions** to prior work:

- We extensively investigate a number of factors affecting the quality of 3D scene reconstructions by focusing on mathematical and structural properties of the employed scene representations. We then propose several methods addressing these factors to enhance the quality of reconstructions in a way to preserve fine-scale details as well as coarse structures.

- We show that common TSDF supervision when combined with surface geometry supervision either by using ground-truth depth maps or 3D meshes, achieves promising results enhancing the quality of reconstructions. We demonstrate that the network can be explicitly enforced to output the correct sign for the distance values when the proposed sign enforcing auxiliary loss terms are employed. This brings a substantial boost to the performance of the regular TSDF supervision. We present an alternative approach to the conventional TSDF fusion algorithm, removing the need for an additional pre-processing step.

- We selectively combine the proposed approaches into a single final approach to target several factors simultaneously.

- We evaluate the final proposed approach, both quantitatively and qualitatively, on the widely employed datasets InteriorNet [16] and ScanNet [17] using the common 2D & 3D metrics. We also present the evaluation results of the individually proposed methods on InteriorNet.

Our quantitative and qualitative evaluations show that our final approach is able to reconstruct fine-scale details as well as coarse structures from many complex scenes, and delivers superior results when compared to state-of-the-art on InteriorNet dataset.

1.3. Outline

This thesis is structured as follows:

In **Chapter 2**, we provide an overview of the related work. We introduce multi-view stereo approaches in section 2.1, by specifically focusing on depth-based and voxel-based methods. We discuss 3D surface reconstruction in section 2.2 and neural implicit representations in section 2.3.

In **Chapter 3**, we summarize the key concepts required for understanding the task of detailed 3D scene reconstruction, such as the camera model and scene representations. Additionally, we introduce the base architecture which we built our methods upon and describe the datasets employed in this work.

In **Chapter 4**, we first provide an overview of the proposed methods, mainly summarizing the underlying observations leading to a particular approach along with the research questions. Then, we provide in-depth descriptions of the proposed methods.

In **Chapter 5**, we start by introducing our experimental methodology, and then continue delivering the results of our experiments. We present the quantitative evaluation results of the proposed methods and describe how we selectively combine these methods into a final approach. We also present the comparisons with the state-of-the-art.

In **Chapter 6**, we briefly summarize this work by further highlighting our contributions and we propose several further research directions.

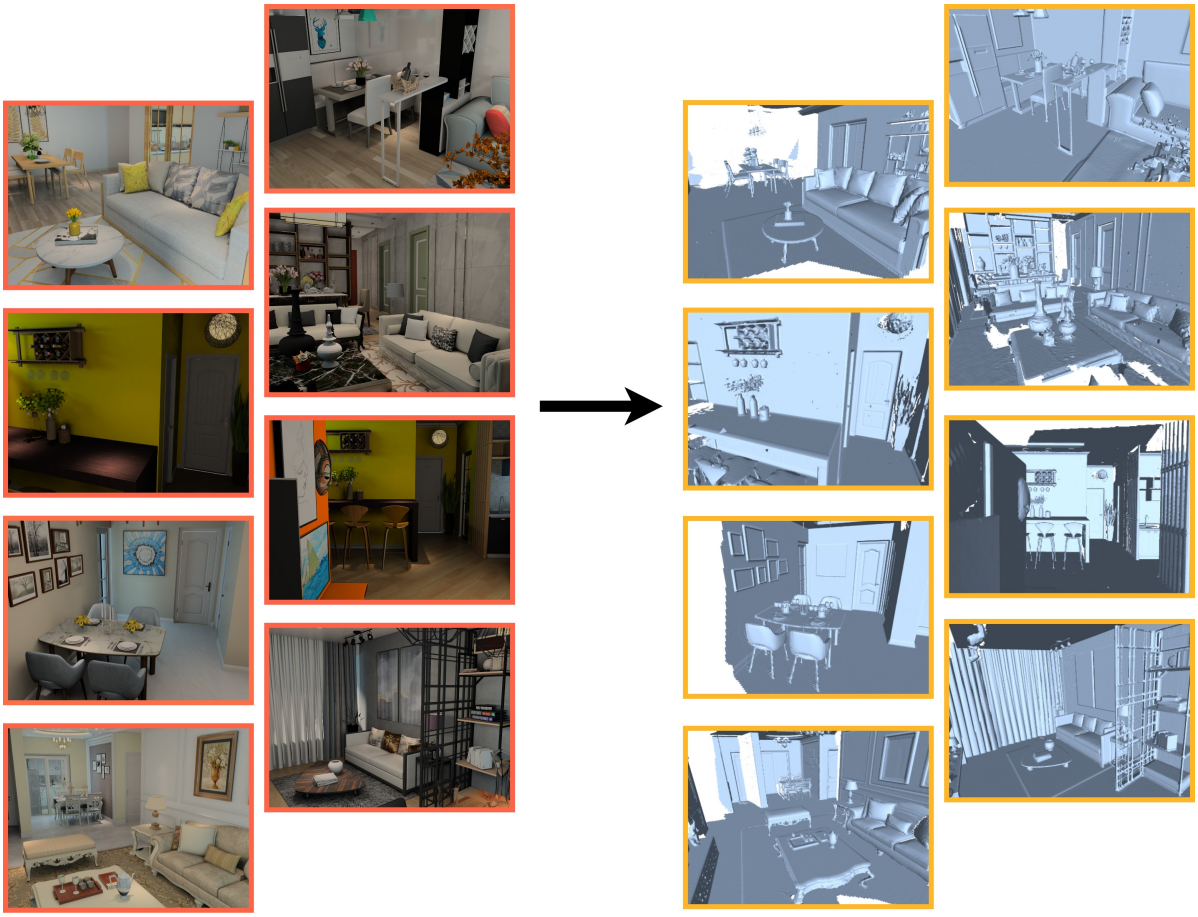

Figure 1.1.: **3D scene reconstruction task.** In this work, we intend to reconstruct 3D scenes from multi-view images, while preserving both fine-scale details and coarse structures simultaneously.

2. Related Work

In this work, we deal with neural detailed 3D scene reconstruction using images with known poses. Before presenting the proposed approaches in the following sections, we discuss the well-established related methods in 3D reconstruction by highlighting their connections with our work. We first provide an overview of multi-view stereo approaches, by specifically focusing on depth-based and voxel-based methods. We also cover 3D surface reconstruction approaches, emphasizing their relationship with multi-view stereo. Lastly, we discuss neural implicit representations.

For our implementations, we use the architecture from *NeuralRecon* as our code basis, and thus we particularly highlight it. We benefit from the differentiable surface parametrization approach presented in *MeshSDF*. The proposed methodology employs a neural implicit field represented on a 3D voxel grid. We use *NeuralRecon* as our main baseline, since our methods are in the same line of work with it.

2.1. Multi-view Stereo

Multi-view stereo (MVS) has been a long-studied research field in computer vision, which intends to reconstruct the 3D geometry of a scene using a set of overlapping images captured from various viewpoints with known camera parameters. MVS problems can be considered equivalent to the correspondence search problem by optimizing geometric or photometric consistency [18]. Considering final scene representations, a taxonomy of MVS algorithms are introduced in [19], which categorizes them into: depth map [20], point cloud [21, 22], mesh [23] and voxel based [24] methods.

2.1.1. Depth-based Multi-view Stereo

In depth-based MVS studies, the task is mainly divided into two major steps: 1) estimating a 2.5D depth map for each of the views by extracting the geometric information from multi-view images, and 2) fusing depth maps to generate a final 3D surface. Though prior work proposes a variety of contributions to the first step, methods generally employ a common pipeline consisting of four main stages: *feature extraction*, *cost volume construction*, *cost volume regularization* and *depth regression & post-processing*.

In feature extraction, either conventional 2D Convolution Neural Networks (CNNs) or more sophisticated architectures to extract multi-level features maps such as Feature Pyramid Networks (FPNs) and Spatial Pyramid Pooling (SPP) modules are adopted. Then, a cost volume is formed using the *plane-sweep* method which uses a differentiable warping operation. More specifically, given a set of images with known camera parameters, depth hypotheses

are obtained by warping extracted feature maps to the reference camera view. The warping operation is based on photo-consistency, which refers to pixel values from multi-view images that correspond to the same 3D points having similar values. A cost metric is formulated between warped and reference features to generate the volume. The built cost volume is then regularized either sequentially or globally to mitigate noise related problems. Having regularized, depth maps are regressed and further refined using post-processing techniques such as outlier removing, eliminating inconsistent values etc. After a collection of depth maps are estimated, depth map fusion algorithms are employed to reconstruct the 3D scene.

Conventional Multi-view Depth Estimation

Previous studies focusing on depth map estimations have achieved improvements using plane-sweep method [25, 26, 27]. In [28], surfaces which are not fronto-parallel but slanted are handled via performing multiple plane sweeps, where sweeping directions are aligned to the normals of the planar surfaces. [20] filters depth maps before the fusion by listing several per-pixel depth hypotheses and enforcing a spatial consistency constraint to choose the true depth. [29] introduces a depth propagation method using geometric guidance term to obtain initial depth hypotheses, and then employs the proposed rolling shutter plane-sweep algorithm.

In order to solve MVS problems in memory-constrained environments, for instance, when images with relatively higher resolutions are used, patch-matching approaches are developed [30, 31]. These approaches benefit from quickly matching patches instead of exhaustively browsing all possible disparity values to select the most suitable one [32]. Although, the well-known *COLMAP* [30] and many other patch-based methods exhibit robust depth estimation performances, they suffer in the presence of indistinctive color features, low-textured and reflective regions.

Learning-based Multi-view Depth Estimation

With the rise of deep neural networks, several promising learning-based approaches in MVS have been proposed. As one of the pioneering works in literature, *MVSNet* [4] follows the previously mentioned standard depth estimation pipeline with the purpose of 3D reconstruction. First, image features are extracted using a 2D Convolutional Neural Network (CNN), then the 3D cost volume is constructed using differentiable homography warping and a variance-based cost metric to incorporate feature differences. 3D CNNs are employed to regularize the built cost volume and to regress an initial depth map. Depth map is further refined using reference image to prevent over-smoothing at the boundaries and to recover the details.

Concurrently to *MVSNet*, *DPSNet* [33] is introduced for the task of dense depth map estimation. Unlike *MVSNet*, the cost volume is constructed by concatenating feature maps of the reference view and warped image features directly. Also, all cost slices of the volume are refined via a context network using the reference view features to mitigate the problems caused by indistinctive regions where the texture is missing.

DeepMVS [5], employs a 3-level disparity map estimation pipeline. First, plane-sweep volumes are obtained, and then patch matching network is used to extract features from patch pairs required for intra-volume aggregation. Using max-pooling, different volumes are fused to predict per-pixel disparity values. Raw disparity maps are post-processed by Conditional Random Fields (CRFs) which enforces the neighboring pixels to have similar disparity values.

Although the regularization steps of *MVSNet* and *DPSNet* enable extremely accurate depth estimations, they also require a high amount of memory and compute power owing to using 3D CNNs on cost volumes. Therefore, as a follow-up work, *R-MVSNet* [6] is introduced to reduce the memory requirement of *MVSNet*. Instead of regularizing the 3D cost volume at a single step, *R-MVSNet* employs *Gated Recurrent Units* to regularize the 2D cost maps in a sequential manner which mitigates the memory consumption issue.

Different from its predecessors built upon *MVSNet*, *Point-MVSNet* [7] presents a point-based depth map refinement, where low-resolution depth predictions are converted to point clouds and initial point clouds are iteratively refined according to the target point cloud using the proposed *PointFlow* module. For each point, 3D flow to the ground-truth surface is predicted using image features from the FPN and 3D geometry priors. As another follow-up work to *MVSNet*, *CasMVSNet* [8] constructs the initial cost volume via sparsely sampled depth hypotheses, and then adaptively adjusts the sampling range of depth hypotheses at each stage to generate volumes with finer details aiming to overcome higher memory consumption. In contrast to *CasMVSNet*, [34] attempts to use an adaptive depth hypothesis range for each pixel instead of each stage. The range is determined by considering the variances of the predicted per-pixel probabilities. By using an adaptive strategy in the level of pixels rather than stages, more accurate depth map predictions are obtained.

Other than multi-view overlapping images, video streams with known camera motion can be also used as an input MVS approaches. Thus, leveraging video frames, *MVDepthNet* [9] generates a cost volume using raw pixel-level observations rather than the extracted features as in the aforementioned approaches. Then, cost volume along with the reference image are passed to a 2D encoder-decoder architecture to estimate the inverse depth map.

Building upon *MVDepthNet*, [35] proposes combining the encoder-decoder network in [9] with a Gaussian Process (GP) at the bottleneck to enable the network to fuse features from frames with similar poses to prevent needless time or memory consumption. To achieve this, a GP prior kernel is defined on the latent space from a similarity measure that considers the overlapping fields of view. This way, the network is encouraged to produce similar latent representations for close by frames than frames far away from each other.

Conversely to *MVDepthNet*, [36] estimates a per-pixel depth probability distribution rather than per-pixel depth values. For each input frame, first, a *Depth Probability Volume (DPV)* is generated by *D-Net*. Generated DPVs are accumulated via Bayesian filtering approach where the difference between predicted DPV from the earlier frames and estimated DPV by *D-Net* is used to obtain the residual. The residual is modified and added back to the predicted DPV via *K-Net* which is then further refined for predicting depth and uncertainty.

DeepVideoMVS [10], estimates depth from posed video streams using an FPN and builds the cost volume with a traditional plane-sweep stereo using the extracted features. The proposed

encoder-decoder architecture, in which a ConvLSTM module is placed at the bottleneck layer, regularizes the cost volume while enabling spatio-temporal information flow between frames. Since capturing the pose-induced image motion between two encodings can be challenging for ConvLSTM, *DeepVideoMVS* considers viewpoint changes while propagating the hidden state to the next time step by warping the hidden state to the current viewpoint using the previous depth prediction.

ESTDepth [37] proposes jointly estimating depth maps of consecutive frames to provide temporal coherence. First, a raw cost volume is constructed, and then further regularized by *MatchNet* to learn 3D local matching information. In parallel, *ContextNet* extracts 2D context information of the target image. Obtained matching and context volumes are then fused to a hybrid cost volume. The introduced *Epipolar Spatio-Temporal transformer* encodes both query and neighboring volumes, and then epipolar warps the encoded adjacent cost volumes into the query volume's camera coordinate space. Using 3D attention on query and warped volumes, an enhanced query volume is obtained and finally fused with the retrieved values from all warped volumes to regress the depth maps.

2.1.2. Voxel-based Multi-view Stereo

Volumetric MVS approaches adopts a voxelized 3D space to represent the geometry as a whole and attempts to directly predict occupancy probability or signed distance value to the closest surface for each voxel of this discretized space. Our work also employs a 3D voxelized grid to store the signed distance values.

In *SurfaceNet* [38, 39], a volume occupancy prediction network, which outputs a binary value for each voxel deciding whether the voxel is on the surface or not, is introduced. Then the surface is reconstructed from the voxelized 3D volume. More specifically, each input image is unprojected to a per-view voxel grid called *Colored Voxel Cube (CVC)* that stores the color values of corresponding pixels. Obtained CVCs are then paired and sent to a network for occupancy probability prediction. For the multi-view case, weighted averaging is used to fuse predicted probabilities from pairs using a triplet network that outputs relative weight of each pair accounting for viewpoint changes and occlusions.

Similar to *SurfaceNet*, *3D-R2N2* [40] predicts volumetric occupancy. A 2D CNN is employed to encode features into a latent space, and then encoded features are fed into Long Short-Term Memory (LSTM) units at the bottleneck to selectively update hidden states considering visible parts of objects. Hidden states are decoded to output the final voxel occupancy grid. In contrast to *3D-R2N2* which uses mainly semantic features for occupancy prediction, [41] also focuses on extracting geometric features. To this end, encoded features are unprojected into per-view 3D feature grids by rasterizing the viewing rays. The employed differentiable unprojection operation implicitly encodes the projective geometry information. Then, feature grids are fused recurrently into a single feature grid to be further processed by 3D CNNs. From the processed grid, voxel occupancies along with depth maps are predicted.

Since memory and compute requirements cubicly grow with resolution, voxel-based methods are limited to lower resolutions which hampers them from representing larger scenes and finer details. To alleviate these limitations, *Octtree-Gen* [42] employs an octree data

structure to increase the resolution up to 512^3 voxels.

Inspired by the aforementioned approaches, *Atlas* [11] proposes an offline MVS approach to directly predicts Truncated Signed Distance Function (TSDF) volume along with semantic segmentation labels for indoor scene reconstruction. In the presented pipeline, extracted features of an entire sequence of images by a 2D CNN, are back-projected into a voxel grid and aggregated using a weighted running average. Accumulated features are passed to a 3D encoder-decoder architecture to further process the volume and then to regress TSDF values.

More recently, *NeuralRecon* [12], which builds upon *Atlas*, proposes a three-level coarse-to-fine real-time 3D scene reconstruction framework to increase the density of sparse voxel grid at each level. By using multi-level features extracted from a set of key-frames in local windows, 3D feature volumes are generated with back-projection. In order to process in real-time, feature volumes are processed by 3D sparse convolutions instead of a standard 3D CNN, which prohibits running efficiently in the presence of large scenes. At each level of the coarse-to-fine hierarchy, feature volumes are concatenated with the upsampled TSDF volume from the previous level and then provided to the Gated Recurrent Unit (GRU) fusion module for global aggregation. Multi-Layer Perceptron (MLP) layers predict both per-voxel occupancy probabilities and TSDF values from the updated hidden states.

As a follow-up work, *TransformerFusion* [43] employs a resembling incremental architecture but adopts a transformer network to fuse coarse and fine level back-projected features separately into a 3D voxel grid. The proposed transformer-based fusion enforces accurate reconstructions by attentively accumulating the most informative features for particular spatial locations.

2.2. 3D Surface Reconstruction

Having estimated depth maps using depth-based MVS approaches, depth fusion algorithms are adopted as the next step to integrate these values into a volume. Then, the 3D surface meshes are extracted. For this last step, offline multi-view approaches, such as [30], often utilizes Poisson reconstruction [44, 45] and Delaunay triangulation [46] to reconstruct the final surface. Although Poisson reconstruction is widely preferred by relatively early methods owing to its hole filling property via plane fitting, it poorly infers missing higher-level structures with finer details such as chair legs [47].

In the seminal work [15], an incremental volumetric fusion approach to accumulate depth maps into a TSDF volume, is presented. Owing to its simplicity and availability of low-cost depth sensors, many works following *TSDF fusion* are introduced. As one of these works, *KinectFusion* [48] fuses captured RGB-D images by a Microsoft Kinect sensor into a 3D volume in real-time using GPGPU processing. More specifically, once the global pose of the camera is obtained using *Iterative Closest Point* method, pixel coordinates of depth measurements are converted into global coordinates to compute vertex and normal maps. Then, these measurements are integrated into a 3D volume which stores a running average of each voxel's signed distance value. The surface is extracted via the *ray casting algorithm*. Later works such as *BundleFusion* and voxel hashing [49, 50] based methods, focus on improving the

performance of *KinectFusion* in terms of robustness and scalability. Although the above-mentioned approaches are preferred by many 3D reconstruction techniques based on TSDFs owing to their parallelization capability, their reconstruction quality drops significantly in the presence of occluded geometry and inadequate amount of measurements.

To alleviate the compute problems of prior work, *OctNet-Fusion* [51] benefits from the data sparsity and employs *grid-octree* data structure where several octrees, whose maximal depths are restricted to a small number, are placed over a regular grid. Once the feature volume computed from depth maps are stored in a *grid-octree*, fused geometry is post-processed by a 3D encoder-decoder architecture hierarchically to obtain further contextual information. Processed features are passed to the introduced *structure module* which adaptively decides whether further subdivision is required or not considering voxel's distance to the surface. This way, complete surfaces can be reconstructed using sub-voxel estimates that provide higher resolutions. In contrast to *OctNet-Fusion* which reconstructs single objects, *ScanComplete* [52] is proposed to reconstruct large scenes leveraging a similar post-processing idea for scene completion.

Similar to the aforementioned work targeting scene completion, *SG-NN* [53] approaches the same problem from a self-supervised perspective using real-world datasets. From the target RGB-D scan which is incomplete by nature due to occlusions etc., a more incomplete input scan is obtained by removing a portion of the frames. The proposed generative network, first, encodes the input scan which is represented as a TSDF volume with 3D sparse convolutions [54] and then converts to a dense 3D grid at a very coarse resolution. Dense convolutions produce feature maps from which coarse occupancy and TSDF values of the completed scan are predicted. Using coarse geometry predictions, sparse representation of the scene is reconstructed and passed to coarse-to-fine hierarchy where each stage outputs the geometry of the next resolution.

RoutedFusion [55] introduces a data-dependent depth fusion approach with real-time capability. Given a raw depth map, an outlier filtered and denoised depth map along with the confidence values are produced by a routing network, which also decides on the update location of TSDF values along each ray. Then, voxel data is extracted for each ray and passed to a fusion network together with corrected depth and confidence maps to predict non-linear updates. As the final step, predicted TSDF updates are fused into the global TSDF volume.

More recently, *NeuralFusion* [56] proposes having a different representation than the output scene representation for geometry fusion. Therefore, unlike *RoutedFusion*, [56] prefers fusing extracted features into a latent space rather than fusing depth information into a TSDF volume. Then, a *translator network* decodes the latent feature space into the output representation. With the proposed decoupling of representations, higher noise levels are handled and reconstructions with thin geometry are obtained.

Having obtained the TSDF volume, the Marching Cubes (MC) algorithm can be employed to extract the mesh from this implicit function by detecting the zero-crossing of the surface along voxel grid edges [14]. Although MC algorithm is widely adopted, it is not suitable for end-to-end training as the linear interpolation used for deciding on vertex positions is not differentiable with respect to topological changes [57]. To enable surface topology

modifications, [57] introduce a learned probabilistic MC surface extraction approach by adding a final layer to a 3D network. Unfortunately, extracted meshes are limited by resolution as the proposed approach requires keeping track of all the topologies at the same time [58]. *Voxel2Mesh* [59] deforms an initial spherical mesh incrementally by adaptively increasing the resolution, which overcomes the limited resolution problem but prevents topological changes.

MeshSDF [58] and its follow-up work *DeepMesh* [60] overcome the previous limitations of differentiable surface extraction methods by focusing on how an infinitesimal perturbation on an Signed Distance Function (SDF) field affects geometry of the local surface. More specifically, they formulate a closed-form expression for the non-differentiable term, which is the gradient of mesh vertices with respect to the SDF field, within the backpropagation step. This way, the non-differentiable MC algorithm can be used to extract the surface mesh, and then gradients can be backpropagated all the way from mesh to the introduced latent vectors using a custom backward pass while allowing changes to topology.

2.3. Neural Implicit Representations

Recently, implicit representations which represent surfaces of 3D objects as the continuous decision boundary, are employed by many studies to generate high-resolution surfaces from SDFs without huge computation and memory costs. Implicit representations are also able to represent shapes with finer details without discretization artifacts.

To this end, [61] attempts to assign occupancy probabilities to every point in 3D space to decide whether they are on the surface or not. During training, a set of points are sampled uniformly from the 3D bounding volume of the object and then binary cross-entropy loss is computed at these sampling locations. At inference, an incremental octree building algorithm called *Multiresolution IsoSurface Extraction (MISE)*, is employed to extract the surface which, first, predicts occupancy probabilities for all discretized locations at the initial resolution and marks the occupied ones. Voxels whose grid points having differing occupancy predictions are further divided until the intended resolution is provided. The algorithm eliminates the need for evaluating densely at all points of an occupancy grid.

Instead of predicting occupancy probabilities, *DeepSDF* [62] focuses on regressing an SDF value for any given spatial point. In the proposed auto-decoder based formulation, random latent codes are attached to the query point and fed into the network which is supervised by the ground-truth SDF values at the query locations. During inference, decoder weights are kept fixed and latent codes are optimized using Maximum-a-Posterior (MAP) estimation. As inference can be performed using an arbitrary number of points, the optimized latent codes can be given to the decoder whose priors are encoded to complete the partial input shapes. As a follow-up work to *DeepSDF*, *DeepLS* [63] intends to overcome the limitations of using a single latent code, such as incapability to represent large scenes owing to scaling and generalization issues.

Building upon the observation that different shapes having different geometric cues on a global scale exhibit similar properties at local scales, [63] first encodes TSDF voxel crops using a 3D CNN. Then, produced latent codes and query point coordinates are fed into a

decoder which is supervised using binary *in/out* labels. To prevent discontinuities caused by partitioning the space into local grids, overlapping grid structure where each grid overlaps with its neighbors are proposed. Latent codes of grids are optimized at inference time similar to [62].

As a follow-up work, *PIFu* [64] tackles the problem of digitizing detailed clothed humans to recover textured 3D surfaces. Extracted per-pixel features along with depth values from these pixels are used to query the MLP network to predict continuous inside/outside probability field of a clothed human together with corresponding RGB values.

To eliminate the need for 3D supervision, [65] directly uses raw point cloud data by employing a loss to enforce the predicted zero-level set to pass through the input points. The proposed method uses an iterative sampling strategy to sample to zero-level set. Similar to [65], *SAL* [66] uses raw point clouds or triangle soups and performs unsigned regression to find a desirable local minimum of the implicit function without 3D supervision to reconstruct human scans. Extending *SAL*, [67] formulates a loss motivated by the Eikonal partial differential equation [13]. The loss encourages the implicit function to vanish on the zero-level set with unit norm gradients equal to the normals, which gives a good global minimum when optimized that produces smooth, detailed surfaces.

3. Background

In this chapter, we provide an overview of several core concepts used throughout the thesis. We start with an introduction of the camera model, and then cover the 3D scene representations employed in this work. Later, we introduce the base architecture which we built our methods upon. Our base architecture closely follows [12], except the minor modifications applied within its 2D and 3D networks for efficiency purposes. The datasets used for evaluations are presented in the last section.

3.1. Camera Model

3.1.1. Pinhole Camera Model

In this thesis, simple pinhole camera model is employed, where the camera is modeled as a box and a hole is located on one of the sides [68]. In Figure 3.1, the underlying geometry of the pinhole camera model is depicted.

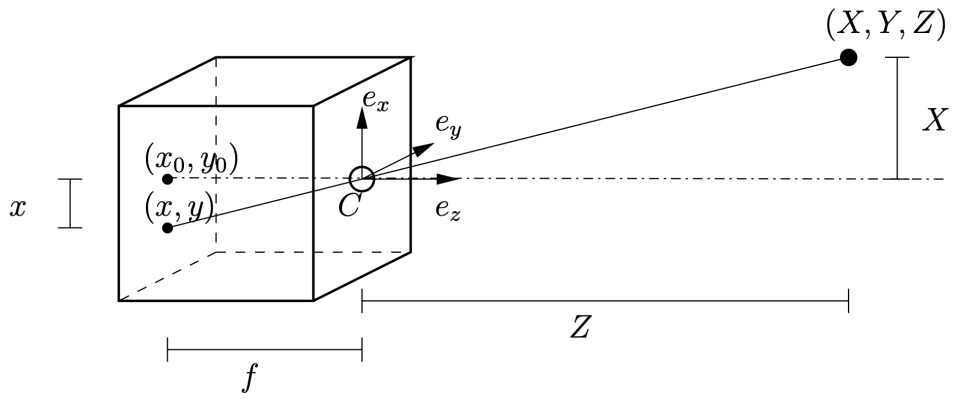

Figure 3.1.: **Pinhole camera model, figure from [68].**

Focal point that represents the coordinate system origin, is at the center of projection and *focal length* denoted by f, defines the distance from the focal point to the image. We obtain Equation 3.1 from the similar triangles, which can be also formulated in a matrix form using homogeneous coordinates as in Equation 3.2, where λ is the depth and $\lambda = Z$.

$$\frac{x}{f} = \frac{X}{Z} \text{ and } \frac{y}{f} = \frac{Y}{Z} \tag{3.1}$$

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.2}$$

3.1.2. Camera Intrinsics and Extrinsics

To provide a formal introduction, camera equation in Equation 3.4 is provided first, where **P** is the camera matrix, **K** is the intrinsic matrix, **x** is the homogeneous pixel coordinates and **X** is the homogeneous world coordinates.

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad \mathbf{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$\lambda \mathbf{x} = \mathbf{K}[\, \mathbf{I}_{3\times 3} \mid \mathbf{0}_{3\times 1} \,]\mathbf{X} = \mathbf{P}\mathbf{X} \quad \text{and} \quad \mathbf{P} = \mathbf{K}[\, \mathbf{I}_{3\times 3} \mid \mathbf{0}_{3\times 1} \,] \tag{3.4}$$

Using the intrinsic parameters, we can replace the intrinsic matrix **K** introduced in Equation 3.3 with a more expressive term as follows:

$$\mathbf{K} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 0 \end{bmatrix} \tag{3.5}$$

where f_x and f_y represent focal length in terms of the number of pixels in x and y directions respectively, (c_x, c_y) is the principal point and γ is the skew coefficient between the axes of the image plane.

To project a 3D point with known 3D world coordinates onto an image plane, these coordinates in the *world coordinate system* are transformed into the coordinate system of the camera which we refer to as the *camera coordinate system*. This mapping between these different coordinate systems is formulated as an Euclidean transformation which uses the translation and rotation information. The parameters required for coordinate system transformation are called as *camera extrinsic parameters*. They are represented with a joint rotation-translation matrix of the form $[\, \mathbf{R} \mid \mathbf{t} \,]$, where rotation and translation matrices are denoted as **R** and **t** respectively.

3.1.3. Perspective Projection

For an ideal pinhole camera, perspective camera model provides a mathematical description of perspective projection operation which maps 3D points in world space to 2D points in image space. Let **x** be the homogeneous coordinates of an image pixel, **X** be the homogeneous coordinates of a 3D point in the world coordinate system and $\mathbf{K}[\, \mathbf{R} \mid \mathbf{t} \,]$ be the perspective camera matrix, the perspective projection is defined as follows:

$$\lambda \mathbf{x} = \mathbf{K}[\,\mathbf{R}\,|\,\mathbf{t}\,]\,\mathbf{X} \tag{3.6}$$

3.2. Scene Representation

Within this thesis, we represent 3D scenes by triangular meshes defined in subsection 3.2.1. We extract meshes by finding the zero crossings of truncated signed distance functions, which we define in subsection 3.2.2. We employ point clouds, which is introduced in subsection 3.2.3, in some of our methods and also for evaluation purposes.

3.2.1. Triangular Mesh

A triangular mesh $\boldsymbol{M} = (\boldsymbol{V}, \boldsymbol{F})$ is represented by a set of vertices and a set of faces connecting them that can be used to explicitly represent objects or scenes in 3D space. Positions of N vertices of a mesh can be represented as a matrix $\boldsymbol{V}$, where each row of $\boldsymbol{V}$ corresponds a vertex $\boldsymbol{v} \in \mathbb{R}^3$ with x, y, z coordinates. The matrix of faces $\boldsymbol{F} \in \{0, ..., N-1\}^{F,3}$ represents each face in a single row with the indices of its three vertices, where F is the number of faces. A mesh model can be also considered as an undirected graph, where $\boldsymbol{V}$ defines the vertex set and $\boldsymbol{F}$ implicitly defines the relationship between the vertices.

3.2.2. Signed Distance Function

A distance function, for a given spatial point, outputs the distance to the closest point on the surface boundary of an object in 3D space [69]. A signed distance function, additionally outputs the sign for the distance to distinguish the inside and outside of an object which can be used to implicitly represent objects or scenes.

Let $\Omega^- \in \mathbb{R}^3$ and $\Omega^+ \in \mathbb{R}^3$ be the space in and outside of an implicit surface respectively, and $\partial\Omega$ be the boundary, the signed distance function $f : \mathbb{R}^3 \to \mathbb{R}$ is defined as:

$$f(x) = \begin{cases} dist(x, \partial\Omega) & \text{if } x \in \Omega \\ -dist(x, \partial\Omega) & \text{if } x \notin \Omega \\ 0 & \text{if } x \in \partial\Omega \end{cases} \quad \text{with } dist(x, \partial\Omega) = \min_{y \in \partial\Omega} = \|x - y\|_2 \tag{3.7}$$

where $dist$ is the Euclidean distance from x to the closest point on $\partial\Omega$. Values of $f(x)$ vanishes on $\partial\Omega$. Function f changes sign depending on whether x is inside or outside of the surface.

The signed distance function $f(x)$ satisfies the Eikonal equation at every point where it is differentiable:

$$\|\nabla_x f(x)\|_2 = 1 \tag{3.8}$$

which encourages the gradients to be of unit norm [70, 71].

For a chosen threshold $f_{thresh} \in \mathbb{R}^+$, the truncated signed distance function truncates the absolute value of the signed distance function at f_{thresh} which is defined as:

$$f_{trunc}(x) = sign(f(x)) \ \min(f_{thresh}, |f(x)|) \tag{3.9}$$

In this work, we store truncated signed distance function representations in voxelized 3D volumes, where each voxel stores the discretized truncated signed distance to the closest surface.

3.2.3. Point Cloud

A point cloud is an unordered set of points which can be used as an explicit representation for objects or scenes in 3D space. A point cloud with N points can be represented as a matrix $\boldsymbol{P} \in \mathbb{R}^{N\times 3}$, where each row of $\boldsymbol{P}$ corresponds to the x, y, z coordinates of a single point. Depending on the application, points can be associated with additional data such as intensity information, RGB color data etc.

3.3. Base Architecture

In this work, we employ a base architecture to build our methods upon. Our base architecture closely follows [12], except the minor modifications applied within its 2D and 3D networks for efficiency purposes. Therefore, before presenting our proposed methods, we provide an extensive overview of [12] for the ease of the reader.

NeuralRecon proposed in [12], is a voxel-based multi-view stereo approach for real-time 3D scene reconstruction, that directly regresses a Truncated Signed Distance Function (TSDF) volume from a monocular video with known per-frame camera poses, without the need of an intermediate depth map estimation step.

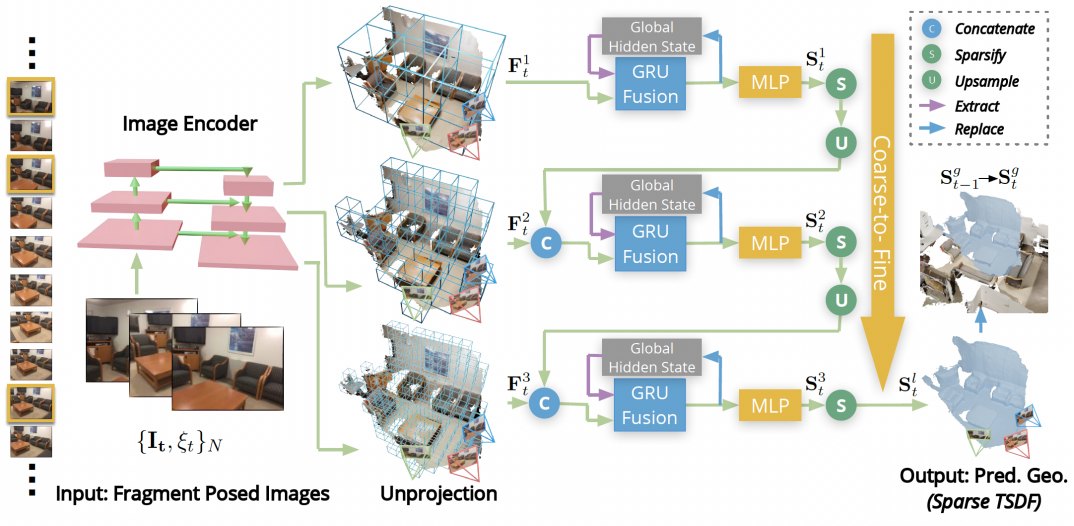

Figure 3.2.: **NeuralRecon architecture, figure from [12].**

More specifically, from a sequence of posed monocular RGB images, first, key-frames are selected for local fragment reconstruction and selected key-frames within this local fragment are fed into a 2D backbone to extract multi-level image features. These features are then back-projected along each ray into a 3D feature volume. At the coarsest level, a dense TSDF volume is predicted from the coarse feature volume. In the following finer levels, feature volume of the corresponding level is concatenated with the upsampled TSDF volume from the previous level and provided to the GRU-based fusion module. Multi-Layer Perceptron (MLP) layers predict both per-voxel occupancy probabilities and TSDF values from the updated hidden states. Overview of the architecture of *NeuralRecon* is presented in Figure 3.2 and further details regarding individual components are delineated in the following subsections.

Key-Frame Selection Contrary to depth-based multi-view stereo (MVS) methods that estimate per-view depth maps, *NeuralRecon* estimates 3D geometry within a local window jointly which brings local coherence to the reconstructions. Let $\{\boldsymbol{I}_t(u, v)\}$ be a sequence of monocular RGB images and $\{\boldsymbol{\xi}_t\} \in \mathbb{SE}(3)$ be the corresponding camera poses, a local window of N key-frames is defined where the relative translation and relative rotation angles of key-frames are larger than the predefined thresholds t_{max} and R_{max}. A Fragment Bounding Volume (FBV) storing view-frustums of N key-frames, is then generated using a depth threshold d_{max}, to reconstruct the region within the volume.

Feature Volume Construction Images within this local window are fed into a 2D backbone, MnasNet [72], that employs Feature Pyramid Network [73] within its architecture to extract multi-level features. The 3D feature volume of each level is generated by back-projecting the obtained features along each ray and using a weighted averaging technique, where the weights are defined based on from how many views a voxel can be seen.

Coarse-to-fine Hierarchy The obtained 3D feature volume $\boldsymbol{F}_t^l$ is processed by 3D sparse convolutions. At the coarsest level, a dense TSDF volume $\boldsymbol{S}_t^1$ is predicted from the feature volume $\boldsymbol{F}_t^1$. In later levels, TSDF volume $\boldsymbol{S}_t^{l-1}$ from the previous level is upsampled by two and concatenated with the feature volume $\boldsymbol{F}_t^l$ of the current level to be given as an input to the GRU-based fusion module. From the updated hidden states, MLPs predict per-voxel TSDF values as well as per-voxel occupancy values to represent the probability of being within the truncation distance ρ, at each level. Voxels of $\boldsymbol{S}_t^l$ are then sparsified, if their occupancy probability is lower than the sparsification threshold θ before being fed to the next level.

GRU Fusion A 3D convolutional version of GRUs, which conditions the current fragment reconstruction on the previous reconstructions is employed to enforce global consistency. Formally defining, the GRU fusion module that fuses 3D geometric features $\boldsymbol{G}_t^l$ obtained from sparse convolutions with the hidden state $\boldsymbol{H}_{t-1}^l$ extracted from the global hidden state $\boldsymbol{H}_{t-1}^g$, is as follows:

$$\boldsymbol{z}_t = \sigma(\text{SparseConv}([\boldsymbol{H}^l_{t-1}, \boldsymbol{G}^l_t], \boldsymbol{W}_z) \tag{3.10}$$

$$\boldsymbol{r}_t = \sigma(\text{SparseConv}([\boldsymbol{H}^l_{t-1}, \boldsymbol{G}^l_t], \boldsymbol{W}_r) \tag{3.11}$$

$$\tilde{\boldsymbol{H}}^l_t = \tanh(\text{SparseConv}([\boldsymbol{r}_t \odot \boldsymbol{H}^l_{t-1}, \boldsymbol{G}^l_t], \boldsymbol{W}_h) \tag{3.12}$$

$$\boldsymbol{H}^l_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{H}^l_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{H}}^l_t \tag{3.13}$$

where $\boldsymbol{W}_*$ is the weights for sparse convolution, σ is the sigmoid function. Represented by $\boldsymbol{z}_t$ and r_t respectively, the update and reset gates decide the amount of information from $\boldsymbol{H}^l_{t-1}$ that will be fused with $\boldsymbol{G}^l_t$ from the current fragment. They also decide on how much information from the current fragment will be integrated into $\boldsymbol{H}^l_t$. Then, by replacing the corresponding voxels, $\boldsymbol{H}^g_t$ is also updated using $\boldsymbol{H}^l_t$.

The final sparsified volume $\boldsymbol{S}^l_t$ from level-3 is incorporated into $\boldsymbol{S}^g_t$ by replacing the respective voxels, as the fusion between $\boldsymbol{S}^l_t$ and $\boldsymbol{S}^g_t$ has been already completed in the fusion step.

The TSDF supervision is applied to all levels of the coarse-to-fine hierarchy using L1 distance between the predicted and the ground-truth TSDF values. Both the prediction and the target are log-transformed before the L1 loss computation. Additionally, binary cross-entropy (BCE) occupancy loss is employed to supervise occupancy predictions at each level. The losses are applied only to the visible regions of the target. The total loss L^l_{tot} at level l has the form given in Equation 3.14, where L^l_{tsdf} and L^l_{occ} are TSDF and occupancy losses respectively with weights λ_{tsdf} and λ_{occ}.

$$L^l_{tot} = \lambda_{tsdf} L^l_{tsdf} + \lambda_{occ} L^l_{occ} \tag{3.14}$$

3.4. Datasets

In this section, we provide an overview of the datasets used in this thesis. We perform our experiments and evaluate our methods on two datasets, namely ScanNet (V2) [17] and InteriorNet [16]. The ScanNet dataset is a widely employed dataset for the task of 3D scene reconstruction which offers richly-annotated scans from real-world environments, while the InteriorNet dataset provides a fully controlled synthetic environment using photo-realistic rendering. Table 3.1 presents the properties of ScanNet and InteriorNet.

3.4.1. InteriorNet

InteriorNet [16] is a dataset that has been developed to address limitations of existing Simultaneous Localization and Mapping (SLAM) benchmarks via providing photo-realism as well as variability and larger scale. The dataset involves approximately 1M CAD furniture models used in the real-world production and 22M interior layouts created from those models via 1,100 professional designers/companies. To manipulate lighting conditions and simulate scene change over time, a number of configurations are generated via a physics engine.

Dataset	InteriorNet	ScanNet
# Training scenes	500	1201
# Validation scenes	120	312
# Test scenes	50	100
# Frames in each scene	1000	changes
Color frame resolution	640 x 480	1296 x 968
Depth frame resolution	640 x 480	640 x 480
Camera intrinsics	✓	✓
Camera extrinsics	✓	✓
Disparity/Depth	✓	✓
Ground-truth mesh	✗	✓

Table 3.1.: **Dataset descriptions of InteriorNet [16] and ScanNet [17].**

InteriorNet data is available in two subsets, and we only use the first subset in this work. The first subset consists of 15K sequences rendered from 10K randomly selected layouts, with 1K images for each sequence. For each layout, multiple trajectories are generated by varying the trajectory type as well as the combination of linear and angular velocities. Different types of lens models and RGB variations are included to dataset. Per-frame ground-truth data including depth obtained as the Euclidean distance of the ray intersection, per-pixel semantic labels, per-object 3D bounding box, instance segmentations and optical flow are available [16].

The main reason of employing InteriorNet is the fact that it provides very accurate ground-truth data including the poses, noise-free depth maps and high-resolution images for all of the trajectories. Therefore, we obtain a setting which is mostly isolated from any kind of noise and inaccuracies that may exist in real-world datasets, e.g. drifted poses, inaccurate depth maps. Additionally, having accurate ground-truth enables creating accurate target 3D surface meshes that present a high amount of details which is ideal for both the supervision and evaluation of our network. This way, unlike the situation for real-world datasets, the qualitatively observable results of our methods can be also reflected to quantitative evaluation metrics.

Having access to sequentially ordered frames per trajectory is another reason of our dataset selection. By processing frames in order, we are able to reconstruct fragments and fuse them sequentially as introduced in our base architecture [12]. When datasets with non-sequential viewpoints are employed, the recurrent fragment reconstruction often fails owing to lack of highly overlapping images.

For the experiments on our final approach, we employ the data with the given properties in Table 3.1. We refer to this data as *InteriorNet-full*. However, since our proposed methodology consists of several approaches, we perform experiments on these individual methods using a smaller subset of data considering the time constraints. Therefore, we employ 200 train & 50 validation scenes sampled from the corresponding splits of the main data and we refer to this

Figure 3.3.: **Sample images and corresponding color coded ground-truth depth maps from InteriorNet dataset [16].**

smaller subset as *InteriorNet-200* for the ease of the reader. Figure 3.3 presents sample frames and their corresponding color coded depth maps from InteriorNet.

3.4.2. ScanNet

For the experiments on our final approach, we employ the ScanNet dataset [17] that provides rich semantic annotations with surface reconstructions, 3D segmentations and camera poses from a large number of scenes. In order to capture and annotate large amounts of 3D data, novice users are provided with a structure sensor which is a commodity RGB-D sensor, attached to an iPad Air2. Aligned poses obtained from BundleFusion [74] are used to perform volumetric integration via VoxelHashing [75] and surface meshes are extracted using the Marching Cubes (MC) algorithm on the TSDF with $4mm$ voxel size. Several filtering steps such as merging close-by vertices and deleting duplicate or isolated mesh parts are applied to obtain cleaner reconstructions. Semantic labeling is completed by crowd workers.

ScanNet consists of 2.5M RGB-D images from 1513 scans collected in 707 distinct indoor spaces with camera poses, calibration parameters, 3D surface reconstructions, textured meshes, semantic segmentations for objects, and CAD models [17]. Additionally, it offers three new benchmarks on several 3D scene understanding tasks: 1) 3D object classification, 2) semantic voxel labeling and 3) CAD model retrieval.

Due to its annotation-rich real-world scans and diversity, ScanNet is widely employed in the field of 3D scene reconstruction. Therefore, it provides a convenient benchmark to compare our final method with the state-of-the-art. In our experiments, we adopt the commonly used train/validation/test splits employed by the previous work [11, 12] for a fair comparison. The test set includes 100 scans with ground-truth scene meshes for benchmarking. Samples from ScanNet are provided in Figure 3.4.

Figure 3.4.: **Sample images and corresponding color coded ground-truth depth maps from ScanNet dataset [17].**

3.4.3. TSDF Generation and Surface Mesh Extraction

Following [12], we generate the target TSDFs using the TSDF fusion implementation of [12] at the preferred resolution, and then extract the target meshes from these TSDFs using the Marching Cubes algorithm [14] for training.

For evaluations, [12] uses the available ground-truth scene meshes provided by ScanNet which are post-processed and further cleaned for benchmarking. Therefore, we also use the available meshes in our evaluations on ScanNet. However, we do not have access to readily available ground-truth meshes for Interiornet data, and thus they should be generated additionally. For evaluations on InteriorNet, we use the TSDF fusion algorithm from Open3D [76] to obtain the ground-truth TSDFs and extract the ground-truth meshes via their Marching Cubes algorithm. The reason for using two different TSDF fusion implementation is explained in section 5.1.

4. Methodology

4.1. Overview

In this section, we present an overview of our employed methodology for the task of detailed 3D scene reconstruction. First, we introduce our notation in subsection 4.1.1. We briefly summarize the underlying assumptions and observations leading to a particular approach along with the investigated research questions, before proceeding to elaborating the proposed methods with the aim of presenting a self-contained methodology section.

4.1.1. Notation

We use bold capital letters to denote vector-valued images, such as RGB images, with pixel coordinates in parenthesis, e.g., $\boldsymbol{I}(u, v)$. Scalar-valued images, e.g., depth maps, are written with capital letters $D(u, v)$. We use bold upper case letters for matrices, e.g., $\boldsymbol{V}$ and bold lower case letters for vectors, e.g., $\boldsymbol{v}$. To denote scalars, lower case, e.g, t or upper case letters, e.g., N are employed. We exploit subscripts for enumeration e.g., s_i denotes TSDF value at the i^{th} voxel. We use superscripts to provide additional information to the reader for clarity. For instance, $\boldsymbol{S}^l$ represents the TSDF volume at level l, while $\boldsymbol{S}^g$ is used as a notation for *global* TSDF volume. If a notation deviates from the described ones here, we specifically define it.

4.1.2. Research Questions & Proposed Methods

Referring back to the motivation behind our proposed methodology, we intend to accurately reconstruct fine-scale details along with the coarse structures within a given 3D scene. With this purpose in mind, we make a number of assumptions and observations regarding the factors affecting the preservation of details, and then investigate these factors. Each approach within this work targets a specific research question. The proposed methods are independent from each other, unless their relation is explicitly stated. Below, an overview of the assumptions and corresponding proposed approaches are listed:

1. **TSDF & Occupancy Supervision:** The supervisory signal based on target occupancy and target TSDF values may be insufficient or overcomplicated for obtaining detailed reconstructions. Different auxiliary loss terms can be derived exploiting the mathematical & structural properties of TSDFs to obtain more consistent and accurate predictions. In an opposite direction the supervisory signal can be simplified to enhance the supervision.

a) **Gradient-based Supervision:** Does employing an auxiliary loss term based on the Eikonal property or directly gradient magnitudes, provide predicting more consistent TSDFs?

b) **TSDF Generation from Meshes:** Do we obtain more accurate target TSDFs when the TSDFs are approximated from the target mesh instead of the fusion algorithm?

c) **Simple Occupancy Probability Supervision:** Is solely deciding on the occupancy, an easier task for the network rather than also reasoning about the distance values?

2. **Surface Geometry Supervision:** Using distance and occupancy values may be insufficient for predicting the surface boundaries, especially if the structures are small or if they involve high-level details. The network tends to omit these finer details as they are enclosed in a few voxels, and thus contribute insignificantly to the overall loss computation.

   a) **Sign Enforcing Loss Term:** Does employing an auxiliary loss function that specifically enforces a TSDF sign change at the zero-crossing, bring back the lost surface details? Does assigning higher weights to voxels representing these finer details, improve the quality of reconstructions?

   b) **Point-level Supervision using Surface Meshes:** Can we obtain details by introducing a loss term that minimizes the distance between the surface samples of the ground-truth and the predicted meshes?

3. **Depth-based Supervision:** Rather than using the actual sensor measurements such as depth maps, using processed representations such as TSDFs or surface meshes for supervision, may introduce flaws to the pipeline owing to possible implementation errors/approximations in the algorithms that they are generated with. Measurements from sensors can be used to reduce the gap between the representations used for supervision and evaluation.

   a) **Occupancy Detection using Depth Maps**: Can we enhance the robustness of the voxel sparsification step by using depth maps as an auxiliary occupancy detection tool?

   b) **Point-level Supervision using Depth Maps**: Does using ground-truth depth maps for minimizing the distance between the surface samples of the predicted meshes and back-projected depth maps, provide a better surface geometry supervision?

   c) **TSDF Supervision using Depth Maps**: Can we obtain more accurate results by eliminating the TSDF fusion process from the pipeline and estimating target TSDF values on-the-fly using depth maps?

4. **Resolution:** The resolution of the voxel-grid is a well-known factor that limits the reconstruction quality.

   a) **4-Level Architecture:** To what extent, increasing the resolution of the finest level by integrating a fourth level to the architecture, affects the quality of reconstructions?

b) **Neural Upsampling:** Can neural upsampling be simply employed to increase the resolution of the finest level?

The overview of our mesh-based and depth-based point-level surface geometry supervision approaches are provided in Figure 4.1.

In our implementations, we employ the base architecture described in section 3.3 as our code basis. As stated previously, our base architecture closely follows [12], except the minor modifications applied within its 2D and 3D networks for efficiency purposes. We use the implementation details of *NeuralRecon*, as our default configuration. This default setup is provided in Table 4.1. When the proposed methods require a different architecture or a different parameter configuration, we explicitly highlight the deviations from the default setting.

Voxel size of the finest level:	$4cm$
Grid dimensions:	$96 \times 96 \times 96$
Truncation distance:	$\rho = 12cm$
t_{max}, R_{max}:	$0.1m$, 15°
Sparsification threshold:	$\theta = 0.5$
Interpolation for upsampling:	Nearest-neighbor interpolation
Maximum depth:	$d_{max} = 3m$
Finest level image feature map dimensions:	$160 \times 120 \times 24$
MLP channels:	$[96, 48, 24]$

Table 4.1.: **Default configuration.**

4.2. TSDF & Occupancy Supervision

4.2.1. Gradient-based Supervision

Motivation

Inspired by the Eikonal partial differential equation, [67] employs an *Eikonal term* in their loss function to implicitly regularize the zero level set and enforce the network to produce a valid SDF field that generates an accurate zero level set surface. The viscous solution to this first-order non-linear partial differential wave equation is unique, if it satisfies the given boundary conditions [13, 67]:

$$f(\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \partial\Omega \tag{4.1}$$

where $\Omega \in \mathbb{R}^3$ is a well-behaved open set with smooth boundary $\partial\Omega$. Then, the solution to the Eikonal equation is a signed distance function f that vanishes on $\partial\Omega$. As given in Equation 4.2, gradients of f have unit magnitude.

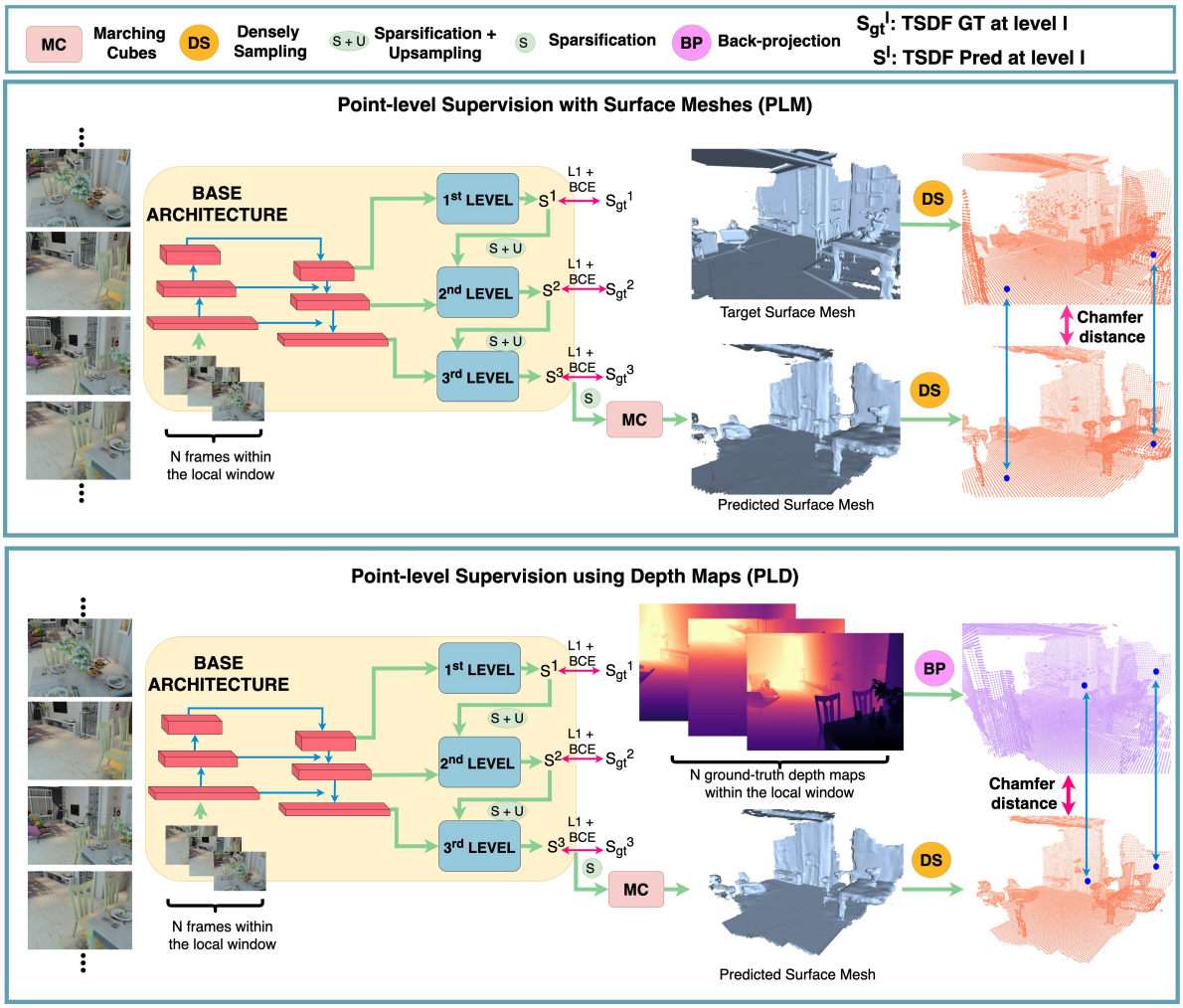

Figure 4.1.: **Overview of the mesh-based and depth-based point-level approaches.** By minimizing the distance between surface samples of the predicted mesh and either back-projected ground-truth depth maps or surface samples of the ground-truth mesh, we explicitly supervise the surface geometry to preserve fine-scale details. **PLM:** The predicted TSDF from the finest level is passed to Marching Cubes algorithm to extract the surface mesh. Both the target and predicted meshes are densely sampled to provide adequate surface samples for the Chamfer distance (CD) minimization. **PLD:** Ground-truth depth maps within a local window are back-projected onto the 3D TSDF grid to minimize the CD between the back-projected depth maps and densely sampled surface points. Distances are fed into the Cauchy function prior to overall CD computation to mitigate the effect of outlier points to the supervision.

$$\|\nabla_x f(\boldsymbol{x})\|_2 = 1 \tag{4.2}$$

However, [67] uses point cloud data to compute the SDF. Therefore, the boundary condition is violated owing to being discrete, and the above-mentioned solution is not unique. However, the optimization of the proposed loss function is still able to reach a good local minima [67]. *SDFDiff* also exploits Eikonal property [77], but uses it as an auxiliary normalization term which is an addition to the main image-based loss. Since *SDFDiff* is a voxel-based method, the gradients are computed using a finite difference approximation. An additional Laplacian loss is also used as a geometry loss.

Motivated by prior work, we explore to what extent the Eikonal property can be adapted to voxelized TSDF fields and whether using gradients as a supervisory signal directly is advantageous towards predicting accurate TSDF volumes.

Approach

To investigate whether the Eikonal property is fulfilled on our ground-truth TSDFs obtained from the fusion algorithm [15], we use central finite difference approximation to compute the gradients. Approximated gradients in x, y and z axes, are used to calculate the per-voxel gradient magnitudes. As the gradients of truncated voxels are not unit norm, they should not be included in gradient computation. Thus, we eliminate truncated voxels and their neighbors prior to finite difference method, which naturally enforces us to evaluate for valid voxels around the zero-crossings.

Figure 4.2 illustrates the TSDF gradient magnitudes of different InteriorNet scenes and their distributions. Although the peaks of the histograms are around the magnitude 1, the distributions show high variances. Considering the fact that the TSDF fusion algorithm computes the closest distance to the surface along the ray direction, it is not a real Euclidean Signed Distance Field (ESDF) in which each free voxel stores the Euclidean distance to the nearest occupied voxel [78]. Thus, owing to all the approximations involved in the process, it is possible that the obtained ground-truth TSDF fields from the TSDF fusion algorithm have inconsistencies, and thus cannot satisfy the Eikonal property in practice.

While our ground-truth TSDFs do not fulfil the Eikonal property, the approximated gradients in each axis can still be beneficial as an auxiliary supervisory signal. Thus, using the computed gradients of non-truncated voxels, we define a gradient-based loss term:

$$L^l_{grad} = \frac{1}{n}\sum_i L_1(\partial_x s_i, \partial_x s^*_i) + L_1(\partial_y s_i, \partial_y s^*_i) + L_1(\partial_z s_i, \partial_z s^*_i) \tag{4.3}$$

where s and s^*_i are predicted and ground-truth TSDF values at the i^{th} voxel, and n denotes the number of all valid voxels. The loss aims to minimize the L1 distance between the ground-truth and approximated gradients in x, y, z axes separately. The loss is only computed in regions of the observed target TSDF and incorporated to the total loss L^l_{tot} with weight λ_{grad}.

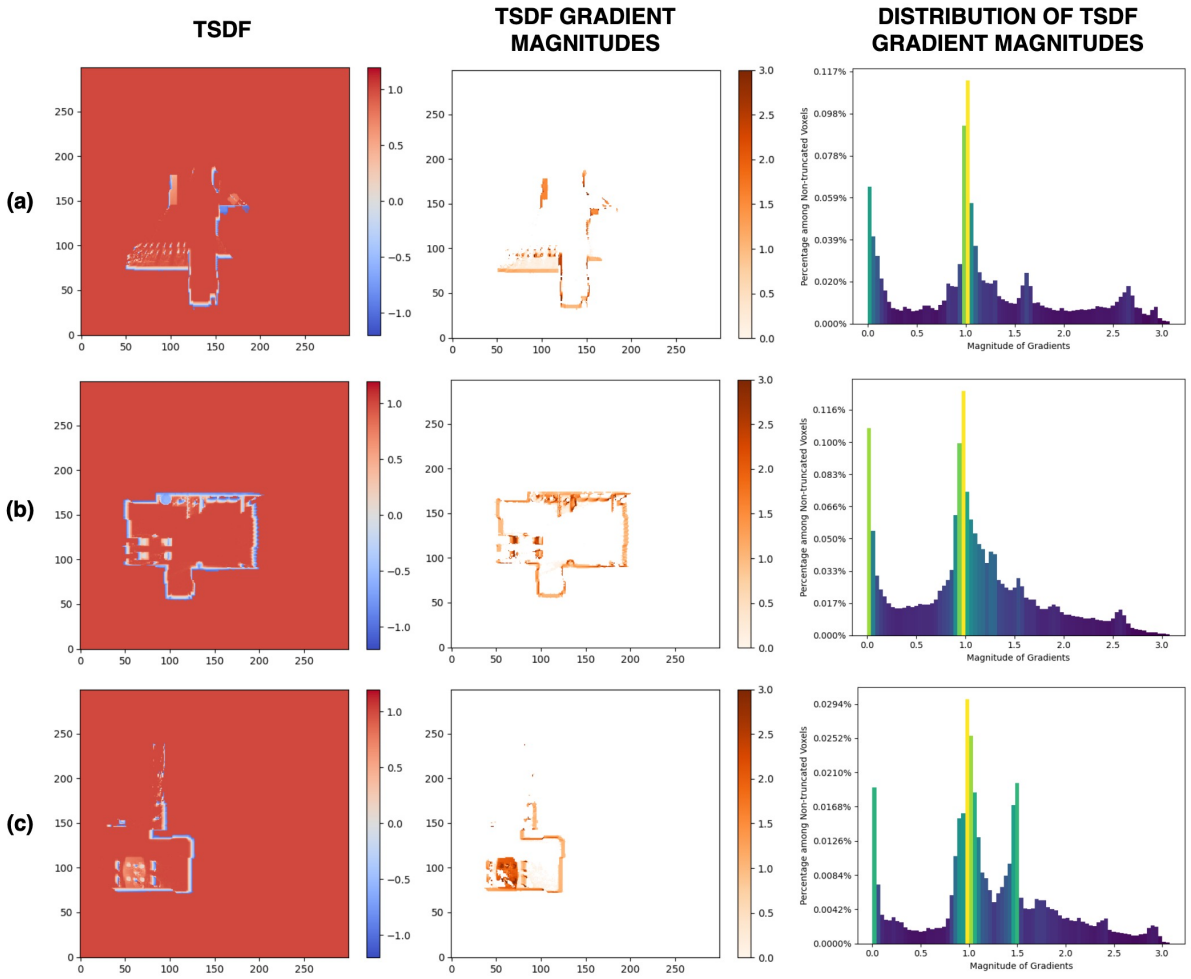

Figure 4.2.: **TSDF gradient magnitude distributions of Interiornet scenes.** Each row represents a single scene with three images, left one is a slice from the TSDF, middle one is the corresponding slice giving gradient magnitudes, right one is the histogram demonstrating the distribution of gradient magnitudes in the whole volume. It can be observed that gradient magnitude histograms show high variances and have the peak values around the magnitude 1. Therefore, we conclude that the Eikonal property is not well enforced in our ground-truth TSDFs obtained from the fusion algorithm, but the gradient magnitudes can still be exploited for a gradient-based supervision.

4.2.2. TSDF Generation from Meshes

Motivation

We reconsider alternatives towards generating a consistent TSDF representation that fulfils the Eikonal property, to be used as ground-truth. Considering a SDF can be computed from a watertight mesh by directly finding the closest triangle on the mesh for a given spatial point, we aim to generate our ground-truth TSDFs using the ground-truth meshes.

To this end, we benefit from the successful approach of *DeepSDF* [62], which introduces a SDF computation procedure from meshes, as a data pre-processing step. Ideally, in the presence of watertight meshes, we expect to generate accurate TSDF representations which are able to produce the same input mesh when Marching Cubes is applied. However, in our case, we aim to reconstruct complex scenes whose meshes are not watertight owing to involving many internal objects and sometimes holes. Also, note that, since ground-truth meshes are not supplied within InteriorNet data, we generate them using the Marching Cubes algorithm on the TSDFs obtained by running TSDF fusion, as described in subsection 3.4.3. Thus, our ground-truth meshes are also inherently prone to possible errors introduced by these two algorithms. Still, to have a better intuition towards consistent ground-truth TSDF generation, we adopt the strategy presented in [62].

Approach

To obtain the TSDF value of each voxel, coordinates of voxels are extracted as query points. Following [62], we then normalize each-ground truth mesh into a unit sphere while leaving some margin. *DeepSDF* proposes two different approaches towards working on watertight and non-watertight meshes. We employ both of the proposed approaches by [62] to generate our ground-truth TSDFs.

In the absence of watertight meshes, *scanning technique* [62] can be used. Using equally spaced virtual cameras located on the unit sphere, the mesh can be rendered and obtained depth maps via rendering can be back-projected onto the surface of the mesh [62] to generate a surface point cloud. Normals of these points are decided by the surface triangle which they are back-projected onto. Using a KD-tree, the closest point from the surface point cloud to each query point can be found. Later, the distance is computed, and then sign is determined using the normals. If several closest points are used for more robust sign determination, majority voting is applied. When watertight meshes are available with accurate face normals, *sampling technique* [62] can be used to sample random points from the surface. SDF computation is the same with the scanning method.

We set the scan resolution and scan count parameters to 640 pixels and 400 respectively. We use 20 closest points from the surface while deciding the sign, for both scanning and sampling methods. Considering the complexity of our scenes, we use 10M points for each scene when sampling technique is used. As the scale of the values in the obtained SDFs are different from the TSDFs from the fusion algorithm, the right scale factor is found and applied on the SDFs. Then, scaled SDFs are truncated at $12cm$ as in *NeuralRecon* to obtain the TSDFs. Central finite difference approximation is used to compute the gradients of SDFs and valid gradients of TSDFs, as in subsection 4.2.1.

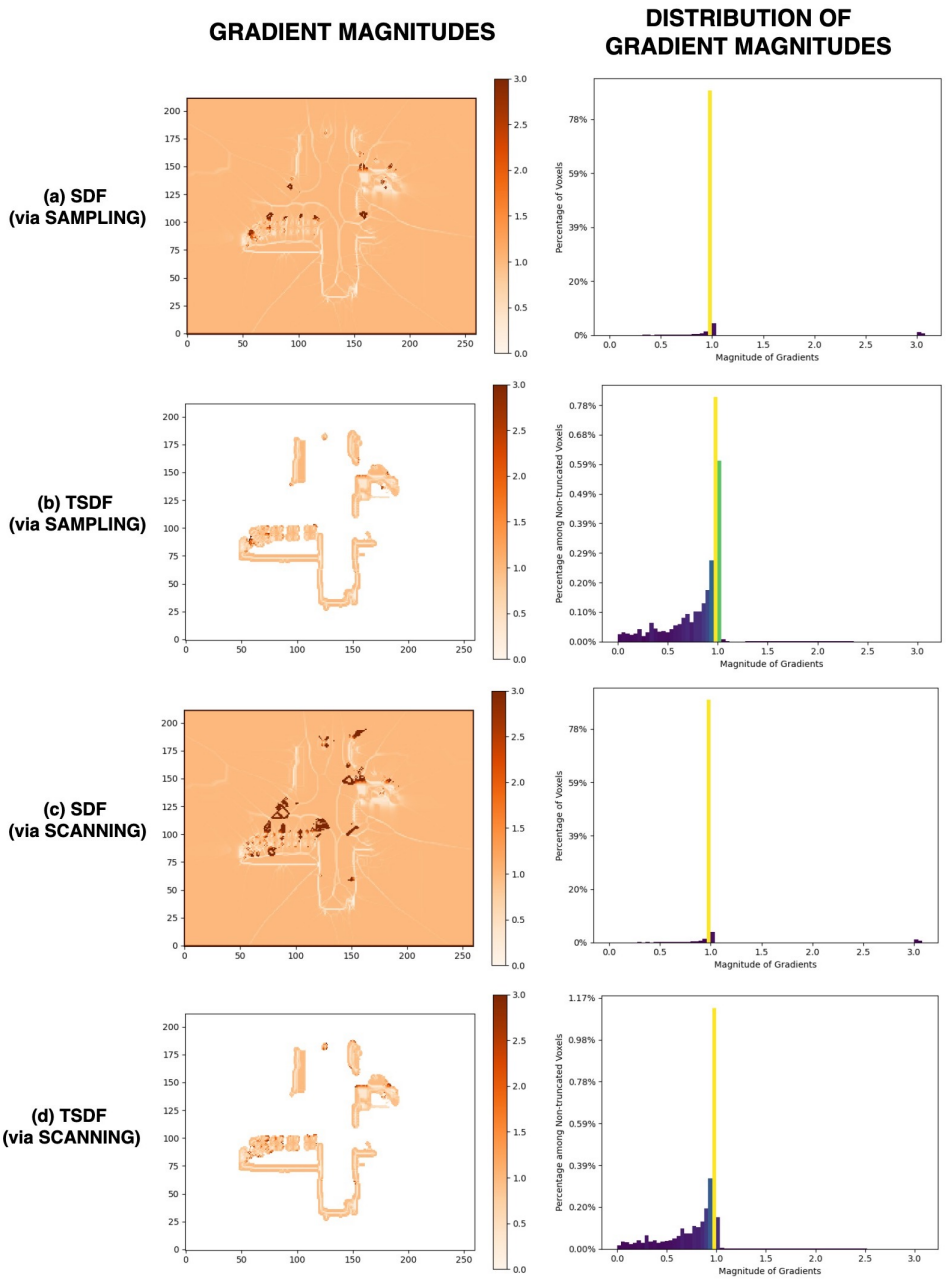

Figure 4.3.: **Gradient magnitudes and gradient magnitude distributions of SDFs & TSDFs obtained by scanning & sampling techniques.** All of the images are from the same InteriorNet scene where gradient magnitudes are visualized with a single slice and histograms demonstrate the distributions in the overall SDFs & TSDFs. Although the SDFs from scanning and sampling involve some observable artifacts, their gradients fulfil the Eikonal property for both of the techniques. TSDFs have gradient magnitude distributions around 1, with some variance.

For both of the techniques, Figure 4.3 presents the gradient magnitudes and gradient magnitude distributions of SDFs and their truncated versions. Despite the minor artifacts observed on SDF gradients, they still fulfil the Eikonal property by having unit magnitude gradients almost all around the voxel volume, except possibly where singularities and artifacts are observed. However, Eikonal property again is not well enforced on the TSDFs. Although, the gradient magnitude distribution histograms of TSDFs have the peak at 1, the distributions have a wide range of values. When compared with our default ground-truth, however, this variance is less significant. When SDFs and TSDFs are considered directly, it is noticed that positive values are assigned incorrectly to some voxels that represent the inside of the objects. Considering the fact that we have more complex meshes with holes and inner structures, meshes may include triangles with incorrectly oriented normals which causes inaccurate results when normals are used for deciding the sign.

To investigate the effect of using TSDFs obtained from the surface meshes, we replace the TSDFs from the fusion algorithm with the generated ones. We employ the following individual approaches for both of the representations obtained via scanning and sampling:

- **Approach (1) - TSDF Supervision:** We employ the generated TSDFs for TSDF supervision, in order to observe the effects of using probably more faithful representations.
- **Approach (2) - Unsigned Distance Supervision:** As previously stated, both the scanning and sampling methods sometimes incorrectly estimate the sign of the distance values. To prevent this, only the absolute value of the estimated distances are employed and sign information is incorporated from the default TSDF representations. Therefore, each voxel is assigned the unsigned distance value estimated by either scanning or sampling, and the sign of the corresponding voxel estimated via the TSDF fusion algorithm. Then the TSDFs are supervised with these sign corrected TSDFs.

4.2.3. Simple Occupancy Probability Supervision

Motivation

By storing the signed distance to the closest surface point, TSDFs provide a straightforward representation towards understanding the surface topology. However, reasoning about the distance of a voxel to the zero-crossing while also deciding on which side of the zero-crossing that it takes place, can be considered as a much more difficult task than deciding whether this voxel is occupied or not. To this end, [61] attempts to assign occupancy probabilities to every point in 3D space. At inference time, a voxel is considered as occupied if its predicted probability is larger than a predefined threshold. Similarly, many robotic applications successfully employ occupancy probabilities for many tasks such as volumetric mapping [2, 3]. Motivated by these works, we investigate whether representing a 3D scene using an occupancy grid is a relatively easier task to accomplish for our network which results in estimating more accurate surfaces.

Approach

To this end, the default model is modified to remove any TSDF related components. After each level, only the occupancy grid is upsampled to be concatenated with the extracted features of the next level. Also, the loss function is updated to only involve the BCE loss for occupancy supervision: $L^l_{tot} = L^l_{occ}$.

4.3. Surface Geometry Supervision

4.3.1. Sign Enforcing Loss Term

Motivation
Although L1 loss is commonly employed for reconstruction tasks, it may be smoothening the surface boundaries when the transition between the inside and the outside of a surface cannot be sufficiently represented by the underlying TSDF field. A possible scenario for this case is presented in Figure 4.4, with a ground-truth mesh and crops from the corresponding ground-truth TSDF volume. The mesh involves visible finer details such as chair legs, flowers in a vase etc., which are represented by small volumes with negative distance values (in blue) in the corresponding TSDFs. These values are relatively small in magnitude as the corresponding voxels are not far away from the zero-crossing. Hence, the supervisory signal required for learning such kind of structures becomes insufficient, considering that a few voxels representing these objects contribute insignificantly to the loss computation. As a result, fine details are prone to be omitted by the network and their corresponding voxels are mostly assigned positive values. The zero-crossing can never be predicted which hinders preserving the local structures. To enforce the prediction of these details, we propose several auxiliary loss functions in this section.

Approach
When fine details are ignored by the network, they are mostly predicted as empty space or some artifacts are produced in the corresponding voxels. To reconstruct the entire zero-crossings of the structures properly, we introduce the following five individual modifications to the loss functions which enforce sign changes at the boundaries. As previously introduced in section 3.3, let s_i be the predicted TSDF and s^*_i be the ground-truth TSDF values at the i^{th} voxel, with n representing the number of voxels.

- **Loss modification (1):** We define the first sign change enforcing auxiliary loss function as follows:

$$L^l_{aux} = \frac{1}{n} \sum_i \text{relu}(-s_i \cdot s^*_i) \tag{4.4}$$

  which penalizes if the sign of the prediction deviates from the ground-truth. If both the ground-truth and predicted TSDF values have the same sign, the zero cost is assigned.

- **Loss modification (2):** The second auxiliary loss function forces the prediction and the

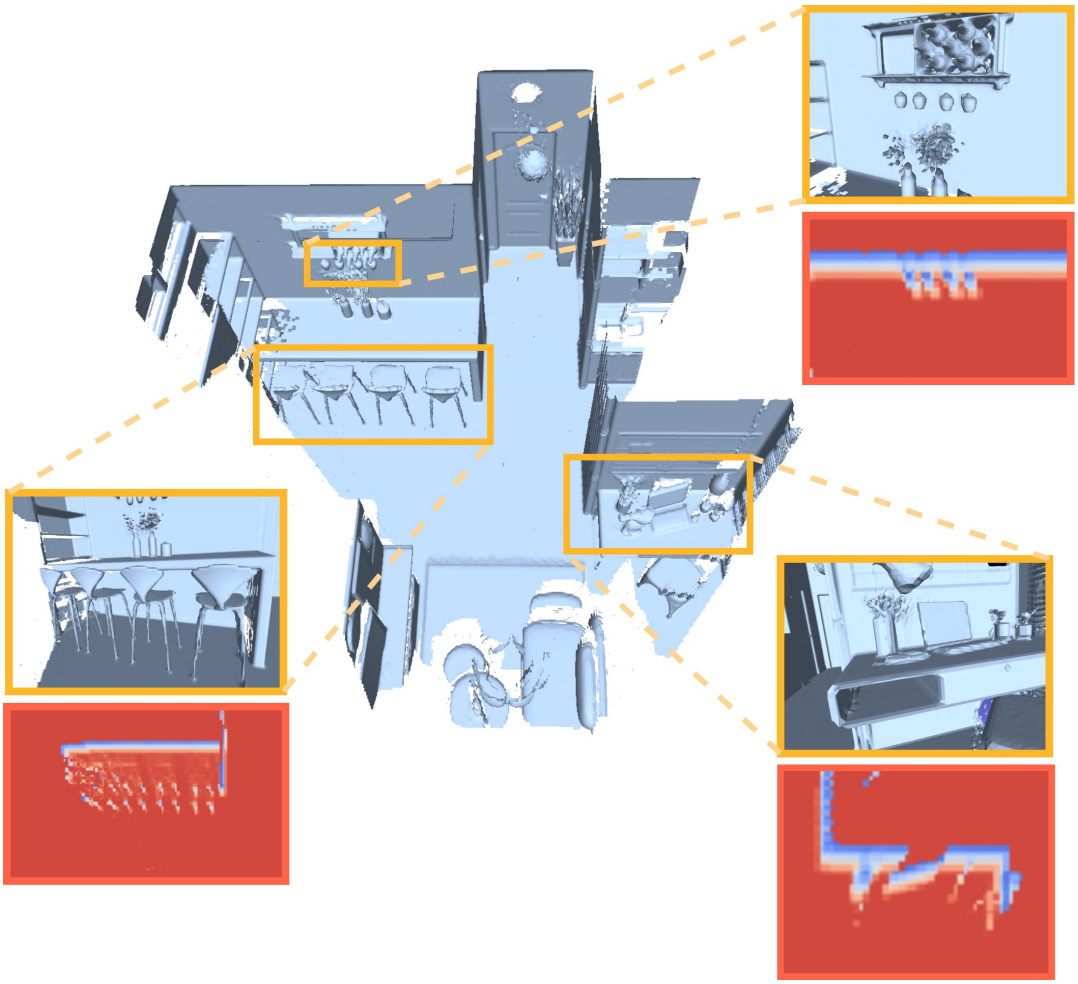

Figure 4.4.: **A sample ground-truth mesh and crops from its corresponding TSDF.** While the ground-truth surface mesh involves visible fine details such as chair legs, flowers in a vase etc., the underlying TSDF field cannot represent these details sufficiently as it can be observed from the TSDF crops. A few voxels enclosing these structures contribute insignificantly to the loss computation, and therefore fine details are prone to be omitted by the network. We address this problem with the proposed explicit surface geometry supervision approaches.

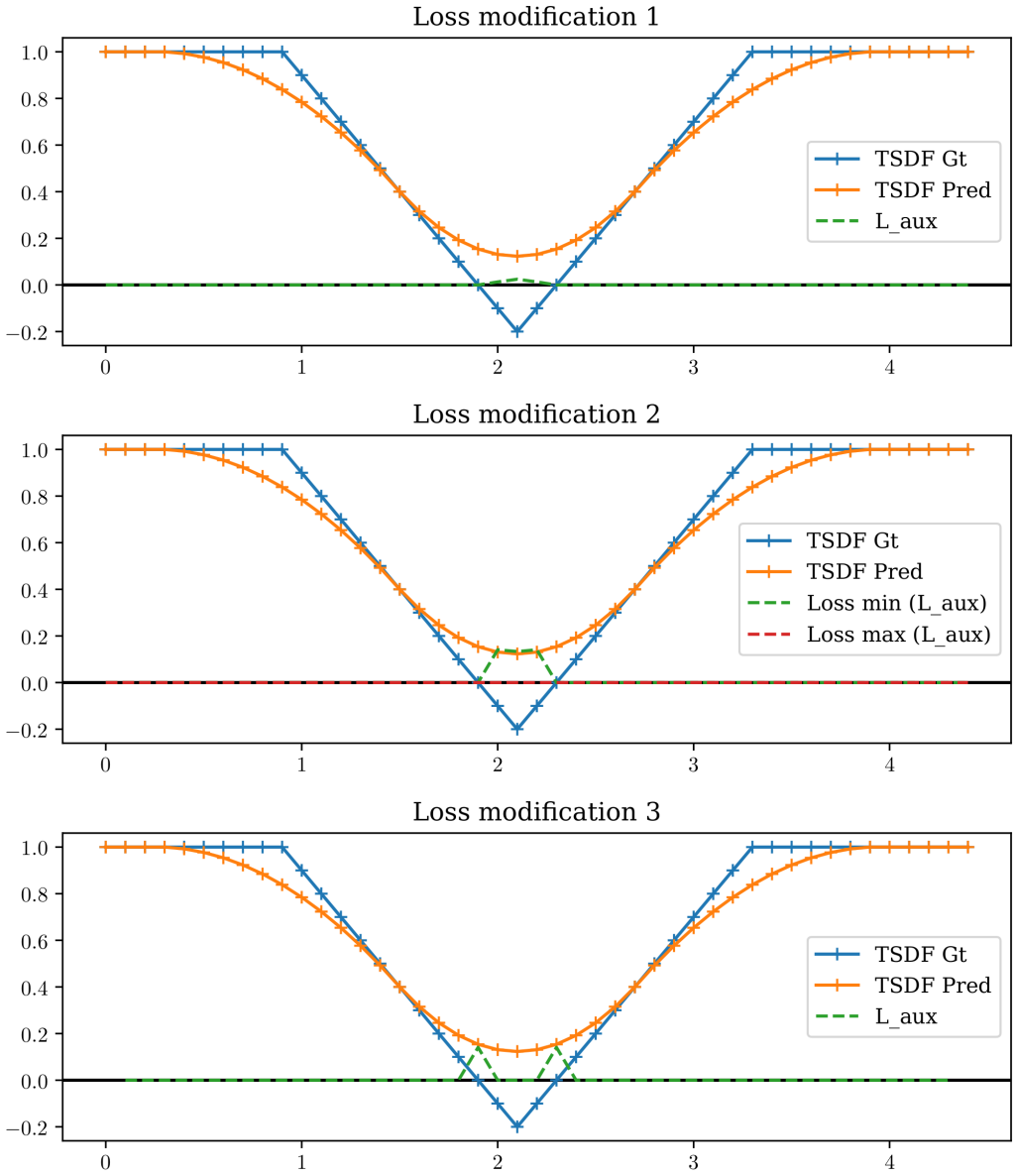

Figure 4.5.: **Behaviors of the introduced auxiliary loss functions in modifications 1-3.** The first and second loss functions penalize the regions where the signs of target and prediction do not match, while the third function penalizes the sign mismatches around the zero-crossing.

ground-truth to have the sign benefiting from L_{max} and L_{min} losses:

$$L_{max}(x) = \text{relu}(-x + \varepsilon) \tag{4.5}$$

$$L_{min}(x) = \text{relu}(x + \varepsilon) \tag{4.6}$$

Using these two loss functions, we define L_{aux} which conditionally enforces either L_{max} or L_{min} where ε is a small margin.

$$L^l_{aux} = \frac{1}{n}\sum_i L_{cond_i}(s_i, s^*_i) \text{ with } L_{cond_i}(s_i, s^*_i) = \begin{cases} L_{max}(s_i) & \text{if } s^*_i \geq \varepsilon \\ L_{min}(s_i) & \text{if } s^*_i \leqslant -\varepsilon \\ 0 & \text{else} \end{cases} \tag{4.7}$$

- **Loss modification (3):** We use pooling to find consecutive cells where the minimum value has a negative sign and the maximum value has a positive sign. Then we label these cells to have a sign change. We compute a loss for each cell to enforce the prediction at the location giving the minimum value to have a negative sign while the prediction at the location giving the maximum value to have a positive sign.
- **Loss modification (4):** We also consider giving higher weights to *hard samples*, which are the voxels that our network has difficulty in regressing the TSDF value for. Owing to the aforementioned reasons, these voxels are mostly the ones near the zero-crossings. Therefore, we define per-voxel weights $\lambda_{tsdf_i} = \frac{1}{|s^*_i| + \varepsilon}$ with $\varepsilon > 0$, to weight each voxel inversely proportional with its ground-truth absolute distance to the zero-crossing.
- **Loss modification (5):** With the same motivation, we also consider assigning a higher weight to voxels that store negative ground-truth TSDF values. Thus, we propose using two different weights λ_{tsdf_+} and λ_{tsdf_-} for voxels with positive and negative ground-truth TSDF values respectively.

As previously stated, these are proposed as individual modifications to the default loss function L^l_{tot}. A selected auxiliary loss L^l_{aux} (loss modifications 1-3) is incorporated into the total loss function L^l_{tot} with weight λ_{aux}. Figure 4.5 presents behaviors of the introduced loss functions, where ground-truth and predicted TSDFs values are generated for illustration purposes.

4.3.2. Point-level Supervision using Surface Meshes

Motivation

Though TSDFs are employed as intermediate representations by most of the voxel-based MVS approaches, the final evaluations are performed using 3D metrics on the extracted surface mesh and 2D metrics on the rendered depth maps from this predicted mesh [11, 12]. Considering this fact, we investigate whether supervising our network using the ground-truth surface meshes enhances the quality of the reconstructions.

One major complication of using the ground-truth surface as a supervisory signal is to make the whole pipeline work in an end-to-end fashion. Since Marching Cube algorithm is not differentiable with respect to topological changes owing to the linear interpolation it involves, it is not suitable for end-to-end training. Prior work proposes several approaches towards obtaining a differential Marching cubes algorithm [57, 59, 58, 60]. Inspired by *MeshSDF* [58], which discusses how an infinitesimal perturbation on a SDF field affects geometry of the local surface, we follow a similar methodology to make our pipeline suitable for backpropagation.

Let $\boldsymbol{\mathcal{M}} = (\boldsymbol{V}, \boldsymbol{F})$ bet the surface mesh with vertex positions $\boldsymbol{V} = (v_1, v_2, ...)$ in $\mathbb{R}^3$ and F faces, *MeshSDF* defines L_{task} as the loss function to minimize on this mesh. Considering f_θ as the MLP predicting the SDF with network parameters θ, a latent code $\boldsymbol{z} \in \mathbb{R}^Z$ which encodes the surface and a 3D point are provided as inputs to f_θ. Then, the chain rule to be evaluated during backpropagation becomes:

$$\frac{\partial L_{task}}{\partial \boldsymbol{z}} = \sum_{v \in V} \frac{\partial L_{task}}{\partial \boldsymbol{v}} \frac{\partial \boldsymbol{v}}{\partial f_\theta} \frac{\partial f_\theta}{\partial \boldsymbol{z}} \tag{4.8}$$

in which only the second term, representing the gradients of mesh vertices with respect to the underlying SDF field, is not differentiable due to Marching Cube algorithm it involves. However, *MeshSDF* approximates this term by the inverse surface normal and plugs into Equation 4.8 directly [58].

$$\frac{\partial \boldsymbol{v}}{\partial f_\theta}(\boldsymbol{v}) = -\boldsymbol{n}(\boldsymbol{v}) = \nabla f_\theta(\boldsymbol{v}) \tag{4.9}$$

Approach

We supervise our network using the ground-truth meshes only at the finest level. All the three levels still benefit from TSDF and occupancy supervisions. The employed forward and backward passes are provided below.

Forward pass: We keep our architecture the same by predicting TSDFs at each level, and then calculating occupancy and TSDF losses. From the per-fragment TSDF predicted by the finest level, we extract the surface mesh $\boldsymbol{\mathcal{M}} = (\boldsymbol{V}, \boldsymbol{F})$ via the Marching Cubes algorithm. In order to create sufficient surface samples for the supervision, we sample both the predicted and ground-truth meshes densely rather than using their vertices. We employ the Chamfer distance given in Equation 4.10 as our mesh-based loss, where $\boldsymbol{P}$ and $\boldsymbol{P^*}$ are the point clouds that consist of sampled surface points from the predicted and ground-truth meshes respectively.

$$L_{CD} = \frac{1}{|\boldsymbol{P}|} \sum_{\boldsymbol{p} \in \boldsymbol{P}} \min_{\boldsymbol{p}^* \in \boldsymbol{P}^*} \|\boldsymbol{p} - \boldsymbol{p}^*\|_2^2 + \frac{1}{|\boldsymbol{P}^*|} \sum_{\boldsymbol{p}^* \in \boldsymbol{P}^*} \min_{\boldsymbol{p} \in \boldsymbol{P}} \|\boldsymbol{p} - \boldsymbol{p}^*\|_2^2 \tag{4.10}$$

As the distances are penalized quadratically by the Chamfer distance, hallucinated surface samples or missing samples in the predictions create noise to the supervision. To alleviate this issue, we employ the provided Cauchy function and input the distances to this function before computing the overall Chamfer distance:

$$\phi(d, \alpha) = \frac{\alpha^2}{2} \log(1 + (\frac{d}{\alpha})^2) \tag{4.11}$$

where $\alpha \in \mathbb{R}$ is the Cauchy cut-off parameter and d represents the distance between $\boldsymbol{p}^* \in \boldsymbol{P}^*$ and $\boldsymbol{p} \in \boldsymbol{P}$. This formulation enforces a decreasing gradient distribution when the cut-off is exceeded. Thus, when the computed distance between the sampled points is larger than the cut-off, their contribution to the supervision decreases gradually as the computed distance diverges from the cut-off.

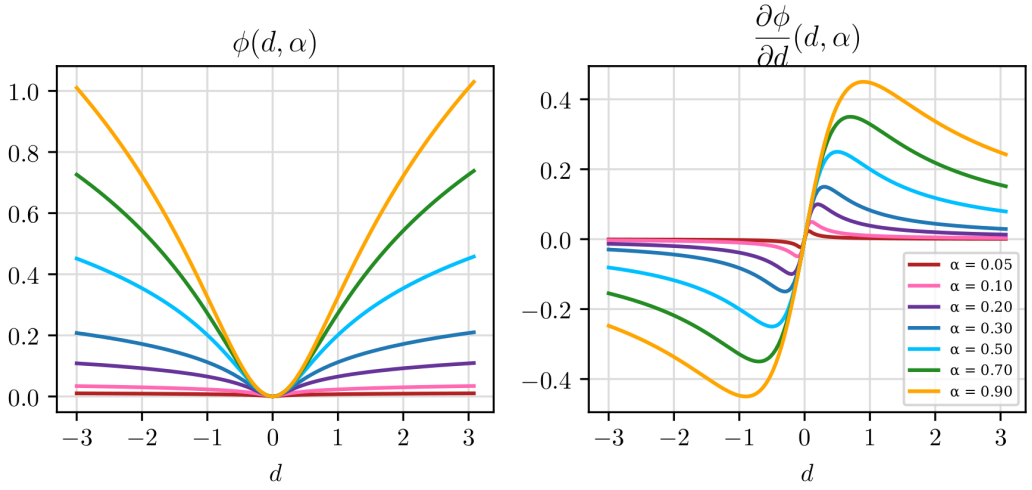

Figure 4.6.: **Cauchy function and its gradient.** When the computed distance between the sampled points diverges from the Cauchy cut-off parameter α, their gradients decrease gradually and contribute less significantly to the supervision. This way, the outliers are avoided.

Although TSDF and occupancy losses act as regularizers that prevent predicting non-smooth meshes when the Chamfer distance is minimized only, we still incorporate several shape regularizers to our loss function.

- Normal consistency loss, L_{NC}, is employed to ensure consistency across the normals of neighboring faces.
- Mesh edge length regularization loss, L_{edge}, is employed with the purpose of minimizing edge lengths of the predicted surface.
- Laplacian smoothing regularizer, L_{Lap}, with uniform weights is used.

The final mesh-based loss has the form given in Equation 4.12 and incorporated to L_{tot}^2 with a weight λ_{mesh}.

$$L_{mesh} = \lambda_{CD} L_{CD} + \lambda_{NC} L_{NC} + \lambda_{edge} L_{edge} + + \lambda_{Lap} L_{Lap} \tag{4.12}$$

Backward pass: The steps of the adapted backward pass from [58] are listed in the following:

1. Having calculated the loss, we compute the upstream gradients $\frac{\partial L_{tot}}{\partial \boldsymbol{p}}$ for each $\boldsymbol{p}$.
2. To compute the inverse surface normals, an additional forward pass is required using the sampled surface points [58]. This forward pass is computed by directly inputting the surface sample and the latent code to the MLP in *MeshSDF*. However, since our architecture is different from *MeshSDF*, we compute the forward pass by performing a trilinear interpolation on the predicted TSDF using the samples.
3. Surface normals $\boldsymbol{n}(\boldsymbol{p})$ are computed for each $\boldsymbol{p}$.
4. *MeshSDF* aims to update the latent codes by using the gradients. Unlike *MeshSDF*, we update different parameters, and thus we compute the gradients of MLP with respect to all these parameters.

4.4. Depth-related Supervision

4.4.1. Occupancy Detection using Depth Maps

Motivation
In the base architecture, to efficiently process the TSDF volume incrementally, the volume is sparsified after each level of the coarse-to-fine hierarchy, if the predicted occupancy probabilities of voxels are not within the truncation distance ρ. Unfortunately, if a voxel is incorrectly sparsified in one of the coarser levels, it is not possible to recover that voxel in the finer levels and predict its TSDF value. This makes the architecture prone to losing details. To overcome this problem, we propose using depth maps to avoid incorrect sparsification of voxels.

Approach
In this regard, we back-project depth maps of N frames within the local fragment window into the voxel-volume using Equation 3.6 to obtain a 3D point cloud. If at least one of the points within this point cloud falls into a particular voxel, we mark this voxel as *occupied*. Marked voxels are not sparsified even if their predicted occupancy probabilities are not within the truncation distance. With this approach, although we do not prevent hallucinating structures in the voxels that are incorrectly predicted as occupied, we avoid inaccurately sparsifying voxels where there is an actual structure. Therefore, occupancy information can be propagated from coarser levels to finer levels accurately.

4.4.2. Point-level Supervision using Depth Maps

Motivation
As previously stated, real-world datasets are prone to errors caused by sensors and also human interference to the process, thus they are limited in terms of quality when compared to synthetic ones. In this work, the reason behind using a synthetic dataset, InteriorNet, is

having a mostly perfect ground-truth data which already presents a great amount of details and thus can be used for the task of detailed 3D reconstruction. When real-world data is used, it may be difficult to assess the reconstruction performance as their ground-truth may be already lack of details.

Nevertheless, since we extract the ground-truth surface meshes via the Marching Cubes algorithm and obtain the resolution-limited ground-truth TSDFs from the TSDF fusion, our ground-truth meshes are prone to possible errors caused by the approximations within these algorithms. This situation mitigates the benefits of employing a highly accurate ground-truth data from InteriorNet. Therefore, target meshes used for Chamfer distance calculation in subsection 4.3.2 may be also introducing noise to the supervision. To avoid these problems, we propose removing the need for ground-truth meshes by directly using depth maps for point-level surface geometry supervision.

Approach
We employ the same approach used in subsection 4.3.2 with the introduced forward and back passes. However, instead of using the surface points densely sampled from the target mesh for supervision, we employ the point clouds obtained from the back-projection of depth maps as our ground-truth points. During point cloud generation, we back-project the depth maps of N views within the local fragment window into the 3D volume.

4.4.3. TSDF Supervision using Depth Maps

Motivation
Although we propose improving the accuracy of point-level supervision using the ground-truth depth maps in subsection 4.4.2, TSDF supervision is still prone to errors due to ground-truth TSDFs obtained from the fusion algorithm. Therefore, we propose eliminating the TSDF fusion completely from the pipeline and supervising the TSDF values in image space using depth maps. By employing this approach, the training can be started immediately without the need of an additional ground-truth generation step.

Approach
In the base architecture, voxels of the 3D TSDF grid are represented using voxel coordinates. These voxel coordinates can be first transformed to real-world coordinates, and then can be transformed to coordinates in the camera coordinate system using Equation 3.6. Also, these real-world coordinates can be projected onto our depth maps. Considering that the distance between a depth value and a *z-coordinate* in the camera coordinate system gives the signed distance value for that voxel, we can compute this value on-the-fly for each voxel as the ground-truth TSDF value. With this approach TSDFs can be supervised without the need of an actual TSDF fusion algorithm. In the following, we introduce the steps of the proposed approach:

1. We convert voxel coordinates to real-world coordinates in the world coordinate system.

2. We transform center real-world coordinates of each voxel to the coordinates within the camera coordinate system using the extrinsic parameters. We store the obtained *z-coordinates* for each voxel.

3. We project center real-world coordinates of each voxel onto the N depth maps within the current local fragment window using Equation 3.6.

4. For each voxel, we first compute N TSDF values, and then we calculate the mean L1 loss for that voxel using the N values. The steps are as follows:

   a) For each of the N image coordinates obtained by projecting the voxel onto the depth maps, we use nearest-neighbor interpolation to obtain the corresponding depth value. Therefore, we obtain N depth values per-voxel.

   b) For each of the N depth values, we compute the distance between the depth value and the previously stored *z-coordinate* in the camera coordinate system. If the distance is larger than the truncation distance ρ, we do not use it for TSDF supervision. We accept the M valid values within the truncation distance, as the ground-truth TSDF values for this voxel. If $M > 0$, we represent the ground-truth occupancy of that voxel with a probability value of 1 and if otherwise, we use the probability value of 0.

   c) For each of the M ground-truth TSDF values, we compute the L1 distance to the predicted TSDF value for that voxel. Then, we estimate the mean of L1 distances. This averaged L1 distance becomes the final TSDF loss from that voxel. We also calculate the BCE loss between the predicted and ground-truth occupancy value for occupancy supervision.

5. We assign different weights to each voxel for TSDF supervision, considering from how many views this voxel can be seen. We use all the views available for a scene rather than using only the N views within the local window. Then, we compute the final TSDF loss by summing up the L1 losses from each voxel and normalizing with the weights. We also compute the mean BCE loss.

4.5. Resolution

4.5.1. 4-Level Architecture

Motivation

Using finer resolutions is a significant factor towards preserving high-level details, and thus faithfully representing the scene. We investigate, to what extent, increasing the resolution of the finest level affects the quality of our reconstructions in terms of preserving the details. To this end, we insert a final fourth level with a voxel size of 2*cm* to the existing 3-level coarse-to-fine hierarchy.

Approach

The extended architecture has a 2*cm* resolution at the finest level and 16*cm* resolution at the coarsest level. The architecture of the first three levels are kept the same and only additional layers are added to support the fourth level. The following modifications are applied to the default configuration:

- To represent the same region with a voxel size of 2*cm* that is previously represented with a voxel size of 4*cm* and grid dimensions of $(96 \times 96 \times 96)$, we increase the grid dimensions to $(192 \times 192 \times 192)$.
- Additional layers are added to both the 2D image backbone and 3D sparse convolutional module. After the modifications, image feature maps of the finest level has the dimension of $(320 \times 240 \times 16)$. MLP channels are modified to $[96, 48, 24, 12]$, to support the fourth level.

4.5.2. Neural Upsampling

Motivation
Inserting a fourth level with a finer resolution requires additional compute power together with a longer training time owing to the increased grid dimensions and inserted convolutional layers. To have a better intuition regarding the processing time of each level, we first benchmark individual operations of the default 3-level coarse-to-fine hierarchy using a single NVIDIA RTX A6000 GPU and provide the results in Table 4.2. As expected, coarsest level requires a longer processing time owing to its dense structure which is also densely processed, while processing time is reduced in the finer levels via sparse convolutions. Among the operations, GRU fusion dominates the other steps of the pipeline.

Operation	**Processing Time (*ms*)**
Level1 - Overall:	649.51
Level1 - Back-projection:	162.11
Level1 - Sparse convolutions:	105.50
Level1 - GRU fusion:	357.36
Level2 - Overall:	195.06
Level2 - Back-projection:	35.52
Level2 - Sparse convolutions:	63.08
Level2 - GRU fusion:	76.46
Level3 - Overall:	164.61
Level3 - Back-projection:	16.67
Level3 - Sparse convolutions:	70.49
Level3 - GRU fusion:	73.27

Table 4.2.: **Processing time of each level.**

To alleviate any compute problems that are intensified with the insertion of an additional level, we consider neural upsampling after the third level. This way, we can eliminate some of the compute demanding steps while increasing the final voxel resolution to 2cm.

Approach
To this end, after obtaining the sparsified TSDF volume from the third level, the upsampling via nearest-neighbor interpolation and back-projection steps are skipped. The sparsified volume is directly processed by a 3D convolutional network with fewer layers to obtain the required resolution. Then, the volume can be used as an input to the Marching Cubes algorithm to extract the 2cm resolution mesh. With this approach, back-projection and GRU fusion are completely eliminated from the fourth level and only the 3D convolutions are employed which take 59.12ms.

5. Experiments and Results

In this chapter, we introduce our experiments and deliver their results in order to perform an extensive evaluation of our methods. First, we explain the experimental methodology followed, and then we continue with presenting our results.

5.1. Experimental Methodology

In this work, we conduct two sets of experiments. First, we individually evaluate the performances of the proposed methods and selectively combine these methods into a final approach considering their validation set accuracy. Then, we perform experiments for comparing our final approach with the state-of-the-art. We present our results in this order, however the interested reader in our comparison with the state-of-the-art can skip to the section 5.4. Below, we delineate the experimental methodology followed:

1. **Experiments on Individual Methods:** The main goal of this first set of experiments, is to observe whether the proposed approaches individually contribute towards reconstructing detailed surfaces. We employ a single proposed method at a time, perform our training based on the introduced method, evaluate the obtained model and compare with the baseline. Therefore, this set of experiments can be also considered as *controlled experiments*. Considering the evaluation results of the individual methods on the validation set, we combine a few of them into a single final approach with the aim of observing to what extent we can improve the 3D reconstruction performance.

   Data: Considering the time required for training, we employ *InteriorNet-200* data, previously introduced in subsection 3.4.1. This smaller subset consists of 200 training and 50 validation scenes which are sampled from the corresponding splits of the main data. We train our models on the training set and evaluate our methods on the validation set.

   Resolution: Unless otherwise stated, we use ground-truth TSDFs with a $4cm$ voxel size at the last level for the training. Again, we employ ground-truth meshes extracted from a TSDF with a $4cm$ voxel size at the finest level for the metric evaluation on the validation set.

   Pre-trained model: Networks for 3D reconstruction tasks require longer training times owing to the amount of data, 2D & 3D convolution operations, cubic representations, convergence requirements etc. As it is not possible to train our network for each of the experiments from scratch, we employ a pre-trained model considering the time constraints. Unless otherwise stated in the related experiment sections,

the pre-trained model is the base architecture from *NeuralRecon* with the default configuration introduced in Table 4.1. It was trained on *InteriorNet-full* data with 500 training scenes and has a voxel size of 4*cm* at the last level. We used Adam optimizer with $\beta_1 = 0.9$ and $\beta_1 = 0.999$, and an initial learning rate of 1e-3 with a 0.5 decay at the epochs 12, 24, 48. We performed the training for 60 epochs. We refer to this model as *PreTrainedInt_4*.

General training details: For further training our models on *PreTrainedInt_4* using the proposed methods, we employed a learning rate of 1.25e-4 with a 0.5 decay at every 12 epochs. We used early stopping in our trainings and we employed the previously mentioned *InteriorNet-200* data for this part of the training.

Baseline: To evaluate whether the proposed methods contribute towards our final goal, we need a baseline that does not involve any modifications. With that purpose, we employed the introduced *PreTrainedInt_4*, trained it further on *InteriorNet-200* data without applying any modifications and used it as our baseline model. The reason for further training on a smaller subset of data, is to provide a fair comparison with the models employing one of the proposed methods. We refer to this baseline model as *baselineInt-4*.

General evaluation details: From the predicted TSDFs, we extract the surface meshes via the Marching Cubes (MC) algorithm and then compute the previously introduced metrics on the meshes. However, owing to the TSDF fusion implementation of *NeuralRecon* [12], the unobserved voxel space is filled with the positive truncation value 1. Therefore, MC algorithm considers this sudden transition from negative sign to positive sign as an implicit surface and produces double-layered meshes. However, other state-of-the-art approaches generate single-layered meshes and perform comparison on them. To make a fair comparison, we also report the results on single-layered meshes.

For this purpose, we first render depth maps from the double-layered predicted meshes. Then, we re-fuse these rendered depth maps into a TSDF volume using the TSDF fusion implementation from *Open3D* [76]. Re-fusing is also beneficial for removing filled holes as they are also penalized by 3D geometry metrics. Final single-layered predicted meshes are obtained by using *Open3D* version of MC. This approach is also used on ground-truth depth maps to obtain single-layered ground-truth surface meshes. 3D metrics, then, can be used to evaluate the predicted 3D geometry.

2. **Comparison with State-of-the-art:** After selecting the best performing approaches to be combined, we perform trainings on the full data to obtain our final models. We employ two datasets, namely InteriorNet and ScanNet, for the training and evaluation of our models. These, final models are used in comparison with state-of-the-art. We describe the experimentation and evaluation details in section 5.4.

We specifically highlight if our experiments require deviations from the introduced experimental methodology.

5.2. Evaluation Metrics

In order to evaluate 3D reconstruction quality, we use two sets of metrics: 1) standard 2D depth metrics defined in [79] and 2) 3D geometry metrics presented in [11].

5.2.1. 2D Depth Metrics

For a ground-truth depth map $D^*(u,v)$ and depth map $D(u,v)$ rendered from the predicted 3D mesh, we apply a depth threshold on both $D^*(u,v)$ and $D(u,v)$ to remove noise. Then, we define n as the number of pixels with both valid ground truth and predictions. $d_{(u,v)}$ is the depth value at the pixel (u,v) in the predicted depth map $D(u,v)$, while $d^*_{(u,v)}$ is the depth value at the pixel (u,v) in the ground-truth depth map $D^*(u,v)$.

We calculate the following error metrics:

Absolute Relative Difference

$$AbsRel = \frac{1}{n} \sum_{(u,v)} |d_{(u,v)} - d^*_{(u,v)}| / d^*_{(u,v)} \tag{5.1}$$

Absolute Difference

$$AbsDiff = \frac{1}{n} \sum_{(u,v)} |d_{(u,v)} - d^*_{(u,v)}| \tag{5.2}$$

Squared Relative Difference

$$SqRel = \frac{1}{n} \sum_{(u,v)} |d_{(u,v)} - d^*_{(u,v)}|^2 / d^*_{(u,v)} \tag{5.3}$$

Linear Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n} \sum_{(u,v)} |d_{(u,v)} - d^*_{(u,v)}|^2} \tag{5.4}$$

2D Completeness Returns the percentage of valid predictions.

5.2.2. 3D Geometry Metrics

We evaluate the 3D geometry of predicted meshes using *accuracy*, *3D completeness*, *precision*, *recall*, *F-score* and *Chamfer distance*. As stated previously, we re-fuse predicted (rendered from the extracted mesh) & ground-truth depth maps using *Open3D* into a TSDF volume and extract the surface meshes. We convert the single-layered meshes to point clouds for computing our metrics. Let $\boldsymbol{P}$ and $\boldsymbol{P}^*$ are the predicted and ground-truth point clouds, $\boldsymbol{p} \in \boldsymbol{P}$ and $\boldsymbol{p}^* \in \boldsymbol{P}^*$, we employ the following error metrics:

Accuracy We calculate the point cloud accuracy as the minimum distance from predicted points to the ground-truth points, as in

$$Acc = \text{mean}_{p \epsilon P}(\min_{p^* \epsilon P^*} \|p - p^*\|) \tag{5.5}$$

Completeness It is calculated as the minimum distance from the ground-truth points to the predicted points:

$$Comp = \text{mean}_{p^* \epsilon P^*}(\min_{p \epsilon P} \|p - p^*\|) \tag{5.6}$$

Precision

$$Prec = \text{mean}_{p \epsilon P}(\min_{p^* \epsilon P^*} \|p - p^*\| < .05) \tag{5.7}$$

Recall

$$Recall = \text{mean}_{p^* \epsilon P^*}(\min_{p \epsilon P} \|p - p^*\| < .05) \tag{5.8}$$

F-score

$$F - score = \frac{2 \times Prec \times Recall}{Prec + Recall} \tag{5.9}$$

which gives the harmonic mean of the precision and recall.

Chamfer Distance

$$CD = \frac{1}{|P|} \sum_{p \epsilon P} \min_{p^* \epsilon P^*} \|p - p^*\|_2^2 + \frac{1}{|P^*|} \sum_{p^* \epsilon P^*} \min_{p \epsilon P} \|p - p^*\|_2^2 \tag{5.10}$$

5.3. Performance of Individual Methods

We study the effects of the proposed methods on detailed 3D scene reconstruction in this section. We follow the same ordering from subsection 4.1.2, and therefore we also organize the experiments presented in this section as follows: 1) TSDF & occupancy supervision, 2) surface geometry supervision, 3) depth-based supervision and 4) resolution. Please note that, although we follow this ordering to present our experiments, our methods are independent from each other and experiments were performed for each method individually.

For each of the methods, we first mention the additional experimentation and evaluation details, if required. Then, we discuss the quantitative evaluation results obtained, while providing a comparison with the baseline. At the end of the section, we provide a brief summary of the findings and introduce our final combined approach.

5.3.1. TSDF & Occupancy Supervision

In this section, we present the experimental results of our methods proposed to improve the TSDF & occupancy supervision. Therefore, we deliver the results for: 1) gradient-based supervision from subsection 4.2.1, 2) TSDF generation from meshes introduced in subsection 4.2.2, and 3) simple occupancy probability prediction from subsection 4.2.3.

Experimentation & Evaluation Details

All of the experiments presented in this section follow the experimental methodology introduced in section 5.1. As the proposed gradient-based supervision and TSDF generation from meshes are more related than the other approaches, we deliver their quantitative results in a single table.

Gradient-based Supervision: The introduced gradient-based loss term is incorporated to the default loss function with weight λ_{grad} and we selected the weight via a grid search.

TSDF Generation from Meshes: We present the results for both scanning & sampling techniques. For each technique, we consider two models where each model corresponds to one of the introduced two approaches:

- **Approach (1):** TSDF supervision using the generated TSDFs as the ground-truth.
- **Approach (2):** Unsigned distance supervision, using distances from the generated TSDFs and signs from the default ground-truth TSDFs, as the ground-truth.

Simple Occupancy Probability Supervision: To convert the occupancy grid into a mesh, we mark the voxels as *occupied* if their occupancy probabilities are larger than a pre-defined threshold, then we extract the surface mesh using these occupied voxels. We set this occupancy threshold to 0.5 in our evaluations.

Evaluation Results

The evaluation results for gradient-based supervision and TSDF generation from meshes are provided in Table 5.1. The sampling based approaches outperform the ones based on scanning, in almost all of the metrics. Although sampling works better on watertight meshes and our meshes are not watertight, this result is expected when the gradient magnitude visualizations of SDFs generated via scanning technique are considered (see Figure 4.3). While SDFs from scanning involve a high number of artifacts which are due to the inaccurately oriented normals causing incorrectly estimated signs, we reduce the tendency to create inconsistencies in both distance and sign estimations by using 10M sampled points in the sampling technique. This situation is better observed when Approach (2) is employed, where the scanning technique almost achieves at par performance with the sampling technique, when reasoning about the sign of a voxel is not required.

Approaches using generated TSDFs from the meshes mostly provide improvements in 2D metrics considering that they generate more consistent TSDFs which is proved by the gradient distributions in Figure 4.3. However, the improvement is remained limited to depth

metrics. Similarly, the method using a gradient-based loss term surpasses the baseline in 2D metrics, while the baseline shows a better performance in geometry prediction. This limited improvement in metrics can be explained by the fact that both methods employ derived representations. For instance, ground-truth gradients are derived from the underlying ground-truth TSDF field. Therefore, they also tend to inherit any inconsistencies within the ground-truth TSDFs or inaccuracies within the TSDF fusion implementation of *NeuralRecon*. Similarly, the generated TSDFs from meshes inherit the inaccuracies within the ground-truth meshes and the MC algorithm employed.

Method	**2D Depth Metrics**				
	AbsRel ↓	AbsDiff ↓	SqRel ↓	RMSE ↓	2D Comp ↑
Gradient-based supr.	0.707	0.213	2.129	0.444	**0.864**
Approach (1) *smpl*	**0.647**	**0.203**	**2.118**	**0.422**	0.854
Approach (1) *scan*	0.937	0.237	3.318	0.489	0.781
Approach (2) *smpl*	0.760	0.213	2.503	0.439	0.838
Approach (2) *scan*	0.782	0.228	2.592	0.462	0.839
baselineInt-4	0.914	0.228	2.970	0.457	0.858
Method	**3D Geometry Metrics**				
	Recall ↑	Prec ↑	F-score ↑	3D Comp ↓	CD ↓
Gradient-based supr.	0.572	0.839	0.673	0.200	0.191
Approach (1) *smpl*	0.564	0.862	0.672	0.214	0.204
Approach (1) *scan*	0.485	**0.885**	0.613	0.302	0.321
Approach (2) *smpl*	0.540	0.859	0.653	0.230	0.209
Approach (2) *scan*	0.536	0.851	0.648	0.225	0.203
baselineInt-4	**0.580**	0.850	**0.682**	**0.197**	**0.184**

Table 5.1.: **Quantitative evaluation results of the introduced gradient-based supervision and supervision using generated TSDFs from meshes.** (See section 5.2 for metric definitions). The upper first model in the top and bottom blocks, was trained on the default ground-truth TSDFs by using a gradient-based loss term in addition to TSDF & occupancy losses. For other models, TSDF supervision was performed by using the generated TSDFs from meshes via either employing *sampling (smpl)* or *scanning (scan)* techniques, as the ground-truth. BCE loss was also employed while gradient-based loss was not used. By the usage of generated TSDFs via sampling, 2D metrics are dominated as the generated TSDFs are more consistent. The improvements obtained by employing the proposed methods are limited to 2D metrics, as the baseline still shows a superior performance on 3D metrics.

Using only the occupancy probabilities as a supervisory signal provides incomplete meshes which can be observed from Figure 5.1. We concluded that supervising with only occupancy values is not suitable to our current architecture, and thus we did not perform any further

quantitative evaluations.

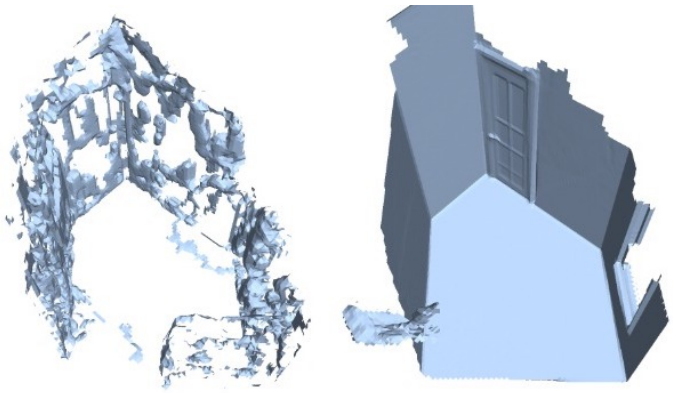

Figure 5.1.: **Qualitative evaluation results of using only occupancy values for supervision.** Considering the incompleteness of the meshes, we concluded that supervising with only occupancy values is not suitable to our current architecture.

5.3.2. Surface Geometry Supervision

In this section, we evaluate the performance of our models trained using explicit surface geometry supervision. More specifically, we deliver the results for: 1) employing a sign enforcing loss term introduced in subsection 4.3.1 and 2) point-level supervision using surface meshes from subsection 4.3.2.

Experimentation & Evaluation Details

All of the experiments presented in this section follow the experimental methodology introduced in section 5.1.

Sign Enforcing Loss Term: We present the evaluation results of the five loss modifications proposed, where we denote each of the proposed modification as **Loss modification (X)**. The proposed first three approaches require incorporating an auxiliary sign enforcing loss term to the default loss function with weight λ_{aux}. Also, the fifth modification introduces two weights, λ_{tsdf+} and λ_{tsdf-}, for positive and negative distances respectively. We selected these weights using a grid search.

Point-level Supervision using Surface Meshes: In this approach, we propose to additionally supervise our network using the ground-truth surface mesh samples. Other than employing Chamfer distance as our loss function, we also propose using the Cauchy function, shape regularizers and a densely sampling strategy. To study the effects of these components, we trained and evaluated the following models, where **PLM** refers to point-level supervision using meshes.

- **PLM only:** Only Chamfer distance is used for surface supervision. It can be considered as the simplest model.

- **PLM & Cauchy**: Cauchy function is used additionally to mitigate the effect of outliers which introduces noise to the supervision.
- **PLM & Cauchy & Smpl**: Both the predicted and ground-truth meshes are densely sampled to create adequate points for the supervision.
- **PLM & Cauchy & Smpl & SReg**: Shape regularizers are employed to prevent predicting non-smooth surfaces when Chamfer distance is minimized only.

Weight λ_{mesh} of this mesh-based loss and Cauchy cut-off parameter α were selected via a grid search. We used the publicly available implementations of the shape regularizers from PyTorch3D [80].

Evaluation Results

Table 5.2 shows how the proposed sign enforcing loss terms affect the metrics. When the proposed loss functions are employed, we generally outperform the introduced baseline in all of the metrics. Among all, Loss Modification (3) enhances the predicted 3D geometry most, which is expected given that it directly acts on the boundary voxels where there is a sign change while Loss Modification (1) and Loss Modification (2) penalize a whole voxel region if voxel signs are incorrectly predicted. Loss Modification (4), that assigns higher weights to voxels near the zero-crossing, improves the 2D metrics significantly while achieving the second best performance in 3D metrics among other loss modifications. This is also expected considering that it encourages the network to learn the zero-crossing transitions by explicitly giving more importance to these boundaries, and therefore can be considered as a good modification to the default loss function.

The results for point-level supervision using surface meshes are shown in Table 5.3. The proposed surface mesh supervision method outperforms the baseline easily when Cauchy function is used together with the densely sampling strategy. With the Cauchy-based loss function, we explicitly enforce the network to consider surface points within a predefined range which are also the points that can provide an accurate supervisory signal. Therefore, the noise generated by outliers are mostly removed from the supervision, which is reflected as a major improvement in 2D metrics when compared to the only Chamfer distance used version. By additionally sampling the surface meshes densely, we achieve a substantial improvement in 3D completeness and Chamfer distance, which is expected given that we introduce more surface samples to estimate the 3D geometry. We observe that, though shape regularizers mitigate the surface artifacts, they also tend to remove the fine-scale details within the surface. They can be still exploited for reconstructing scenes consisting of mostly larger coarse structures, however they are not suitable for complex scenes with higher-level details. Therefore, 2D and 3D metrics drop slightly in the utilization of shape regularizers.

When the quantitative evaluation results of surface geometry supervision are considered, both the proposed loss modifications and the point-level surface mesh supervision contribute significantly towards bringing back higher-level details and enhancing the predicted 3D geometry as a whole. While a major improvement on the accuracy of rendered depth maps is

Method	2D Depth Metrics				
	AbsRel ↓	AbsDiff ↓	SqRel ↓	RMSE ↓	2D Comp ↑
Loss modification (1)	0.824	0.238	2.672	0.482	0.839
Loss modification (2)	**0.743**	0.225	2.515	0.460	0.848
Loss modification (3)	0.843	**0.220**	2.717	0.457	0.862
Loss modification (4)	0.783	0.223	**2.357**	**0.449**	0.859
Loss modification (5)	1.020	0.255	3.063	0.492	**0.863**
baselineInt-4	0.914	0.228	2.970	0.457	0.858
Method	**3D Geometry Metrics**				
	Recall ↑	Prec ↑	F-score ↑	3D Comp ↓	CD ↓
Loss modification (1)	0.557	0.863	0.667	0.224	0.211
Loss modification (2)	0.570	**0.864**	0.678	0.216	0.203
Loss modification (3)	**0.593**	0.843	**0.690**	**0.193**	**0.179**
Loss modification (4)	0.578	0.857	0.682	0.203	0.193
Loss modification (5)	0.572	0.843	0.674	0.209	0.193
baselineInt-4	0.580	0.850	0.682	0.197	0.184

Table 5.2.: **Quantitative evaluation results of the introduced loss modifications.** (See section 5.2 for metric definitions). Loss Modification (3) enhances the predicted 3D geometry most, which is expected given that it directly acts on the boundary voxels where there is a sign change. Loss Modification (4), that assigns higher weights to voxels near the zero-crossing, improves the 2D metrics significantly while achieving the second best performance in 3D metrics among other loss modifications. From the comparison with the baseline, we concluded that utilizing an auxiliary loss term to encourage a correct TSDF sign prediction is advantageous.

Method	2D Depth Metrics				
	AbsRel ↓	AbsDiff ↓	SqRel ↓	RMSE ↓	2D Comp ↑
PLM only	0.842	0.228	2.564	0.464	0.854
PLM & Cauchy	**0.594**	**0.191**	**1.880**	**0.421**	0.858
PLM & Cauchy & Smpl	0.657	0.198	2.035	0.431	**0.876**
PLM & Cauchy & Smpl & SReg	0.625	0.209	2.018	0.443	0.859
baselineInt-4	0.914	0.228	2.970	0.457	0.858
Method	**3D Geometry Metrics**				
	Recall ↑	Prec ↑	F-score ↑	3D Comp ↓	CD ↓
PLM only	0.588	0.842	**0.686**	0.193	0.187
PLM & Cauchy	0.558	0.845	0.662	0.225	0.224
PLM & Cauchy & Smpl	**0.598**	0.822	**0.686**	**0.184**	**0.182**
PLM & Cauchy & Smpl & SReg	0.580	0.842	0.679	0.204	0.196
baselineInt-4	0.580	**0.850**	0.682	0.197	0.184

Table 5.3.: **Quantitative evaluation results of the introduced point-level supervision using surface meshes.** (See section 5.2 for metric definitions). The surface mesh supervision approach combined with the Cauchy function and the densely sampling strategy, improves both the 2D and 3D metrics considerably, while shape regularizers also smoothen the fine-scale details causing a slight drop in the metrics. The employed Cauchy function (PLM & Cauchy) succeeds in mitigating the effects of outliers and enhances the metrics significantly when compared to only Chamfer distance used version (PLM only). With sampling (PLM & Cauchy & Smpl), a substantial improvement is achieved in 3D completeness and Chamfer distance as we provide more surface samples to estimate the 3D geometry.

observed with the use of surface mesh supervision, this improvement is relatively limited for the proposed loss modifications.

5.3.3. Depth-based Supervision

Within this section, we study the effects of using depth maps in supervision rather than relying on processed representations such as TSDFs from the TSDF fusion algorithm and surface meshes extracted via the MC algorithm. We present the results for three depth-based proposed methods: 1) occupancy detection using depth maps from subsection 4.4.1, 2) point-level supervision using depth maps introduced in subsection 4.4.2 and 3) TSDF supervision using depth maps from subsection 4.4.3.

Experimentation & Evaluation Details

All of the experiments presented in this section follow the experimental methodology introduced in section 5.1.

Point-level Supervision using Depth Maps: Weight of the introduced loss term and Cauchy cut-off parameter α were selected by performing a grid search. Having found out, shape regularizers do not contribute as expected to the supervision, we did not use them in this experiment. We denote this model as **PLD & Cauchy & Smpl**. Please recall that, this approach is introduced as an alternative to the point-level supervision using surface meshes. Therefore, we compare these two approaches whenever required.

TSDF Supervision using Depth Maps: We selected the truncation distance ρ as $12cm$, similar to [12].

Evaluation Results

The results for occupancy detection using depth maps are provided in Table 5.4. The proposed approach surpasses the baseline in almost all of the 2D metrics and achieves comparable results in 3D metrics. Specifically, we observe improvements in 3D completeness and Chamfer distance which are due to the accurate propagation of occupancy information to the finer levels. By also keeping the voxels in which a back-projected depth pixel falls into, we remove the possibility of a voxel being sparsified when an incorrectly lower occupancy probability is predicted for that voxel. However, as occupancy predictions are still used together with the back-projected depth maps, our approach do not prevent hallucinating structures in the voxels that are incorrectly predicted as *occupied*.

We present the results of point-level supervision using depth maps in Table 5.5. Similar to the point-level supervision using surface meshes, using back-projected depth maps for this purpose improves both the 2D and 3D metrics considerably. As this approach relies on actual sensor measurements for supervision, it can be also considered as a more robust alternative for surface geometry supervision when compared to using surface meshes for the same purpose.

Method	2D Depth Metrics				
	AbsRel ↓	AbsDiff ↓	SqRel ↓	RMSE ↓	2D Comp ↑
Occupancy det. w. depth	**0.747**	**0.210**	**2.440**	**0.435**	0.855
baselineInt-4	0.914	0.228	2.970	0.457	**0.858**
Method	**3D Geometry Metrics**				
	Recall ↑	Prec ↑	F-score ↑	3D Comp ↓	CD ↓
Occupancy det. w. depth	0.562	**0.864**	0.671	**0.194**	**0.181**
baselineInt-4	**0.580**	0.850	**0.682**	0.197	0.184

Table 5.4.: **Quantitative evaluation results of the introduced occupancy detection technique.** (See section 5.2 for metric definitions). The proposed approach surpasses the baseline in almost all of the 2D metrics and achieves comparable results in 3D metrics. Specifically, the 3D completeness and Chamfer distance are improved by enabling an accurate propagation of the occupancy information to the finer levels.

Method	2D Depth Metrics				
	AbsRel ↓	AbsDiff ↓	SqRel ↓	RMSE ↓	2D Comp ↑
PLD & Cauchy & Smpl	**0.790**	**0.200**	**2.527**	**0.432**	**0.874**
baselineInt-4	0.914	0.228	2.970	0.457	0.858
Method	**3D Geometry Metrics**				
	Recall ↑	Prec ↑	F-score ↑	3D Comp ↓	CD ↓
PLD & Cauchy & Smpl	**0.595**	0.836	**0.688**	**0.185**	**0.178**
baselineInt-4	0.580	**0.850**	0.682	0.197	0.184

Table 5.5.: **Quantitative evaluation results of the introduced point-level supervision using depth maps.** (See section 5.2 for metric definitions). The proposed approach using back-projected depth maps for surface geometry supervision improves both the 2D & 3D metrics substantially. We consider this approach as a more robust alternative for surface geometry supervision, when compared to point-level supervision approach that uses surface meshes for the same purpose.

Table 5.6 shows the results of TSDF supervision using depth maps. The proposed approach exhibits at par or slightly worse performance in both 2D & 3D metrics when compared to the baseline. This is because the TSDF fusion algorithm [15] uses all the frames available, while our approach uses only use N views within the local fragment window to estimate the ground-truth TSDF values. However, it is promising to see that the proposed approach shows a close-by performance to the baseline with a limited amount of views, since the accuracy of our method can be further improved by increasing the number of frames used. Additionally, the proposed approach removes the dependency to an off-the-shelf TSDF fusion algorithm. This way, the pipeline can be cleared from the additional pre-proccesing step and training can be started directly. Also, by reducing the gap between the ground-truth data and the scene representation, we provide robustness to the overall TSDF supervision and avoid possible inaccuracies within the implementation of the employed TSDF fusion algorithm.

Method	**2D Depth Metrics**				
	AbsRel ↓	AbsDiff ↓	SqRel ↓	RMSE ↓	2D Comp ↑
DepthTSDFSupr	0.944	0.237	**2.852**	0.474	0.848
baselineInt-4	**0.914**	**0.228**	2.970	**0.457**	**0.858**
Method	**3D Geometry Metrics**				
	Recall ↑	Prec ↑	F-score ↑	3D Comp ↓	CD ↓
DepthTSDFSupr	0.572	0.847	0.674	0.212	0.210
baselineInt-4	**0.580**	**0.850**	**0.682**	**0.197**	**0.184**

Table 5.6.: **Quantitative evaluation results of the introduced TSDF supervision approach using depth maps.** (See section 5.2 for metric definitions). The proposed approach provides at par or slightly worse results in the metrics. Please note that, while TSDF fusion [15] uses all the frames available, we only use N views within the local fragment window to generate the ground-truth TSDF values on-the-fly. Considering the accuracy of our method can be further improved by increasing the number of frames used, it is promising that the proposed approach shows a close-by performance to the baseline with a limited number of views.

5.3.4. Resolution

We study the effects of increasing the resolution of the finest level in this section, by focusing on the proposed 4-level architecture and neural upsampling from subsection 4.5.1 and subsection 4.5.2 respectively.

Experimentation & Evaluation Details

Our experimentation and evaluation details deviate from the introduced experimental methodology owing to the change in resolution. We cannot employ a pre-trained model this time

due to the architectural changes proposed, therefore we performed all of our trainings from scratch. We employed early stopping during training. We used the training set of *InteriorNet-full* dataset in our from-scratch trainings and again evaluated our models on the validation set of *InteriorNet-200* dataset.

It is not fair to compare an architecture that has a voxel size of 2*cm* at the finest level, with an architecture that uses 4*cm* at its last level. Therefore, we employ a 3-level baseline that is trained from scratch by using a voxel size of 2*cm* at its last level on the *InteriorNet-full* dataset. We refer to this baseline as *baselineIntScratch-2*. We compute the 3D geometry metrics for each model on the ground-truth meshes with 2*cm* resolution, considering that the improvements can be much more easily observed when higher resolution meshes are employed.

Evaluation Results

Table 5.7 demonstrates the quantitative evaluation results of resolution related experiments. Inserting a fourth layer with a voxel resolution of 2*cm* achieves better results in both 2D & 3D metric evaluations when compared to baseline, while the model using neural upsampling falls behind both the other models. Since neural upsampling eliminates several steps including back-projection and GRU fusion, it cannot benefit from the multi-level information incorporated from these stages. This shows the contribution of these steps to the base architecture while also revealing that a direct neural upsampling is not suitable for increasing the resolution in our network.

We relate the minor improvement in the performance of the 4-level architecture when compared to 3-level one with the increased capacity of the network. However, the difference in metrics between these two architectures, is not significant. Therefore, considering the trade-off between accuracy and efficiency, either 4-level or 3-level architecture with a voxel size of 2*cm* at the finest level can be selected.

5.3.5. Combined Approach

In this section, we briefly summarize our findings and introduce our final approach that is a combination of the individually proposed approaches. We select the approaches to be combined, considering the quantitative results presented in the previous sections.

From the experiments on individual methods, we observed that surface geometry supervision using either back-projected depth maps (**PLD**) or sampled surface points from the target mesh (**PLM**), improves the accuracy of the reconstructions significantly. We further compare these two surface geometry supervision approaches using a cumulative distribution plot for the distances between the predicted vertices and the target vertices. As it can be observed from Figure 5.2, **PLD** exhibits better accuracy, which is an expected behavior as it relies on actual sensor measurements for supervision rather than samples of the approximated 3D meshes. In addition to that, considering the robustness of using depth maps as the source of supervision and possible inaccuracies in the target meshes, we select **PLD** as our main surface geometry supervision approach.

Method	2D Depth Metrics				
	AbsRel ↓	AbsDiff ↓	SqRel ↓	RMSE ↓	2D Comp ↑
4-level arch.	**1.181**	**0.261**	**3.569**	**0.494**	0.848
Neural upsampling	1.480	0.366	4.427	0.609	0.798
baselineIntFull-2	1.266	0.272	3.950	0.510	**0.849**
Method	**3D Geometry Metrics**				
	Recall ↑	Prec ↑	F-score ↑	3D Comp ↓	CD ↓
4-layer Architecture (2cm)	**0.552**	**0.855**	**0.661**	**0.213**	**0.210**
Neural upsampling (2cm)	0.462	0.816	0.578	0.317	0.360
baselineIntFull-2 (2cm)	0.544	0.844	0.652	0.220	0.219

Table 5.7.: **Quantitative evaluation results of the introduced 4-level architecture and neural upsampling technique.** (See section 5.2 for metric definitions). 4-level architecture achieves better results in both 2D & 3D metric evaluations when compared to baseline. However, neural upsampling falls behind the other models owing to not exploiting the multi-level information incorporated from steps like back-projection and GRU fusion.

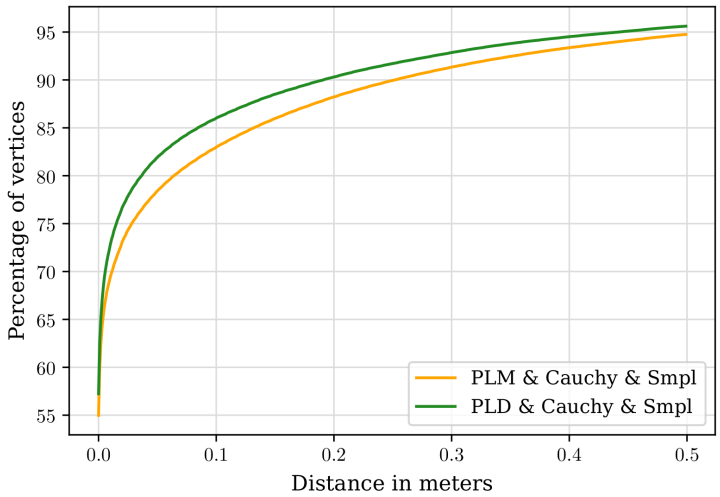

Figure 5.2.: **Cumulative error distributions of PLM and PLD approaches.** Using back-projected depth maps for surface geometry prediction yields superior accuracy when compared using target surface samples for the same purpose.

By using a sign enforcing loss term, we specifically encouraged the network to predict the signs of the distances correctly, and therefore we obtained more accurate zero-crossing predictions.

We avoided the loss of details at the earlier levels caused by an incorrect sparsification of voxels, via exploiting depth maps as an auxiliary tool. This is also reflected to evaluations with a considerable increase in 2D & 3D metrics. By eliminating the TSDF fusion algorithm from the pipeline and employing an on-the-fly ground-truth TSDF value estimation procedure, we could achieve on par or slightly worse performance to the baseline approach using TSDF fusion in most of the metrics. Although our approach does not use all of the available frames, we consider obtaining a close-by performance with only N frames within the local window as a promising achievement. Considering that the accuracy of our method can be further improved by increasing the number of views used, we decided to include this method to our final approach.

By benefiting from a mesh to SDF conversion algorithm, we showed that more consistent SDF representations can be obtained that satisfy the Eikonal property. However, while approaches using TSDFs generated from the meshes or a gradient-based loss term contributed to some of the metrics, improvements were limited to 2D metrics only. We concluded that, as both the TSDFs generated from meshes and gradients are derived from the ground-truth meshes or ground-truth TSDFs respectively, these are inherently prone to inaccuracies within them. In addition to that, they also provide limited additional information to the supervision owing to being derived representations.

Considering that the state-of-the-art employs a voxel resolution of $4cm$ and using a finer resolution can be always employed to improve the accuracy in the presence of sufficient compute power and memory, we do not include 4-level architecture to our combined approach. This way, we can obtain a fair final model to be compared with state-of-the-art.

Considering the quantitative evaluation results of the proposed methods and these observations, we choose the approaches to be combined as follows:

- Sign enforcing loss term, Loss Modification (3)
- Point-level supervision using depth maps
- Occupancy detection using depth maps
- TSDF supervision using depth maps

5.4. Comparison with State-of-the-art

In this section, we present quantitative and qualitative evaluation results of our final combined approach introduced in the previous section. We perform our experiments on both InteriorNet [16] & ScanNet [17] datasets and provide a detailed comparison with state-of-the-art. The descriptions of the datasets are available in Table 3.1.

5.4.1. Evaluations on InteriorNet

Experimentation & Evaluation Details

Since these are the experiments for evaluating the performance of the final combined approach, we use the whole data that is readily available instead of using smaller subsets. Therefore, we employ the previously introduced *InteriorNet-full* for both training and evaluation. We use ground-truth TSDFs at 4*cm* resolution in our training and adopt ground-truth meshes at 2*cm* resolution for the metric evaluation on the test set.

We again employ a pre-trained model and further train it using our final combined approach. The pre-trained model is the base architecture from *NeuralRecon* with the default configuration introduced in Table 4.1. It was trained on *InteriorNet-full* data with 500 training scenes at a voxel-grid resolution of 4*cm*. We used Adam optimizer with $\beta_1 = 0.9$ and $\beta_1 = 0.999$, and an initial learning rate of 1e-3 with a 0.5 decay at the epochs 12, 24, 48. We performed the training for 60 epochs. For further training our model on the pre-trained model, we employed a learning rate of 1.25e-4 with a 0.5 decay at every 12 epochs. During training, we did not apply early stopping and we waited until convergence.

For comparison purposes, we trained a model from scratch on *InteriorNet-full* data without any modifications. We used the aforementioned training details used for the pre-trained model. However, to provide a fair comparison with our final approach, we employed a learning rate of 1.25e-4 with a 0.5 decay at every 12 epochs, after training this model for 60 epochs. During training, we did not apply early stopping and we waited until convergence. We use this model as our baseline from *NeuralRecon* and denote it as **NeuralRecon [12]**. Following the evaluation procedure of individual approaches described in section 5.1, we generate single-layered meshes from our double-layered predicted meshes using the TSDF fusion and MC implementations of *Open3D*. We also use the same approach to generate single-layered ground-truth meshes at 2*cm* resolution and perform our evaluations on them.

Evaluation Results

Table 5.8 shows how our method performs when compared to *NeuralRecon* [12] on InteriorNet data. Our method outperforms *NeuralRecon* in 3D metrics by improving the metrics consistently. Specifically, in 3D completeness and Chamfer distance, we achieve approximately a 15% decrease with respect to *NeuralRecon*. We also achieve a considerable improvement in *F-score*, which reflects the accuracy and again completeness of our reconstructions. For 2D metrics, we have comparable performance to [12].

Additionally, we would like to highlight that we mostly improve in fine-scale geometric details which are usually not visible in these average-based metrics. Therefore, to better exhibit the performance of our approach, we provide the cumulative distribution plot in Figure 5.3, for distances between the vertices of the predicted surface meshes and the target surface meshes. Our method outperforms *NeuralRecon* with a significant margin by having more than 85% of the vertices in the first 0.05 meters, where the same number is 76% for *NeuralRecon*. In the first 0.5 meters, we still deliver superior performance. This result proves that our method can successfully preserve fine-scale details while reconstructing the overall

Method	2D Depth Metrics				
	AbsRel ↓	AbsDiff ↓	SqRel ↓	RMSE ↓	2D Comp ↑
NeuralRecon [12]	**0.583**	**0.169**	1.823	**0.384**	0.875
Ours	0.584	0.174	**1.687**	0.397	**0.890**
Method	**3D Geometry Metrics**				
	Recall ↑	Prec ↑	F-score ↑	3D Comp ↓	CD ↓
NeuralRecon [12]	0.556	**0.878**	0.672	0.240	0.246
Ours	**0.600**	0.854	**0.698**	**0.206**	**0.205**

Table 5.8.: **Comparison in 2D depth metrics & 3D geometry metrics between our method and *NeuralRecon* [12] on InteriorNet dataset.** Our method exhibits superior performance in 3D metrics when compared to *NeuralRecon*, by improving the metrics consistently. Especially in 3D completeness and Chamfer distance, we achieve more than a 15% decrease with respect to [12]. In 2D metrics, our method shows a similar performance to [12]. We would like the recall that we mostly improve in fine-scale geometric details which are usually not visible in these average-based metrics. Please see Figure 5.3 which addresses this issue and provides a more suitable quantitative comparison for our task. Qualitative results are presented in Figure 5.5 and Figure 5.6.

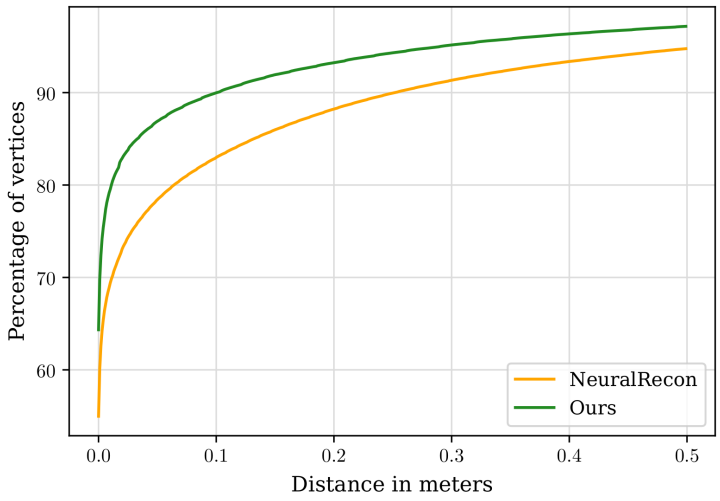

Figure 5.3.: **Cumulative distribution of the distances between the vertices of the predicted surface meshes and the target surface meshes.** Our method performs better when compared to *NeuralRecon* by predicting more than 85% of the vertices in the first 0.05 meters, while the same number is 76% for *NeuralRecon*. This result shows that we can reconstruct fine-scale details while reconstructing the overall scene accurately.

scene accurately. We additionally provide Figure 5.4 to represent the amount of predicted closest vertices within the first 0.05 meters to the corresponding vertices of the target mesh. The green areas represent where we can predict vertices within the 0.05 meters to the target, while the red areas represent the vertices that we cannot predict in this range. It can be observed that our method is able to reconstruct many fine-scale details such as the chair legs, flowers in the vase, sharp edges as well as corners, by predicting the vertex locations accurately within the first 0.05 meters. Our method also performs better on coarser structures, e.g. reconstructs the wall more straight in this example. However, *NeuralRecon* predicts less vertices within the first 0.05 meters.

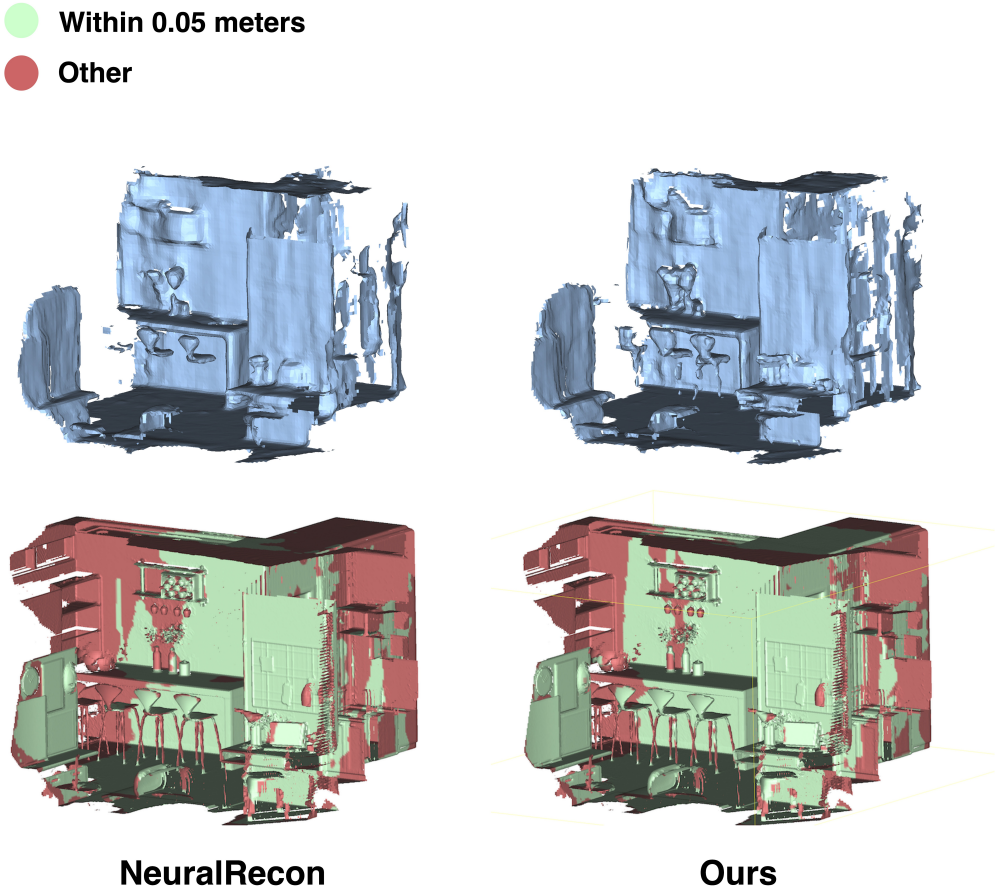

Figure 5.4.: **Qualitative representation of the amount of predicted closest vertices within the first 0.05 meters to the corresponding vertices of the target mesh.** The upper row represents the predictions from our method and *NeuralRecon*. Closest predicted vertices within the 0.05 to target vertices are represented in green, while the others are represented in red. Our method is able to predict more vertices within this range when compared to *NeuralRecon*. It can be observed that we reconstruct many fine-scale details such as the chair legs, flowers in the vase, sharp edges as well as corners.

Unfortunately, the widely employed average-based 2D & 3D metrics tend to overlook such improvements in detailed objects, owing to the relatively smaller magnitude of contributions of these details to the metrics when compared to the contributions of larger structures. Also, these metrics are prone to be dominated by the outliers. Therefore, more suitable quantitative evaluations such as the one illustrated in Figure 5.3 and qualitative results provide a better intuition towards the performance on detailed 3D reconstruction. With this purpose, we present the qualitative comparison of rendered depth maps in Figure 5.5 and 3D surface meshes in Figure 5.6. Our results reveal that we can successfully reconstruct more challenging structures in many cases and produce a sharper geometry when compared to [12]. Even on high-frequency objects such as trees, our method is able to deliver a relatively good performance.

5.4.2. Evaluations on ScanNet

Experimentation & Evaluation Details

For experiments on ScanNet, we use the readily available train/validation/test splits with the given descriptions in Table 3.1. We use ground-truth TSDFs generated at 4*cm* resolution in our training.

We employ the model provided by the authors of *NeuralRecon* [12] as our pre-trained model and further train it based on the final combined approach. For further training our model, we employed a learning rate of 1.25e-4 with a 0.5 decay at every 12 epochs. During training, we did not apply early stopping and we waited until convergence.

We consider the model provided by the authors of *NeuralRecon* as our state-of-the-art baseline and compare our final model using the reported numbers in [12]. As the state-of-the-art also employs the same train/validation/test splits on ScanNet, this direct comparison is fair. We again obtain single-layered predicted meshes from the double-layered ones. Then, we perform our evaluations on the provided ground-truth meshes by ScanNet.

Evaluation Results

In Table 5.8, we report the results of our method and *NeuralRecon* on ScanNet dataset. Our method shows comparable performance to *NeuralRecon* in 2D metrics and achieves similar or slightly worse results in 3D metrics. We present the qualitative results for the extracted 3D surface meshes in Figure 5.7. Although it is not directly reflected to quantitative evaluations, the qualitative results show that our method is able to preserve local structures while reconstructing the overall 3D scene geometry accurately.

Referring back to the main reason of employing a synthetic dataset, we consider employing high quality data, for both training and evaluation purposes, more suitable for our task as it is able to represent fine-scale details. Employing almost perfect ground-truth data provides us with an ideal setting that is cleared from the inaccuracies of real-world datasets. However, as ScanNet is a real-world dataset, the provided ground-truth color images, depth maps, 3D surface meshes involve inaccuracies and noise. This can be better observed from Figure 3.4 which provides sample RGB images and their corresponding color coded depth

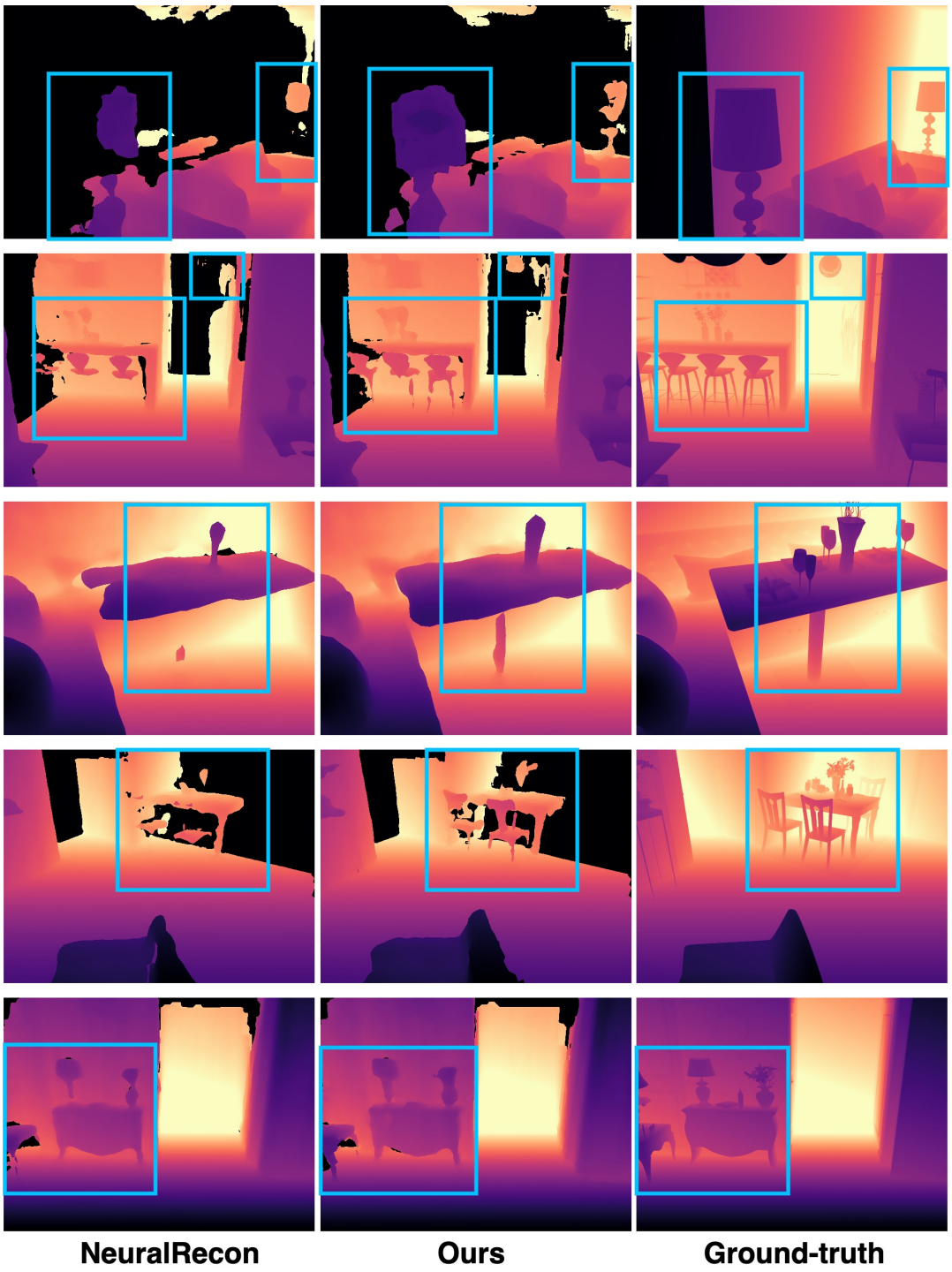

Figure 5.5.: **Qualitative comparison of rendered depth maps**. Our method is able to produce much more detailed reconstructions with sharper geometry. Notice that we recover fine-scale details such as chair legs, table lamp, the structure of the vase etc. when compared to *NeuralRecon*, which exhibits the effectiveness of our approach. Please see Table 5.8 for metric evaluation results.

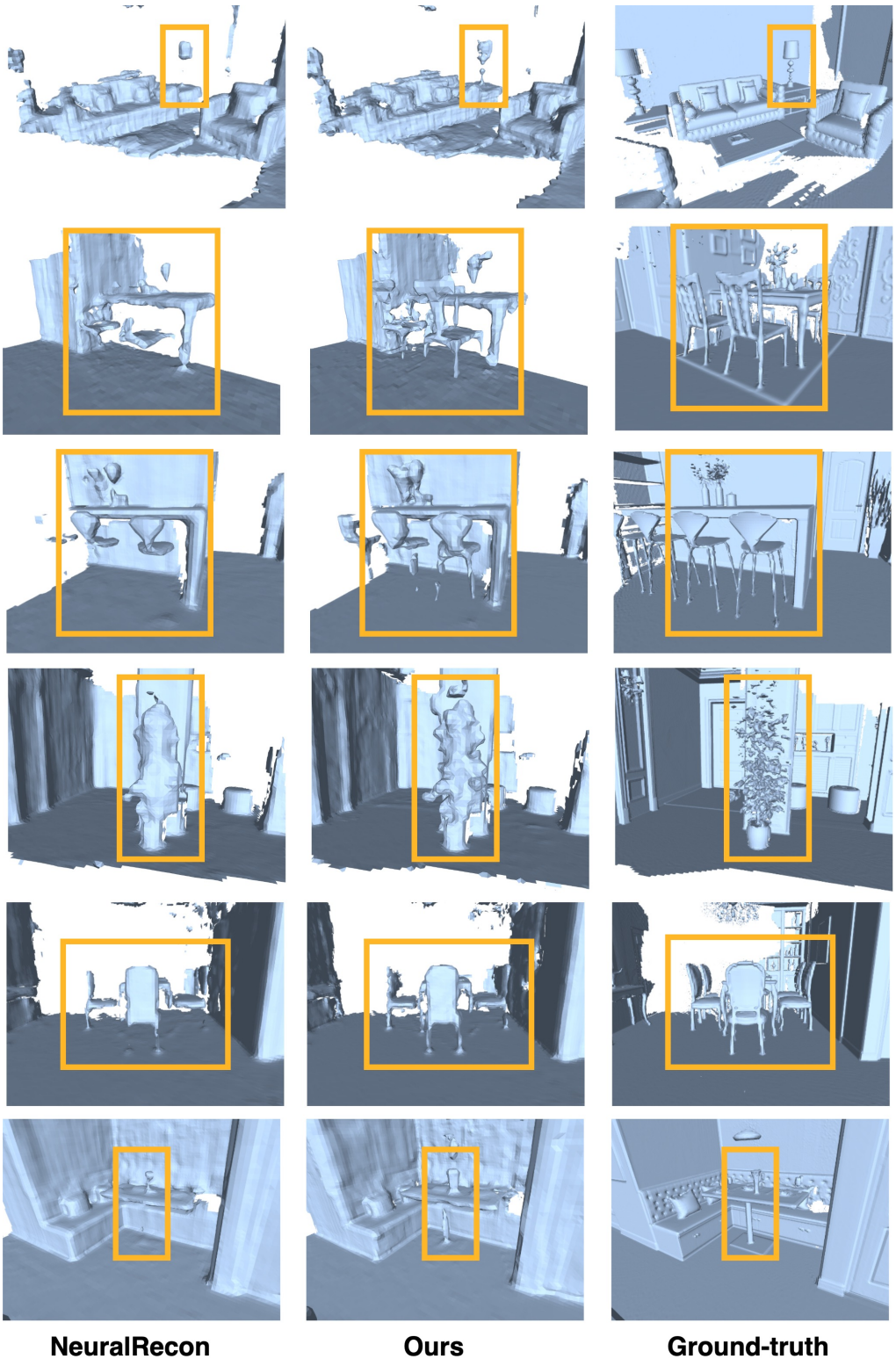

Figure 5.6.: **Qualitative comparison of extracted surface meshes**. Our method delivers a superior performance to *NeuralRecon* by preserving details from many challenging structures. Please see Table 5.8 for metric evaluation results.

Method	2D Depth Metrics				
	AbsRel ↓	AbsDiff ↓	SqRel ↓	RMSE ↓	2D Comp ↑
NeuralRecon [12]	0.065	0.106	**0.031**	**0.195**	0.909
Ours	**0.064**	**0.100**	0.034	**0.195**	**0.901**
Method	**3D Geometry Metrics**				
	Recall ↑	Prec ↑	F-score ↑	3D Comp ↓	CD ↓
NeuralRecon [12]	**0.479**	**0.684**	**0.562**	**0.128**	0.080
Ours	0.474	0.654	0.548	0.134	**0.079**

Table 5.9.: **Comparison in 2D depth metrics & 3D geometry metrics between our method and *NeuralRecon* [12] on ScanNet dataset.** Our method exhibits a comparable performance to [12] in 2D metrics and presents similar or slightly worse results in 3D metrics. Please see Figure 5.7 for the qualitative results.

maps. Therefore, since the ground-truth is already limited at representing high-level details, it is reflected negatively to both supervision and evaluation of our model.

5.5. Discussion

The presented qualitative results and cumulative distribution plot for distances between the target & predicted vertices given in Figure 5.3 show that, our method is able to accurately preserve high-level details while producing a sharper geometry when compared to *NeuralRecon*. Although this achieved reconstruction accuracy is reflected to 3D metrics, the improvements observed on both the depth and 3D geometry metrics remained limited than the expected amounts. This is because the widely employed average-based 2D & 3D metrics used for evaluating the performance of 3D reconstructions, tend to unfortunately overlook improvements in detailed objects.

A major reason of this issue is the surfaces representing these details occupy only a small portion of the whole 3D surface mesh representing the overall scene. Therefore, fine-scale details contribute in small amounts to metrics when compared to larger structures. After the averaging is applied, the metrics are mostly dominated by the costs from larger objects. This makes the reliability of these metrics limited for tasks such as detailed 3D reconstruction since the improved accuracy cannot be directly reported as the reconstructions get more detailed.

Another issue is that these metrics are prone to be dominated by the outliers. Even if a majority of the vertices are predicted within a very close distance to the target vertices, a single outlier point that is extremely far away from the closest target surface sample may cause a jump in the metrics such as the Chamfer distance. This mitigates all the positive contributions of correctly predicted vertices and makes the metric dominated by the outlier point.

Considering that recovering fine-scale details while reconstructing complex scenes is crucial

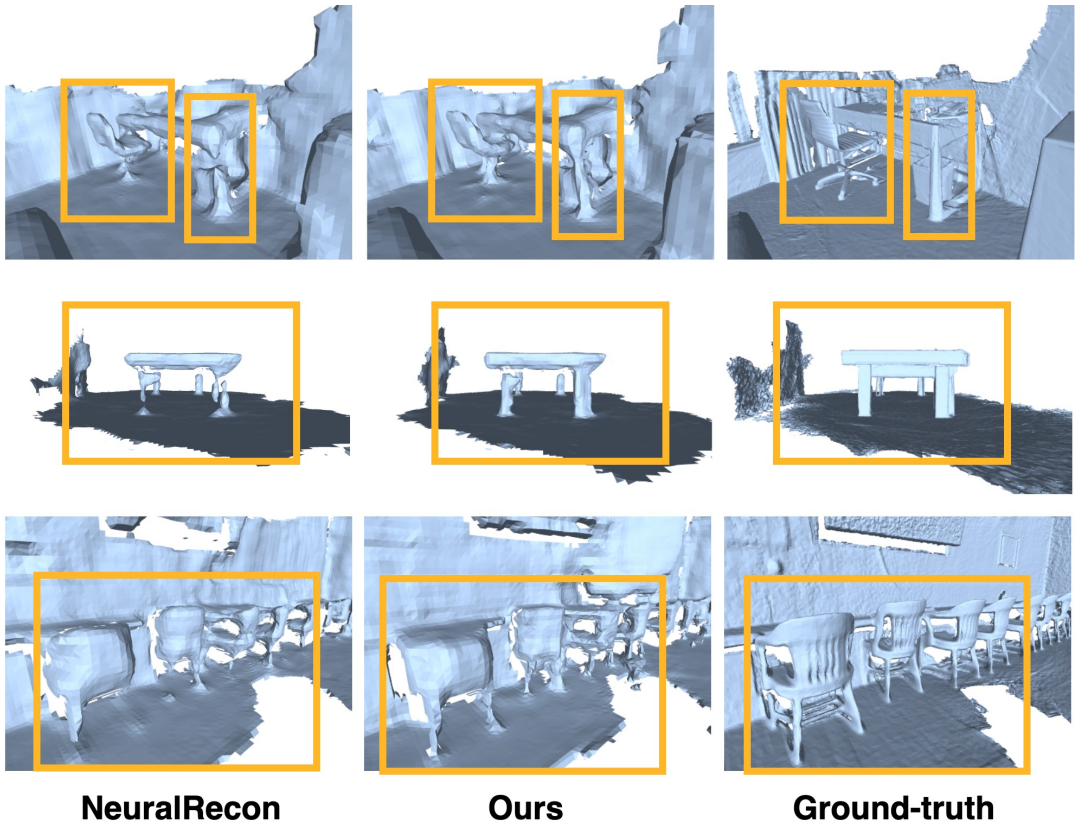

Figure 5.7.: **Qualitative comparison of extracted surface meshes**. Our method better preserves the details from high-frequency objects when compared to *NeuralRecon*. Please see Table 5.9 for metric evaluation results.

for many applications ranging from robotic navigation to augmented reality, more reliable metrics are required to accurately assess the detailed 3D reconstruction performance. We suggest employing a cumulative distribution of distances between the target and predicted vertices within a specified range as an additional evaluation tool. Such a quantitative comparison given in Figure 5.3 provides, observing the reconstruction behavior especially on fine-scale structures much more easily. Additionally, the percentage of vertices at a particular distance, e.g. 0.05 meters, can be reported and this number can be used as a metric for evaluating fine-scale 3D geometry.

6. Conclusion

The task of 3D scene reconstruction from multi-view images has crucial importance for many applications including autonomous driving, augmented reality and motion planning. However, preserving finer-scaled details while reconstructing the overall scene has a challenging nature. In this work, we investigated the factors affecting the quality of reconstructions by exploiting the mathematical along with structural properties of the employed scene representations, and proposed approaches addressing each of the factors. We then selectively combined the presented methods into a single approach and introduced our final voxel-based neural detailed 3D scene reconstruction method. We extensively evaluated our approach on the widely employed InteriorNet and ScanNet datasets using the common 2D & 3D metrics.

One of the main contributions of this thesis is the proof that, surface geometry supervision either by using target surface samples or back-projected depth maps, is extremely beneficial for the recovery of fine-scale details. We further boost the performance of the proposed surface geometry supervision approaches by densely sampling the meshes rather than using only the vertices to provide adequate data for the supervision. The employed Cauchy function mitigate the effects of outliers to the supervision by enforcing a decreasing gradient distribution when the cut-off is exceeded. We observe that the employed shape regularizers smoothen also the fine-scale details, and therefore they are not suitable for complex scenes like ours.

We show that the target TSDF values required for TSDF supervision can be estimated on-the-fly using depth-maps, removing the dependency to the TSDF fusion algorithm. This way, the pipeline can be cleared from the additional pre-proccesing step and training can be started directly. Additionally, by reducing the gap between the ground-truth sensor measurements and the scene representation, we provide robustness to the overall TSDF supervision and avoid any kind of inaccuracies within the implementation of the employed fusion algorithm. Please recall that, our approach uses only N views within a local window, unlike the TSDF fusion algorithm which uses all the available views. Therefore, our current metric results provided in subsection 5.3.3 can be further improved by increasing the number of views used.

The experimental results of using a sign enforcing auxiliary loss term presented in subsection 5.3.2 demonstrate that the network can be explicitly enforced to output the correct sign for the distance values. This brings a substantial boost to the performance of the regular TSDF supervision and provides an accurate zero-crossing prediction. Modifying the existing TSDF loss function by assigning weights to voxels that are inversely proportional with their corresponding ground-truth absolute distance to the zero-crossing, has also exhibited a notable performance.

Using back-projected depth maps to detect the occupied voxels proves to be advantageous, when the improvements in metrics, especially in 3D completeness and Chamfer distance are considered. As the possibility of a voxel being sparsified in the case of an incorrect lower

occupancy probability prediction is avoided, information can be propagated to the finer levels accurately.

Gradient magnitude distributions provided in subsection 4.2.1 show that the Eikonal property cannot be exploited to produce valid signed distance fields in our case. This is because this property is not well enforced in the ground-truth TSDFs obtained from the TSDF fusion implementation of *NeuralRecon*. From these findings, we conclude that the ground-truth TSDFs from the fusion algorithm may not be sufficiently consistent and accurate. The experiments in subsection 5.3.1 reveal that the alternatively proposed gradient-based loss term provides limited advantage. Considering gradients are derived quantities and they inherit the inaccuracies within the ground-truth TSDFs they are derived from, we conclude that employing gradients is also not suitable to our setup.

Although the employed mesh to TSDF generation procedure proposed to obtain more consistent TSDF representations shows promising results in some metrics, the improvement remained limited to depth metrics only. We again relate this finding to several approximations involved in this TSDF to mesh and then mesh to TSDF conversion process, which makes the whole procedure error prone.

In our experiments, we did not observe a major difference between using a 4-level or a 3-level architecture as long as the resolution of the finest level is the same. Therefore, considering the trade-off between accuracy and efficiency, either 4-level or 3-level architecture with a finer voxel size at the last level can be selected.

The experimental evaluations on both InteriorNet and ScanNet datasets show that the performance of the proposed approach in detailed 3D scene reconstruction depends directly on the quality of the data. In the presence of high quality data, the ground-truth used for supervision and evaluation also represent fine-scale details. In this case, our method shows a superior performance to state-of-the-art. However, if the data is noisy, it is directly reflected to ground-truth TSDFs and meshes. Therefore, the learning task for reconstructing fine-scale details becomes much more challenging, which negatively affects the performance of our method. Other than learning, noisy data also affects the evaluation procedure negatively as the details are not visible also in the target.

As a future work, the effect of the number of views on the TSDF supervision using depth maps can be further explored, to find the adequate number of frames required to provide a more accurate supervision. With the employment of additional views, the accuracy and robustness gained by relying on actual sensor measurements can be better observed. Furthermore, for the point-level supervision approach using surface meshes, higher-quality meshes than the current ones can be employed. This will bring a substantial boost to the performance of the current approach, while enabling producing much more detailed surfaces.

A. Appendix

A.1. Data Pre-processing

For ScanNet dataset, we resize the images to 640×480. We apply random rotation and translation as well as padding to the ground-truth TSDF fragments for training.

The images from InteriorNet already have the size of 640×480, therefore they are not resized prior to training. Random rotation and translation are applied to the ground-truth TSDF fragments, together with the padding.

List of Figures

List of Tables

Acronyms

BCE binary cross-entropy. 19, 33, 41

CNN Convolutional Neural Network. 1, 6–10, 12

CRF Conditional Random Field. 8

CVC Colored Voxel Cube. 9

DPV Depth Probability Volume. 8

ESDF Euclidean Signed Distance Field. 28

FBV Fragment Bounding Volume. 18

FPN Feature Pyramid Network. 6, 8

GP Gaussian Process. 8

GRU Gated Recurrent Unit. 10, 18, 42, 43

LSTM Long Short-Term Memory. 9

MAP Maximum-a-Posterior. 12

MC Marching Cubes. 11, 12, 22, 45

MLP Multi-Layer Perceptron. 10, 13, 18, 37, 39, 42

MVS multi-view stereo. 1, 2, 6–10, 18, 36

SDF Signed Distance Function. 12, 26, 28, 30–32, 37, 48, 59, 73

SLAM Simultaneous Localization and Mapping. 19

SPP Spatial Pyramid Pooling. 6

TSDF Truncated Signed Distance Function. 1–3, 10–12, 17–19, 22–25, 28–34, 36–41, 43–49, 54, 56, 59, 72, 73, 75, 76

List of Symbols

$\mathcal{M}$	Triangular mesh
$\boldsymbol{F}$	Faces of mesh $\mathcal{M}$
$\boldsymbol{F}_t^l$	Feature volume from level l at time step t
$\boldsymbol{G}_t^l$	3D geometric features from level l at time step t
$\boldsymbol{H}_t^g$	Global hidden state from level l at time step t
$\boldsymbol{H}_t^l$	Hidden state from level l at time step t
$\boldsymbol{S}_t^l$	TSDF volume from level l at time step t
$\boldsymbol{V}$	Vertices of mesh $\mathcal{M}$
α	Cauchy cut-off parameter
λ_{aux}	Weight of L_{aux}^l
λ_{grad}	Weight of L_{grad}^l
λ_{occ}	Weight of L_{occ}^l
λ_{tsdf+}	Weight for voxels with positive ground-truth TSDF value
λ_{tsdf-}	Weight for voxels with negative ground-truth TSDF value
λ_{tsdf}	Weight of L_{tsdf}^l
ρ	Truncation distance
$\boldsymbol{D}(u,v)$	Depth map
$\boldsymbol{I}(u,v)$	Color image
θ	Sparsification threshold
d_{max}	Depth threshold
L_{grad}^l	Gradient-based loss term from level l
L_{occ}^l	BCE occupancy loss from level l

L^l_{tot} Total loss from level l

L^l_{tsdf} L_1 TSDF loss from level l

L^l_{aux} Auxiliary loss from level l

L^l_{CD} Chamfer distance based loss

L_{edge} Mesh edge length regularization loss

L_{Lap} Laplacian regularizer

L^l_{max} Loss max

L_{mesh} Mesh-based loss

L^l_{min} Loss min

L_{NC} Normal consistency loss

R_{max} Relative rotation threshold

t_{max} Relative translation threshold

Bibliography

[1] M.-D. Yang, C.-F. Chao, K.-S. Huang, L.-Y. Lu, and Y.-P. Chen. "Image-based 3D scene reconstruction and exploration in augmented reality". In: *Automation in Construction* 33 (2013), pp. 48–60.

[2] N. Funk, J. Tarrio, S. Papatheodorou, M. Popović, P. F. Alcantarilla, and S. Leutenegger. "Multi-Resolution 3D Mapping With Explicit Free Space Representation for Fast and Accurate Mobile Robot Motion Planning". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3553–3560. DOI: 10.1109/LRA.2021.3061989.

[3] M. Popović, F. Thomas, S. Papatheodorou, N. Funk, T. Vidal-Calleja, and S. Leutenegger. "Volumetric Occupancy Mapping With Probabilistic Depth Completion for Robotic Navigation". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 5072–5079. DOI: 10.1109/LRA.2021.3070308.

[4] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan. "Mvsnet: Depth inference for unstructured multi-view stereo". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 767–783.

[5] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang. "Deepmvs: Learning multi-view stereopsis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2821–2830.

[6] Y. Yao, Z. Luo, S. Li, T. Shen, T. Fang, and L. Quan. "Recurrent mvsnet for high-resolution multi-view stereo depth inference". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5525–5534.

[7] R. Chen, S. Han, J. Xu, and H. Su. "Point-based Multi-view Stereo Network". In: *The IEEE International Conference on Computer Vision (ICCV)*. 2019.

[8] X. Gu, Z. Fan, S. Zhu, Z. Dai, F. Tan, and P. Tan. "Cascade cost volume for high-resolution multi-view stereo and stereo matching". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2495–2504.

[9] K. Wang and S. Shen. "MVDepthNet: Real-Time Multiview Depth Estimation Neural Network". In: *2018 International Conference on 3D Vision (3DV)*. Los Alamitos, CA, USA: IEEE Computer Society, Sept. 2018, pp. 248–257. DOI: 10.1109/3DV.2018.00037. URL: https://doi.ieeecomputersociety.org/10.1109/3DV.2018.00037.

[10] A. Duzceker, S. Galliani, C. Vogel, P. Speciale, M. Dusmanu, and M. Pollefeys. "DeepVideoMVS: Multi-view stereo on video with recurrent spatio-temporal fusion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15324–15333.

[11] Z. Murez, T. van As, J. Bartolozzi, A. Sinha, V. Badrinarayanan, and A. Rabinovich. "Atlas: End-to-End 3D Scene Reconstruction from Posed Images". In: *ECCV*. 2020. URL: https://arxiv.org/abs/2003.10432.

[12] J. Sun, Y. Xie, L. Chen, X. Zhou, and H. Bao. "NeuralRecon: Real-Time Coherent 3D Reconstruction from Monocular Video". In: *CVPR* (2021).

[13] M. G. Crandall and P.-L. Lions. "Viscosity solutions of Hamilton-Jacobi equations". In: *Transactions of the American mathematical society* 277.1 (1983), pp. 1–42.

[14] W. E. Lorensen and H. E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM siggraph computer graphics* 21.4 (1987), pp. 163–169.

[15] B. Curless and M. Levoy. "A volumetric method for building complex models from range images". In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 1996, pp. 303–312.

[16] W. Li, S. Saeedi, J. McCormac, R. Clark, D. Tzoumanikas, Q. Ye, Y. Huang, R. Tang, and S. Leutenegger. "InteriorNet: Mega-scale Multi-sensor Photo-realistic Indoor Scenes Dataset". In: *British Machine Vision Conference (BMVC)*. 2018.

[17] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. "ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes". In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*. 2017.

[18] J. Choe, S. Im, F. Rameau, M. Kang, and I. S. Kweon. "Volumefusion: Deep depth fusion for 3d scene reconstruction". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16086–16095.

[19] Y. Furukawa and C. Hernández. "Multi-View Stereo: A Tutorial". In: *Foundations and Trends® in Computer Graphics and Vision* 9.1-2 (2015), pp. 1–148. ISSN: 1572-2740. DOI: 10.1561/0600000052. URL: http://dx.doi.org/10.1561/0600000052.

[20] N. D. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. "Using multiple hypotheses to improve depth-maps for multi-view stereo". In: *European Conference on Computer Vision*. Springer. 2008, pp. 766–779.

[21] Y. Furukawa and J. Ponce. "Accurate, dense, and robust multiview stereopsis". In: *IEEE transactions on pattern analysis and machine intelligence* 32.8 (2009), pp. 1362–1376.

[22] M. Lhuillier and L. Quan. "A quasi-dense approach to surface reconstruction from uncalibrated images". In: *IEEE transactions on pattern analysis and machine intelligence* 27.3 (2005), pp. 418–433.

[23] C. H. Esteban and F. Schmitt. "Silhouette and stereo fusion for 3D object modeling". In: *Computer Vision and Image Understanding* 96.3 (2004), pp. 367–392.

[24] C. Hernández, G. Vogiatzis, and R. Cipolla. "Probabilistic visibility for multi-view stereo". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8.

[25] R. Collins. "A space-sweep approach to true multi-image matching". In: *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1996, pp. 358–363. DOI: 10.1109/CVPR.1996.517097.

[26] R. Yang and M. Pollefeys. "Multi-resolution real-time stereo on commodity graphics hardware". In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* 1 (2003), pp. I–I.

[27] H. Ha, S. Im, J. Park, H.-G. Jeon, and I. S. Kweon. "High-quality depth from uncalibrated small motion clip". In: *Proceedings of the IEEE conference on computer vision and pattern Recognition*. 2016, pp. 5413–5421.

[28] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. "Real-Time Plane-Sweeping Stereo with Multiple Sweeping Directions". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383245.

[29] S. Im, H. Ha, G. Choe, H.-G. Jeon, K. Joo, and I. S. Kweon. "Accurate 3D Reconstruction from Small Motion Clip for Rolling Shutter Cameras". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.4 (2019), pp. 775–787. DOI: 10.1109/TPAMI.2018.2819679.

[30] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys. "Pixelwise view selection for unstructured multi-view stereo". In: *European Conference on Computer Vision*. Springer. 2016, pp. 501–518.

[31] E. Zheng, E. Dunn, V. Jojic, and J.-M. Frahm. "PatchMatch Based Joint View Selection and Depthmap Estimation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1510–1517. DOI: 10.1109/CVPR.2014.196.

[32] S. Galliani, K. Lasinger, and K. Schindler. "Massively Parallel Multiview Stereopsis by Surface Normal Diffusion". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 873–881. DOI: 10.1109/ICCV.2015.106.

[33] S. Im, H.-G. Jeon, S. Lin, and I. S. Kweon. "Dpsnet: End-to-end deep plane sweep stereo". In: *arXiv preprint arXiv:1905.00538* (2019).

[34] S. Cheng, Z. Xu, S. Zhu, Z. Li, L. E. Li, R. Ramamoorthi, and H. Su. "Deep stereo using adaptive thin volume representation with uncertainty awareness". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2524–2534.

[35] Y. Hou, J. Kannala, and A. Solin. "Multi-view stereo by temporal nonparametric fusion". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2651–2660.

[36] C. Liu, J. Gu, K. Kim, S. G. Narasimhan, and J. Kautz. "Neural rgb (r) d sensing: Depth and uncertainty from a video camera". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10986–10995.

[37] X. Long, L. Liu, W. Li, C. Theobalt, and W. Wang. "Multi-view depth estimation using epipolar spatio-temporal networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8258–8267.

[38] M. Ji, J. Gall, H. Zheng, Y. Liu, and L. Fang. "Surfacenet: An end-to-end 3d neural network for multiview stereopsis". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2307–2315.

[39] M. Ji, J. Zhang, Q. Dai, and L. Fang. "SurfaceNet+: An end-to-end 3D neural network for very sparse multi-view stereopsis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11 (2020), pp. 4078–4093.

[40] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction". In: *European conference on computer vision*. Springer. 2016, pp. 628–644.

[41] A. Kar, C. Häne, and J. Malik. "Learning a multi-view stereo machine". In: *Advances in neural information processing systems* 30 (2017).

[42] M. Tatarchenko, A. Dosovitskiy, and T. Brox. "Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2088–2096.

[43] A. Božič, P. Palafox, J. Thies, A. Dai, and M. Nießner. "TransformerFusion: Monocular RGB Scene Reconstruction using Transformers". In: *Proc. Neural Information Processing Systems (NeurIPS)* (2021).

[44] M. Kazhdan, M. Bolitho, and H. Hoppe. "Poisson surface reconstruction". In: *Proceedings of the fourth Eurographics symposium on Geometry processing*. Vol. 7. 2006.

[45] M. Kazhdan and H. Hoppe. "Screened poisson surface reconstruction". In: *ACM Transactions on Graphics (ToG)* 32.3 (2013), pp. 1–13.

[46] P. Labatut, J.-P. Pons, and R. Keriven. "Robust and efficient surface reconstruction from range data". In: *Computer graphics forum*. Vol. 28. 8. Wiley Online Library. 2009, pp. 2275–2290.

[47] A. Dai, C. Ruizhongtai Qi, and M. Nießner. "Shape completion using 3d-encoder-predictor cnns and shape synthesis". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5868–5877.

[48] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. "Kinectfusion: Real-time dense surface mapping and tracking". In: *2011 10th IEEE international symposium on mixed and augmented reality*. IEEE. 2011, pp. 127–136.

[49] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. "Real-time 3D reconstruction at scale using voxel hashing". In: *ACM Transactions on Graphics (ToG)* 32.6 (2013), pp. 1–11.

[50] O. Kähler, V. Prisacariu, J. Valentin, and D. Murray. "Hierarchical voxel block hashing for efficient integration of depth images". In: *IEEE Robotics and Automation Letters* 1.1 (2015), pp. 192–197.

[51] G. Riegler, A. O. Ulusoy, H. Bischof, and A. Geiger. "Octnetfusion: Learning depth fusion from data". In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 57–66.

[52] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner. "Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4578–4587.

[53] A. Dai, C. Diller, and M. Nießner. "Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 849–858.

[54] B. Graham, M. Engelcke, and L. van der Maaten. "3D Semantic Segmentation With Submanifold Sparse Convolutional Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[55] S. Weder, J. Schonberger, M. Pollefeys, and M. R. Oswald. "Routedfusion: Learning real-time depth map fusion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4887–4897.

[56] S. Weder, J. L. Schonberger, M. Pollefeys, and M. R. Oswald. "NeuralFusion: Online depth fusion in latent space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3162–3172.

[57] Y. Liao, S. Donne, and A. Geiger. "Deep marching cubes: Learning explicit surface representations". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2916–2925.

[58] E. Remelli, A. Lukoianov, S. Richter, B. Guillard, T. Bagautdinov, P. Baque, and P. Fua. "Meshsdf: Differentiable iso-surface extraction". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 22468–22478.

[59] U. Wickramasinghe, E. Remelli, G. Knott, and P. Fua. "Voxel2mesh: 3d mesh model generation from volumetric data". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2020, pp. 299–308.

[60] B. Guillard, E. Remelli, A. Lukoianov, S. Richter, T. Bagautdinov, P. Baque, and P. Fua. "DeepMesh: Differentiable Iso-Surface Extraction". In: *arXiv preprint arXiv:2106.11795* (2021).

[61] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. "Occupancy networks: Learning 3d reconstruction in function space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4460–4470.

[62] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[63] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, T. Funkhouser, et al. "Local implicit grid representations for 3d scenes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6001–6010.

[64] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li. "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2304–2314.

[65] M. Atzmon, N. Haim, L. Yariv, O. Israelov, H. Maron, and Y. Lipman. "Controlling neural level sets". In: *Advances in Neural Information Processing Systems* 32 (2019).

[66] M. Atzmon and Y. Lipman. "Sal: Sign agnostic learning of shapes from raw data". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2565–2574.

[67] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman. "Implicit geometric regularization for learning shapes". In: *arXiv preprint arXiv:2002.10099* (2020).

[68] A. Heyden and M. Pollefeys. "Multiple view geometry". In: *Emerging topics in computer vision* 90 (2005), pp. 180–189.

[69] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences. Springer New York, 2006. ISBN: 9780387227467.

[70] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman. "Implicit Geometric Regularization for Learning Shapes". In: *Proceedings of Machine Learning and Systems 2020*. 2020, pp. 3569–3579.

[71] C. Dapogny and P. Frey. "Computation of the signed distance function to a discrete contour on adapted triangulation". In: *Calcolo* 49.3 (2012), pp. 193–219.

[72] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. "Mnasnet: Platform-aware neural architecture search for mobile". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2820–2828.

[73] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[74] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration". In: *ACM Transactions on Graphics (ToG)* 36.4 (2017), p. 1.

[75] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. "Real-time 3D reconstruction at scale using voxel hashing". In: *ACM Transactions on Graphics (ToG)* 32.6 (2013), pp. 1–11.

[76] Q.-Y. Zhou, J. Park, and V. Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).

[77] Y. Jiang, D. Ji, Z. Han, and M. Zwicker. "Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1251–1261.

[78] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart. "Signed distance fields: A natural representation for both mapping and planning". In: *RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*. University of Michigan. 2016.

[79] D. Eigen, C. Puhrsch, and R. Fergus. "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network". In: *CoRR* abs/1406.2283 (2014). arXiv: 1406.2283. URL: http://arxiv.org/abs/1406.2283.

[80] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari. "Accelerating 3D Deep Learning with PyTorch3D". In: *arXiv:2007.08501* (2020).