Technical University of Munich

Department of Civil, Geo and Environmental Engineering

Chair of Computational Modelling and Simulation

Point Cloud Completion by Deep Learning

Master Thesis

for the Master of Science Degree in Civil Engineering

Author: [redacted]

Matriculation Number: [redacted]

1. Supervisor: [redacted]

2. Supervisor: [redacted]

3. Supervisor: [redacted]

Date of Issue: 15. January 2021

Date of Submission: 15. July 2021

Abstract

Today, measuring of buildings and rooms is still undertaken with tachymeters and partly also with measuring tapes. However, in recent years the share of 3D laser scanners has rapidly increased, since they are more accurate, less prone to human error and faster in capturing a scene. When scanning a scene, some parts of the scene are missing due to occlusions by objects. An example of this is a cabinet standing in front of a wall. The 3D scanner then only captures a point cloud with the surface of the cabinet and the surface information of the wall behind it is missing.

In order to predict these occluded points in a point cloud, this master thesis analyzes the existing methods of scene completion and proposes a novel approach to convert a point cloud into a truncated signed distance field. This truncated signed distance field is then used on an existing volumetric scene completion network that is fully self-supervised.

Additionally, the suitability of the LiDAR sensor of iPhones and iPads for generating a dataset is examined.

Zusammenfassung

Die Vermessung von Gebäuden und Räumen wird heutzutage zum Großteil immer noch mit Tachymetern und teilweise auch mit Maßbändern vorgenommen. In den letzten Jahren hat jedoch der Anteil der 3D-Laserscanner rasant zugenommen, da sie genauer, weniger anfällig für menschliche Fehler und schneller bei der Erfassung einer Szene sind. Beim Scannen einer Szene fehlen einige Teile der Szene aufgrund von Verdeckungen durch Objekte. Ein Beispiel hierfür ist ein Schrank, der vor einer Wand steht. Der 3D-Scanner erfasst dann nur eine Punktwolke mit der Oberfläche des Schranks und die Oberflächeninformation der dahinter liegenden Wand fehlt.

Um diese verdeckten Punkte in einer Punktwolke vorhersagen zu können, analysiert diese Masterarbeit die bestehenden Methoden der Szenenvervollständigung und schlägt einen neuartigen Ansatz vor, um eine Punktwolke in ein TSDF umzuwandeln. Dieses TSDF wird dann auf ein bestehendes volumetrisches Szenenvervollständigungsnetzwerk angewendet.

Zusätzlich wird die Eignung eines LiDAR-Sensors von iPhones und iPads zur Erzeugung eines 3D Datensatzes untersucht.

Content

List of Figures

List of Tables

List of Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
3D-CNN	Three-Dimensional Convolutional Network
3D-EPN	3D-Encoder Predictor Network
CAD	Computer-Aided Design
CNN	Convolutional Neural Network
DF	Distance Field
DL	Deep Learning
DoF	Three Domains of Freedom
DNN	Deep Neural Network
GAN	Generative Adversarial Network
LiDAR	Light Detection and Ranging
ML	Machine Learning
OC	Object Completion
PCN	Point Completion Network
RGB	Red-Blue-Green
RGB-D	Red-Blue-Green-Depth
SC	Scene Completion
SDF	Signed Distance Function
SG-NN	Sparse Generative Neural Network
SSC	Semantic Scene Completion
TDF	Truncated Distance Field
TSD	Truncated Signed Distance Field

1 Introduction

1.1 Motivation

Compared with other industries such as the automotive industry, the construction industry remains largely dominated by manual labor. One of the reasons is that each project is unique, and it is difficult to realize them with traditional methods such as robots in the automotive industry. However, with the rapid advance of artificial intelligence in recent years, a major leap in efficiency in the design and construction phase is expected in the next few years.

In other areas of our lives, machine learning has already become an increasingly important aspect and it is changing our future forever. It is now ubiquitous in areas such as customer service automation, communications, cybersecurity, and object recognition.

The range of opportunities with machine learning in the field of the building industry is immense. In the construction phase, the possibilities range from autonomous robots that – for example – excavate foundations, erect walls and transport building materials to construction monitoring with drones that automatically perform a cross-check with the construction plans. In the planning phase, the possibilities range from simulations of the expected construction costs through to automatic verification of compliance with building regulations and automatic measurement of buildings.

This master thesis deals with the latter-mentioned topic, namely the measurement of buildings. Tachymeters and even sometimes measuring tapes are still a widely used method for measuring buildings today. However, in recent years 3D laser scanners have become increasingly popular for measuring existing buildings. The laser scanner outputs the environment as a point cloud. The advantage of the 3D laser scanner is obvious, whereby they allow a much faster and completer capture of the scene. Moreover, it is also less prone to human error. Thus, 3D laser scanners are perfectly suited for generating the digital models of buildings. Digital building models are needed for renovation measures, but since the original models were often drawn by hand, they do not exist. For example, in North America 90% of houses do not even have a digital floor plan [34].

A persisting drawback of 3D scans is that the environment and scene cannot be captured completely due to occlusions by objects. An example of this is a cabinet standing in front of a wall, whereby the 3D scanner then only captures a point cloud with the surface of the cabinet and the surface information of the wall behind it is missing.

To complete the missing points caused by occlusion in a 3D scan, this master thesis focuses on scene completion.

1.2 Research Objective

The main goal of this master thesis is to develop an algorithm for the completion of point clouds. As a first step, existing methods are analyzed. There are numerous methods for the completion of scenes acquired with RGB-D sensors. However, unfortunately there are only a few approaches for completion based on point clouds. To overcome this issue, this master thesis aims to provide an approach that accomplishes this. The main contributions are listed below.

Contributions:

- The source code of SG-NN [12] was updated from Pytorch 1.1.0 and Python 2.7 to run with Pytorch 1.8.1 and Python 3.8.

- A novel method for transforming a point cloud into a truncated signed distance field (TSDF) was proposed.

- It was investigated whether a truncated distance field (TDF) could be used instead of a TSDF.

- An analysis of the usability of LiDAR sensors from iPhones and iPads to generate a large-scale dataset was conducted.

1.3 Structure of This Master Thesis

This master thesis is structured as follows.

- Chapter 2 provides an overview of the five different 3D data representations and their advantages and disadvantages regarding machine learning.

- Chapter 3 describes the various 3D datasets available for training machine learning models.

- Chapter 4 provides a brief overview of segmentation approaches.

- Chapter 5 offers a broad overview of shape completion and presents different approaches with their advantages and disadvantages.

- Chapter 6 describes the chosen approach, including the architecture of the Sparse Convolutional Neural Network, an approach to convert point clouds to a TSDF, and the investigation of the usability of LiDAR sensors from an iPhone to generate a large-scale dataset.

- Chapter 7 discusses the results of the chosen approach and proposes improvements.

- Chapter 8 summarizes the main results and proposes approaches for further development.

2 3D Representations

This chapter describes the five main 3D data representations, as illustrated in Figure 1.

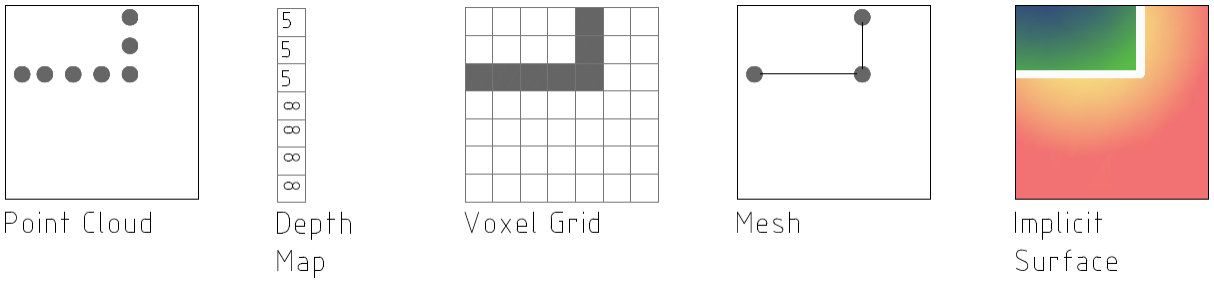

Figure 1: 3D Data Representations

2.1 Point Cloud

Point cloud is the most well-known 3D data representation. It is described by a set of unordered points with 3D coordinates (x, y, z). Only in a point cloud is the known space represented, and free or unknown space is not saved. Because of this, there is no information about neighboring points. This is why it is challenging to process point clouds in machine learning tasks. Figure 2 represents the Stanford bunny as a point cloud.

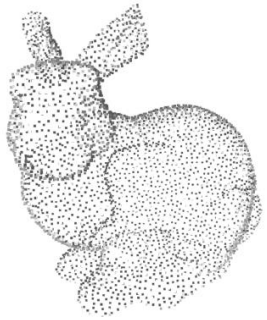

Figure 2: Point cloud of the Stanford' bunny [27]

2.2 Depth Map

Depth maps are images in which each pixel is assigned a distance. The distance or depth is the length between the camera and the object or scene. The most common of these data representations involves linking an RGB image with a depth image. These RGB-D images can easily be created with widely available sensors such as Microsoft Kinect.

2.3 Voxel Grid

A Voxel is a three-dimensional grid. Each cell of the grid is described either as occupied or free. As shown in Figure 3, one of the problems with a voxel gird is that it needs a high voxel resolution to represent a detailed scene. This results in a memory problem as the memory need increases by a power of three.

Figure 3: Chair in voxels [9]

To combat this problem, octrees are often applied in which multiple resolutions of voxel grids are used. Finer details can be represented by a finer voxel grid, as illustrated in Figure 4. This enables a more efficient memory usage, but the implementation is more difficult due to the multiple resolutions.

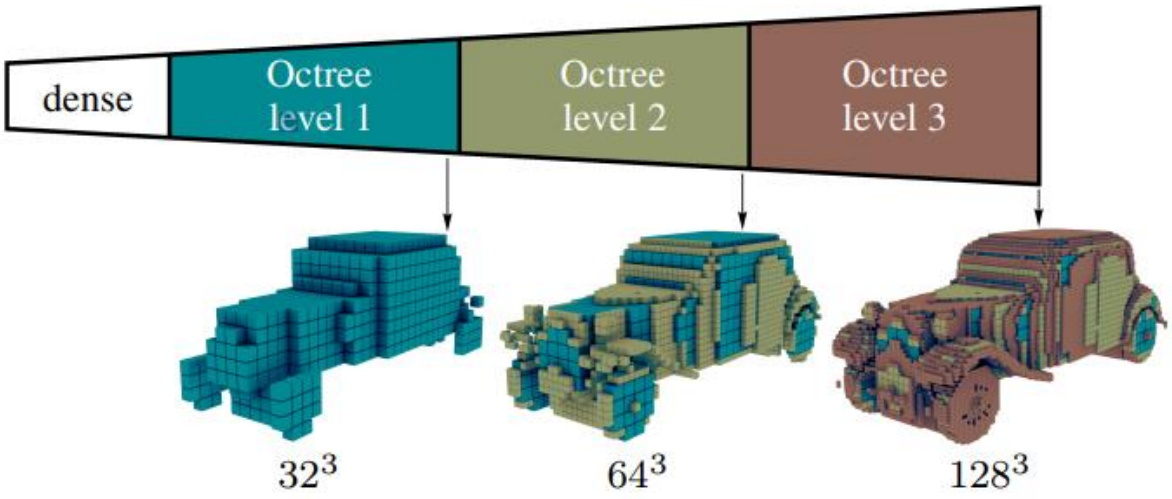

Figure 4: Octree [53]

2.4 Mesh

Meshes explicitly represent surfaces using a set of polygons. Meshes are very adaptive since they can represent flat surfaces with just one face, and detailed structures are represented with multiple faces. This makes them very efficient, but it is difficult to implement machine learning algorithms for them. Figure 5 represents the Stanford bunny as a mesh.

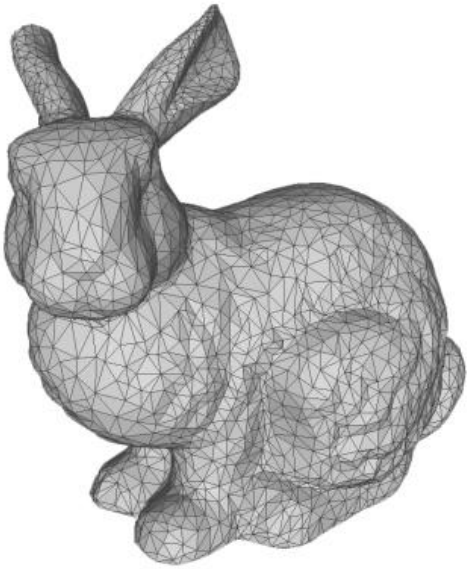

Figure 5: Mesh of the Stanford Bunny [27]

2.5 Implicit Surface

Implicit surfaces define the 3D object as a function. The function determines the probability that a point is inside or outside the object. Two very common implicit representations are the signed distance function (SDF) and the truncated signed distance function (TSDF). These both use voxels to support their implicit surface definitions.

The SDF is also called signed distance field as it provides the distance from each voxel to the closest surface. If the voxel is inside the object shape the distance is signed negative, otherwise it is signed positive. This is visualized in Figure 6.

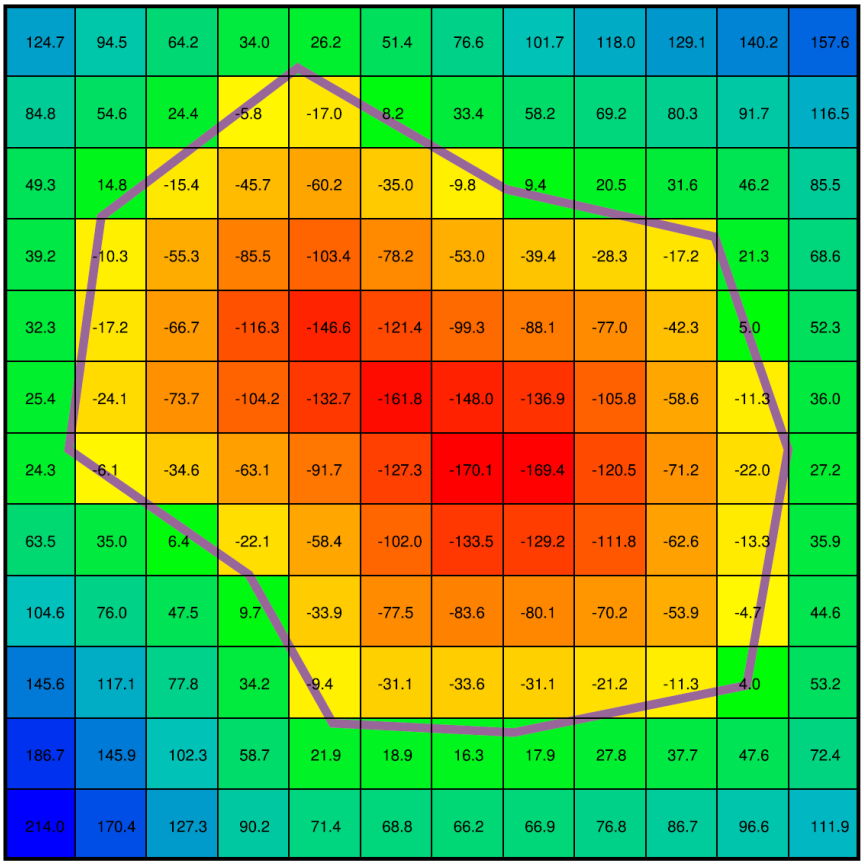

Figure 6: Signed distance field of a 2D polygon [70]

TDSF, also called truncated signed distance field, was first introduced by Curless and Levoy [10]. The function is a combination of distance and weight functions retrieved from depth images. The distance is measured by line of sight from the sensor. After a defined voxel distance, the positive and negative distances are truncated.

3 Datasets

As the performance and robustness of a deep learning model heavily depends on its training data, this chapter gives an overview of the common datasets used for shape completion and segmentation. Table 1 lists these datasets and their distinctive features.

Table 1: Overview of datasets

Dataset	Year	Data Type	Type	Nature	Scenes
NYUv1 [50]	2011	RGB-D	Real world	Indoor scene	67
NYUv2 [51]	2012	RGB-D	Real world	Indoor scene	464
SUN3D[65]	2013	RGB-D	Real world	Indoor scene	254
UZH 3D[55]	2014	Point clouds	Real world	Indoor scene	40
SceneNet[24]	2015	CAD models	Synthetic	Indoor scene	57
ModelNet [64]	2015	CAD models	Synthetic	Objects	-
ShapeNet[3]	2015	CAD models	Synthetic	Objects	-
NYUCAD[17]	2016	CAD models	Synthetic	Indoor scene	462
SUNCG[52]	2016	CAD	Synthetic	Indoor scene	45,622
Matterport3D [2]	2017	RGB-D	Real world	Indoor scene	2,056
ScanNet[11]	2017	RGB-D	Real world	Indoor scene	1,513
2D-3D-S[26]	2017	RGB-D Point clouds	Real world	Indoor scene	270
FloorNet[34]	2018	RGB-D	Real world	Indoor scene	155
SemanticKITTI[1]	2019	Point clouds	Real world	Outdoor scene	22

Most datasets are acquired with RGB-D sensors since they are widely available and cheap, but these lack accuracy compared to professional devices such as the Faro 3D laser scanner. Since acquiring a good ground truth is one of the most difficult challenges, several synthetic datasets were created. Most datasets contain mainly commercial and educational buildings since capturing private living spaces is problematic due to privacy issues. The most important datasets for scene completion are described in greater detail in the following. In Table 1 listed outdoor and object datasets are not further described, since they are so relevant for this master thesis.

NYUv2 Dataset [51]: This is most frequently used real-world dataset for indoor scene and semantic scene completion. It is an extension of the NYUv1 dataset with 464 instead of 67 scenes. The dataset is composed of residential and commercial buildings. Each pixel is labeled, and if there are multiple objects from the same object category in a scene, they are each given a separate label. In [21], the ground truth was generated for scenes containing completed, detailed 3D models.

SUN3D [65]: This dataset covers 254 scenes with RGB-D videos. It also contains camera poses, object segmentations and point clouds registered in a global coordinate frame. It differs from the NYUv2 dataset in that it obtains the camera pose and global alignment, and the scenes are less view based. This translates to a more complete representation of a scene, and thus it is better suited for scene completion.

Matterport3D [2]: Matterport3D is the largest publicly available RGB-D dataset, with 2,054 rooms containing 10,800 panoramic views, each with 18 RGB-D images covering 46,561 sqm. All camera poses with panoramic views are provided and globally registered. Since the panoramas were taken with three RGB-D cameras with their positions pointing slightly upwards, horizontal, and slightly downwards, the floor and ceiling in the scenes were often not fully scanned.

ScanNet [11]: ScanNet is a large RGB-D dataset with 1,513 scenes captured by 2,492,518 RGB-D frames covering an area of 34,453 sqm. The dataset consists of commercial buildings and apartments. The majority of rooms are hotel bedrooms. The scans are annotated with camera poses, surface reconstructions and semantic segmentations. For 107 scenes, aligned CAD models are additionally provided.

SUNCG [52]: This dataset was used most when it came to semantic scene completion It is a synthetic dataset containing over 45,000 scenes that were manually created through the Planner5D platform. Unfortunately, this dataset is no longer available due to legal matters between the company Planner5D and Princeton University.

2D-3D-S [26]: Six large-scale indoor areas with 270 rooms and 6,000 sqm are covered in this dataset. The 70,000 RGB-D images were taken from an office and educational environment. Point clouds were generated from this information and afterwards semantically annotated and assigned the following 13 object classes on a per-point basis: ceiling, floor, wall, beam, column, window, door, table, chair, sofa, bookcase, board and clutter.

SceneNet [24]: Since the SUNCG dataset is no longer available an alternative synthetic dataset is SceneNet. However, it is significantly smaller with only 57 rooms. Each scene is composed of 15 to 250 objects. All models are available in the .obj format. SceneNet RGB-D [29] builds on the SceneNet dataset by providing rendered RGB-D images from over 15 trajectories with random but physically simulated object poses. It also creates the pixel-perfect ground truth.

UZH 3D [55]: This dataset contains 40 point clouds of office rooms at the University of Zurich. The point clouds are provided as ASCII PTX files with color (x, y, z, intensity, r, g, b). This dataset is more relevant for civil engineering since it was acquired using a Faro Focus 3D laser range scanner. This translates to much better precision compared to other datasets, most of which were captured with consumer-level scanners like Microsoft Kinect.

FloorNet [34]: Since the previously discussed datasets concentrate on semantic annotations and are not suited for vector-graphics reconstruction problems, Liu et al. [34] introduced a dataset containing RGB-D videos from 155 residential units with full floorplan annotations. They also provided the associated point clouds with their annotations and associations.

4 Segmentation

4.1 Context

3D segmentation is a well-researched topic and there exist numerous open-source methods for point cloud segmentation. One possible approach to retrieve the layout of a room would be to first use segmentation, and then use the segmented walls, floor and ceiling to generate the 3D layout as, for example, in DeepPerimeter [43]. This chapter gives a brief overview of 3D segmentation.

4.2 3D Segmentation

Segmentation is the assignment of voxels, points, pixels, etc. to a specific label. Thus, the respective objects are filtered out. 3D segmentation can be subcategorized into three types as shown in Figure 7, namely semantic, instance and part segmentation.

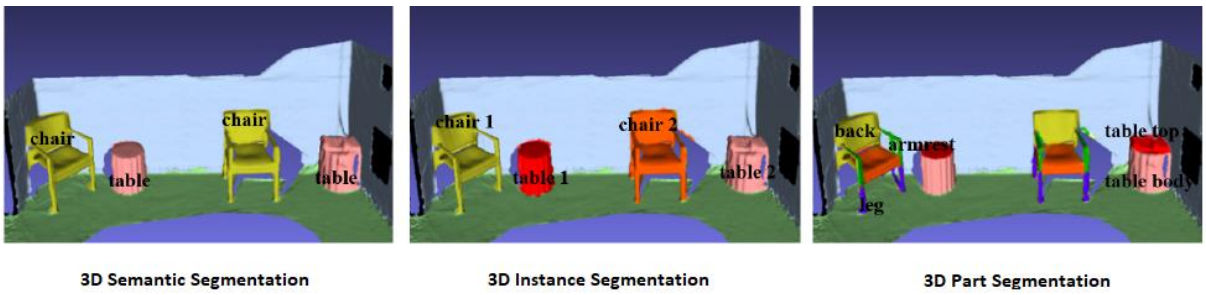

Figure 7: Types of Segmentation [25]

The semantic segmentation predicts labels for each object. For example, all chairs have the label chair. Instance segmentation, on the other hand, not only predicts the label of the object, but also separates instances with the same label. In the same example with the chairs, the chairs would be assigned the label chair, but in addition a distinction would be made between chair 1, chair 2, chair 3. In part segmentation, the segmented object is further subdivided. For example, the armrest, the backrest, the seat and the feet of a chair are labeled. Given that the field of 3D segmentation is very large, and that the focus of this thesis is the completion of point clouds, point-based segmentation is the most relevant segmentation method for the purposes of this paper.

A detailed overview was recently published by He et al.[25], in which all other methods, such as voxel base segmentation and deep image segmentation, are also described.

The pioneering work of point-based segmentation was presented by Qi et al. [44] with PointNet. Figure 8 illustrates the network architecture of PointNet. The network consists of three main modules. To bundle the information of the points, the first module is a max-pooling layer as a symmetric function. The second module contains local and global information combination structure and to the align the input point and point features the third module consists of two joint alignment networks.

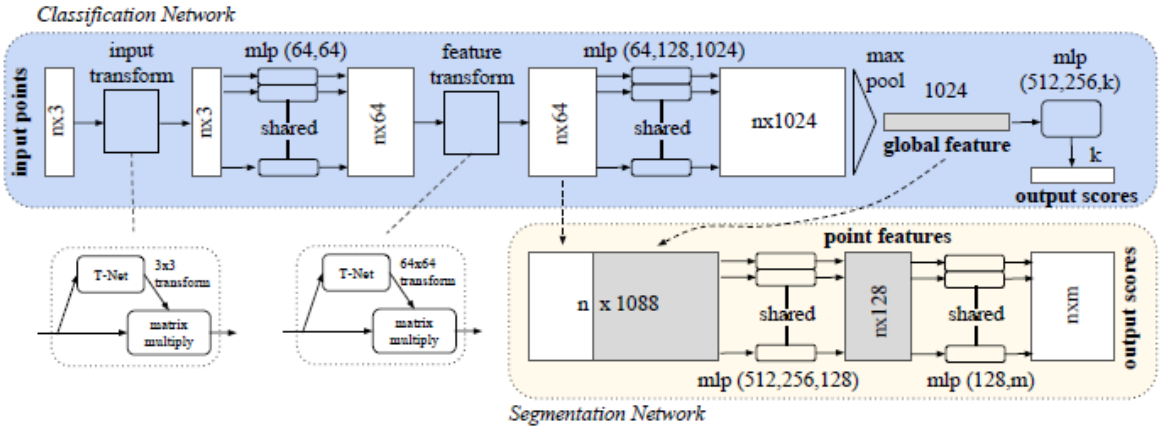

Figure 8: Architecture PointNet [44]

After PointNet, numerous methods have been developed based on it. The most prominent of these is PointNet++ [45]. PointNet++ is a hierachical feature learning network, which addresses the limitations of PointNet, in capturing local features at different scales. The PointNet++ architecture hierarchically groups points and gradually extracts larger and larger local regions up the hierarchy. In Figure 9 the architecture of PointNet++ is illustrated.

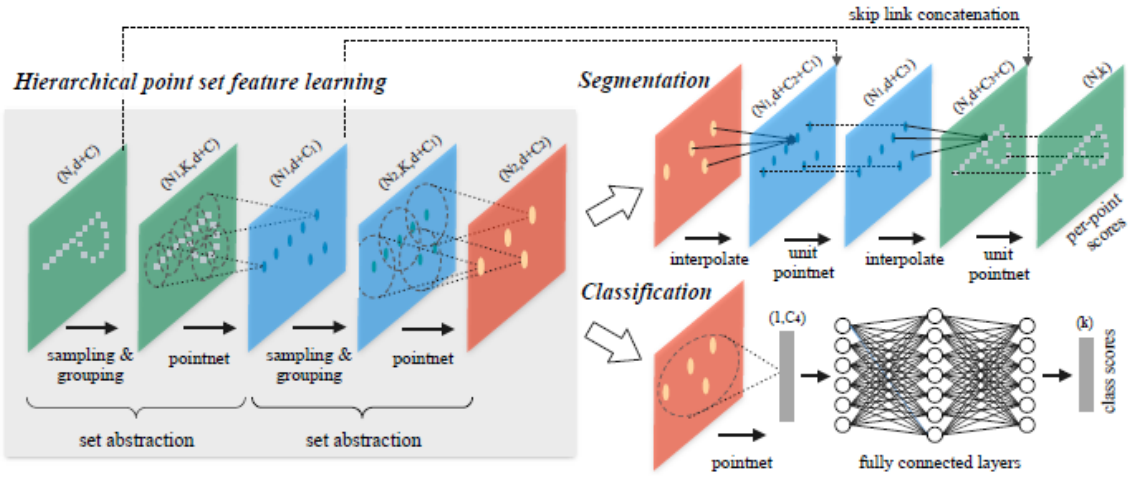

Figure 9: Architecture PointNet++ [45]

He et al. [25] categorized the many methods developed based on PointNet and PointNet++ into three types: multiple layer perceptron based, point convolution based and graph convolution based. These subcategories are split further into PointNet based frameworks and PointNet++ based frameworks.

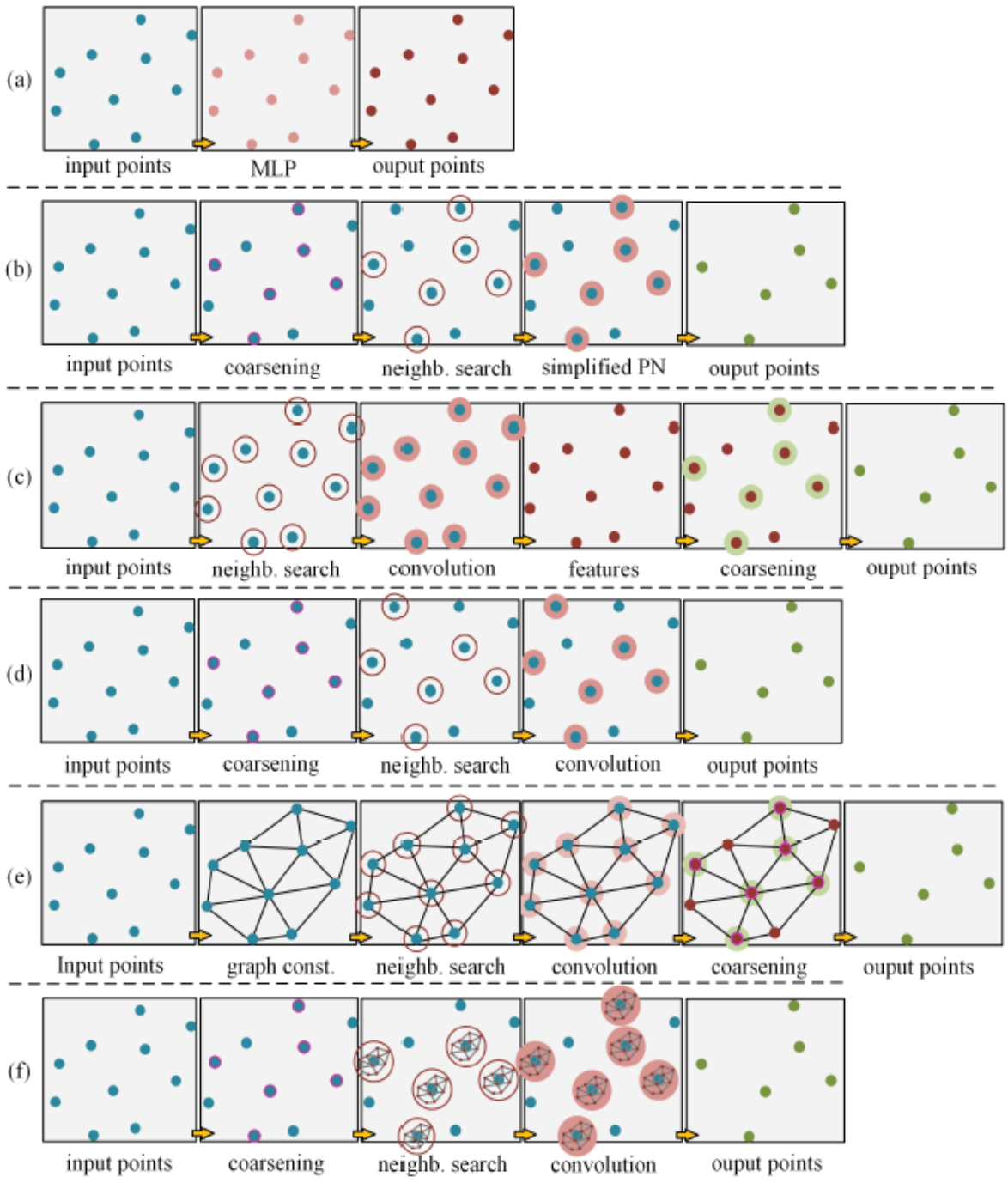

Figure 10: Point-based segmentation approaches [25]

Figure 10 visualizes the different frameworks in a very simplified way for a point- based approach. Figure (a) and (b) show the previously described multiple layer perceptron-based approach of PointNet and PointNet++. Figure (c) and (d) describe the point convolution-based approach. Framework (c) is based on PointNet and performs convolution on all neighboring points of each point in contrast to framework (d) which is based on PointNet++ and performs convolution only on specific points. The last two figures (e) and (f) represent the graph convolution-based networks. As can be seen in the figure, in the PointNet (e) based approach, the graph spans the points globally and performs convolution on all neighboring points from each point. The graph-based PointNet++ (f) approaches, on the other hand, perform convolution only on local points with a graph structure.

5 Shape Completion

As point cloud segmentation is already greatly developed, this master's thesis focuses more on 3D shape completion, which has been extensively researched, starting with small hole-filling algorithms. These traditional approaches use symmetry to complete the surface [40] or solve it with a sparse linear system [19]. Older methods use also the Poisson equation for reconstructing surfaces [30, 38, 69]. Although these methods can achieve good results with small holes, they fail to do so when applied to larger regions. In the last years, many deep learning approaches for shape completion have been developed. This chapter provides a broad overview of these methods. In general, the methods can be subcategorized into object completion (OC), scene completion (SC) and semantic scene completion (SSC). In OC, only one object is completed, in contrast to SC where many objects and the scene are completed. SSC is a combination of completion and segmentation.

Table 2 gives an overview of the existing methods for shape completion, which is comprehensive with respect to indoor scene completion. Regarding OC and outdoor scene completion, only the most prominent papers are shown as these are not so important for this master's thesis and are presented only to explain the general context. For each method, Table 2 indicates the suitable type of completion (object, indoor scene, outdoor scene, semantics). It also shows whether the source code is publicly available for further development, the input data representation of the algorithm and the datasets used for training. In the architecture column, a distinction is made between volumetric, view-volumetric and point-based approaches. A more detailed explanation of these approaches can be found in Sections 5.1 and 5.2. In addition, the defining specification is given for each architecture.

Table 2: Overview of shape completion methods

Method	Object	Scene	Indoor	Outdoor	Semantics	Open source	Input	Architecture	Dataset(s)
2015									
3D ShapeNets[64]	✓					✓	Occ. grid	**Volumetric:** Convolutional DeepBelief Network	ModelNet NYUv2
2016									
SSCNet [52]		✓	✓		✓	✓	f-TSDF	**Volumetric:** 3D CNN	SUNCG NYUv2
Nguyen et al.[41]	✓						Occ. grid RGB image	**Volumetric:** Convolutional DeepBelief Network (Markov Random Field)	ModelNet SUN Own dataset
3D-EPN [13]	✓					✓	TSDF	**Volumetric:** 3D-Encoder-Predictor	ShapeNet
2017									
Varley et al.[56]	✓					✓	Occ. grid	**Volumetric:** 3D CNN	Own dataset
Han et al[23]	✓					✓	SDF	**Volumetric:** Global structure and local geometry inference 3D-Encoder-Predictor	ShapeNet
Guedes et al. [20]		✓	✓		✓		f-TSDF RGB image	**Volumetric:** 3D CNN	NYUv2
VVNet [22]		✓	✓		✓	✓	f-TSDF Depth image	**View-volumetric:** 2D CNN + 3D CNN	SUNCG NYUv2
2018									
Cherabier et al.[7]		✓	✓		✓		TSDF	**Volumetric:** Primal dual optimization	Own dataset ScanNet
PCN[66]	✓					✓	Point cloud	**Point-based:** Encoder-Decoder architecture	ShapeNet
Scancomplete[14]		✓	✓		✓	✓	TSDF	**Volumetric:** 3D CNN	SUNCG ScanNet
SGC [67]		✓	✓		✓	✓	f-TSDF	**Volumetric:** 3D sparse CNN	SUNCG NYUv2
Wang et al.[61]		✓	✓		✓	✓	Depth image	**View-volumetric:** Adversarial learning	SUNCG NYUv2
SATNet [49]		✓	✓		✓	✓	RGB image Depth image	**View-volumetric:** 2D CNN + 3D CNN	SUNCG NYUv2

Method	Object	Scene	Indoor	Outdoor	Semantics	Open source	Input	Architecture	Dataset(s)
2019									
Li et al [32]		✓	✓		✓	✓	RGB image Depth image	**View-volumetric:** 2D DDR blocks + Atrous Spatial Pyramid Pooling	NYUv2 NYUCAD
TopNet[54]	✓					✓	Point cloud	**Point-based:** Encoder-Decoder	Shapenet
Garbade et al. [18]		✓	✓		✓		RGB image Voxel grid	**View-volumetric:** 2D CNN + 3D CNN	NYUv2 NYUCAD
EdgeNet[15]		✓	✓		✓	✓	f-TSDF RGB image	**Volumetric:** 3D CNN	SUNCG NYUv2
RL-GAN-Net[48]	✓					✓	Point cloud	**Point-based:** Reinforcement learning agent-controlled GAN network	ShapeNet
AM²FNet[4]		✓	✓		✓		f-TSDF RGB image	**Volumetric:** Multi-scale and modality fusion	NYUv2 NYUCAD
Wang et al.[60]		✓	✓		✓		Octree	**Volumetric**: Primal-dual optimi-zation	SUNCG ScanNet
Chen et al.[6]		✓	✓		✓		TSDF Depth image	**View-volumetric**: Adversarial learning	SUNCG NYUv2 NYUCAD
CCPNet[68]		✓	✓		✓		f-TSDF	**Volumetric:** Cascaded context pyramid	SUNCG NYUv2
ForkNet[62]		✓	✓		✓	✓	SDF	**Volumetric:** Single encoder →multiple generators	SUNCG NYUv2
2020									
GRFNet[35]		✓	✓		✓		RGB image Depth image	**View-volumetric:** 2D DDR blocks + Atrous Spatial Pyramid Pooling	NYUv2 NYUCAD
SG-NN[12]		✓	✓			✓	TSDF	**Volumetric:** Sparse convolution	Matterport3D
3D Sketch[5]		✓	✓		✓	✓	TSDF RGB image	**View-volumetric**: DDR blocks	SUNCG NYUv2 NYUCAD
AIC-Net [31]		✓	✓		✓	✓	RGB image Depth image	**View-volumetric:** Anisotropic Convolutional Networks	NYUv2 NYU-CAD
Wang et al.[58]		✓	✓		✓	✓	Octree	**Volumetric:** Octree-based CNN	SUNCG
Dourado et al.[16]		✓	✓		✓		f-TSDF RGB image	**Volumetric:** 3D CNN	SUNCG NYUv2 2D-3D-S own dataset

Method	Object	Scene	Indoor	Outdoor	Semantics	Open source	Input	Architecture	Dataset(s)
2020									
PALNet[36]		✓	✓		✓	✓	TSDF Depth image	**View-volumetric:** Position awareness loss	NYUv2 NYUCAD
AMFNet[33]		✓	✓		✓		RGB image Depth image	**View-volumetric**: 2D CNN + 3D CNN	SUNCG NYUv2
SPCNet[39]		✓	✓		✓		Point cloud RBG image	**Point-based:** Encoder decoder architecture	NYUv2 NYUCAD
SCFusion[63]		✓	✓		✓		Occ. grid	**Volumetric:** 3D CNN	Scan2CAD ScanNet
LMSCNet[47]		✓		✓	✓	✓	Occ. grid	**View-volumetric**: 2D CNN + Atrous Spatial Pyramid Pooling	SemanticKITTI
S3CNet[46]		✓		✓	✓		f- TSDF Depth image	**View-volumetric**: Sparse convolution	SemanticKITTI

5.1 Object Completion

As mentioned in the introduction, a wide variety of traditional 3D shape completion methods have been proposed, from minimizing surface areas to exploiting object symmetries, along with many more approaches. However, this thesis focuses only on the deep learning methods. These can be further subcategorized into volumetric shape completion and point-based shape completion. Volumetric-based shape completion was introduced chronologically before point-based completion. The use of point-based completion approaches surged greatly after the development of the PointNet encoder and its successor PointNet++ for segmentation. In the last two years in particular, many papers have adopted the point-based approach for completion as the method requires much less computation and memory costs than a volumetric approach. Four of the most relevant volumetric-based approaches and two of the most notable point-based approaches are described below in order of their introduction.

With 3D ShapeNets Wu et al. [64] introduced the first 3D deep learning model for shape completion; the model uses a single-view RGB-D image and converts the object into a voxel grid with the probability distribution of binary variables. To achieve this, the authors proposed a convolutional deep belief network. In contrast to normal convolutional deep learning models, this one does not do any pooling in the hidden layers as it would

lead to greater uncertainty for shape completion. For training purposes, the authors introduced ModelNet, a 3D CAD model dataset.

Since 3D ShapeNets is trained on synthetic CAD models, it does not work well with real-world data. Furthermore, it does not utilize the valuable color information from RGB-D images. To exploit this unused potential, Nguyen et al. [41] built on the work of 3D ShapeNets and proposed a Markov Random Field model for the representation of 3D objects. The priors that capture the local geometric information are learned by a convolutional deep belief network, and the respective completion reparation of an object is formulated as a maximum a posteriori estimation.

A similar method to 3D ShapeNets was introduced by Dai et al. [13]: a 3D-encoder predictor network (3D-EPN). But instead of using an occupancy grid, these authors used a TSDF for the input and a distance field (DF) for the ground truth. The architecture of 3D-EPN is shown in Figure 11. The 3D deep convolutional network uses the partial input TSDF and predicts the distance field. This is accomplished by first compressing the input TSDF with the 3D encoder and using a 3D-CNN shape classifier to predict the semantic class in the hidden space volume. Two fully connected layers then embed the semantic information of the scan into the latent space. Afterwards, the predicter uses 3D up-convolutions to output the distance field. Skip connections are embedded between the encoder and decoder to counteract the information loss.

.

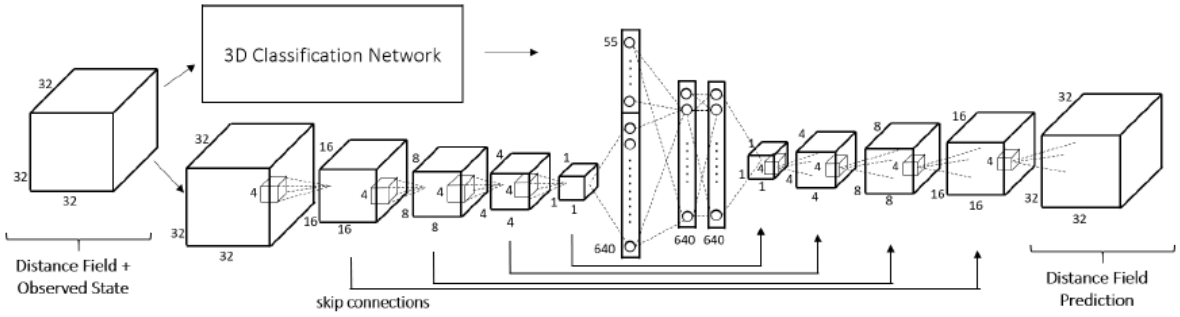

Figure 11: Network architecture of 3D Encoder-Predictor Network [13]

To generate a higher resolution output (256^3 grid) compared to 3D-EPN (32^3 grid) Han et al. [23] proposed to learn a local encoder-predicter network to perform patch-level surface inference. Figure 12 visualizes their approach. The architecture includes two networks trained in conjunction with each other. The global structure inference network predicts the global structure of the shape, while the local geometry refinement network

further refines local patches in an iterative manner under the guidance of the global structure inference network.

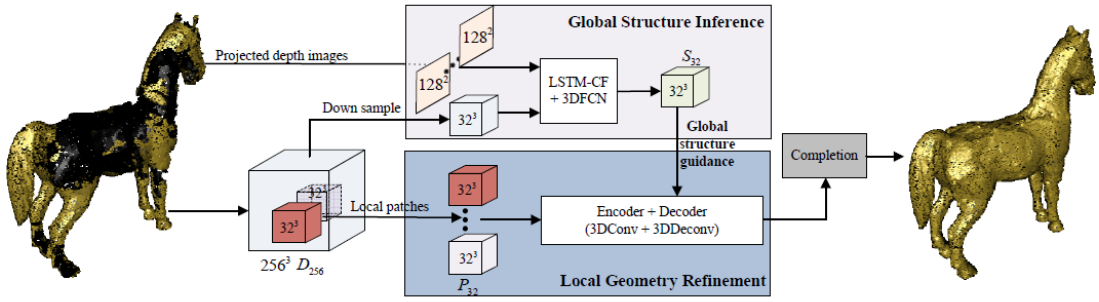

Figure 12: Architecture of the high-resolution shape completion method of Han et al. [23]

One of the pioneer works addressing point-based methods is that of Yuan et al [66]. Their proposed point completion network (PCN) formulates the solution as a generative model with an encoder-decoder network in a coarse-to-fine fashion, as presented in Figure 13. The input point cloud is compressed to a feature vector *v* by the encoder. This feature vector is then used by the decoder to first produce a coarse point cloud and then a detailed output point cloud. To extract the global feature from a point cloud, the encoder uses two PointNet layers and two shared multilayer perceptrons. PCN outperforms the volumetric approach of 3D-EPN.

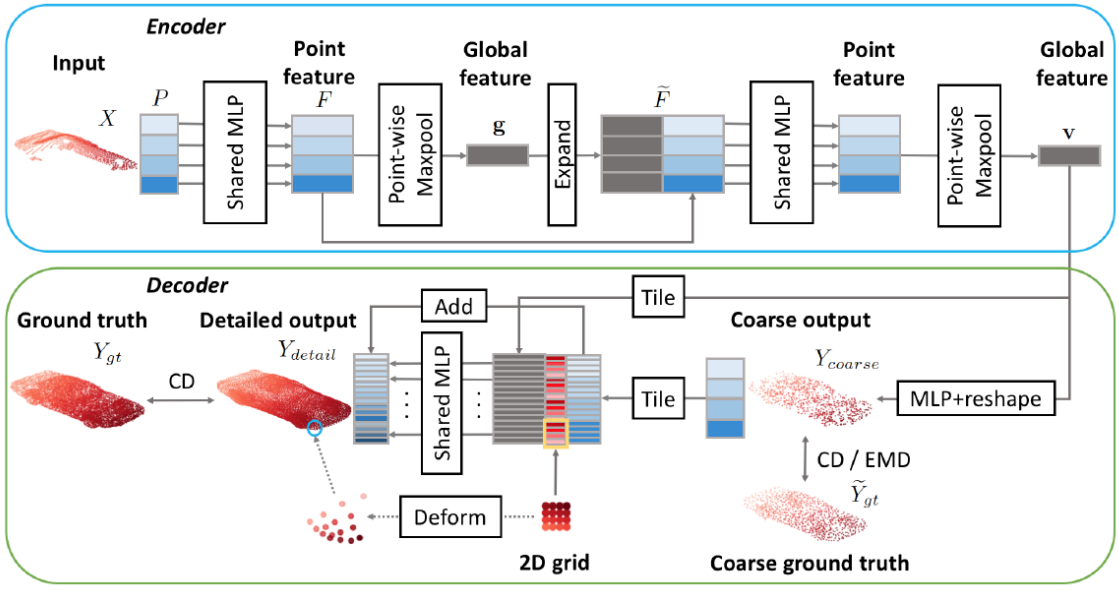

Figure 13: Point Completion Network: Architecture [66]

TopNet [54] improved on the structure of the PCN [66] decoder with its rooted tree architecture.

Sarmad et al. [48] introduced the first approach, combining reinforcement learning with a generative adversarial network (GAN) to complete a point cloud in the work RL-GAN-Net. As illustrated in Figure 14, RL-GAN-Net consists of three modules: the autoencoder (shown in green), a latent-space generative adversarial network (shown in blue) and a reinforcement learning agent (shown in gray). In the first step, the autoencoder—composed of an encoder and decoder—is trained. The encoder transfers the noisy point cloud into a noisy global feature vector (shown in yellow). The reinforcement learning agent then choses the right seed for the GAN generator. The generator then creates a clean global feature vector, and the decoder predicts the completed point cloud.

.

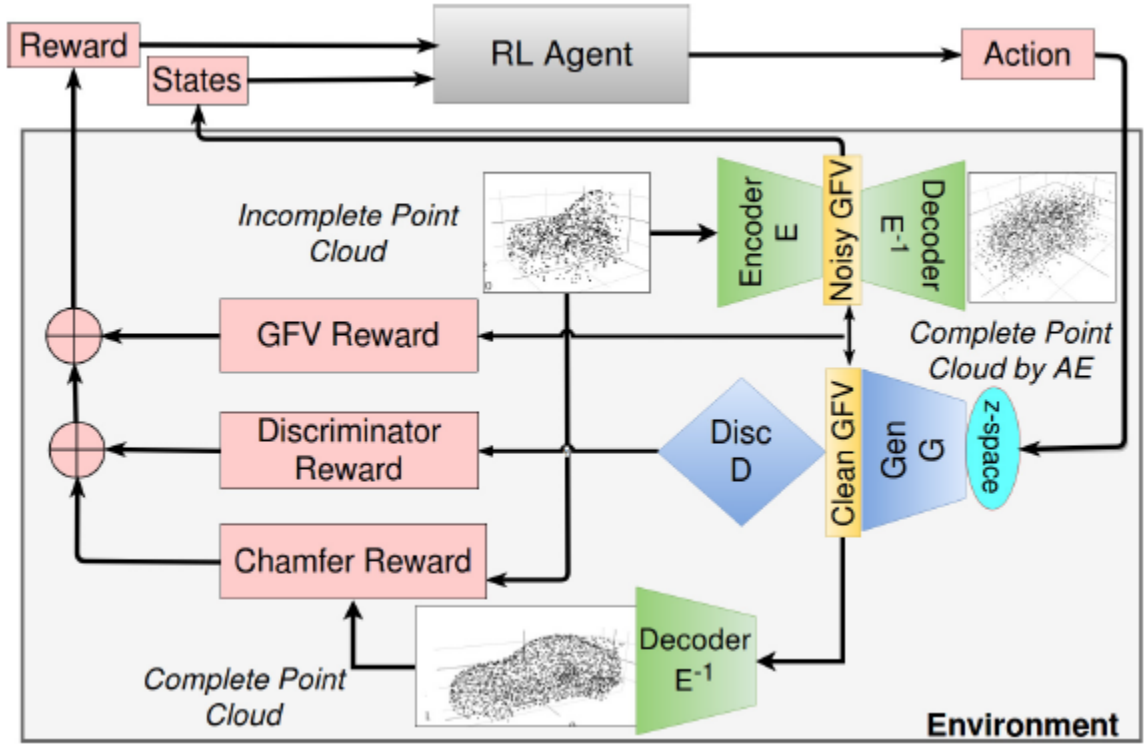

Figure 14: Training RL-GAN-Net [48]

5.2 Scene Completion and Semantic Scene Completion

This section focuses on recent developments in scene completion (SC) and semantic scene completion (SSC). SC has been achieved to a small degree by a variety of small hole-filling algorithms, but in the last few years, new deep learning methods have become available, and large datasets have emerged, representing a significant step forward. Many SC methods also include segmentation, and since the architectures of SC and SSC are very similar, they are summarized together in this section. Most methods use a volumetric approach, which means they use a grid as input, for example an occupancy grid or a truncated signed distance field, and the architecture often consists of a 3D convolutional neural network (3D CNN). Another popular approach is the view-volumetric network, first introduced by Guo et al.[22]. This architecture uses either 2D input data or compresses 3D input data into 2D data. The most common method is then to use a 2D CNN in combination with a 3D CNN. This can be advantageous since 2D CNNs are less computationally demanding. The third category is the point-based approach, which, in contrast to OC, has not been very commonly used up to now. The losses are mostly cross entropy; in the case of the occupancy grid, binary cross-entropy loss is quite common. In the following, the most relevant methods are described in greater detail to provide an overview of the existing approaches.

The first researcher to combine predicting the volumetric occupancy and object category was Song in 2016 with SSCNet [52]. He referred to the combination of scene completion and scene labeling as semantic scene completion. SSCNet uses a 3D convolutional network. As input, it takes a single depth image and constructs a view independent TSDF by calculating the distance to the nearest point on the surface. The network architecture is displayed in Figure 15. It consists of 3D convolutional layers for learning the local geometry representation. To reduce the resolution of the input to a quarter, convolutional layers with stride and pooling layers are employed. To gather the higher interobject contextual information afterwards, a dilation-based module is utilized. Then, the data is processed by two convolutional layers and one voxel-wise softmax layer.

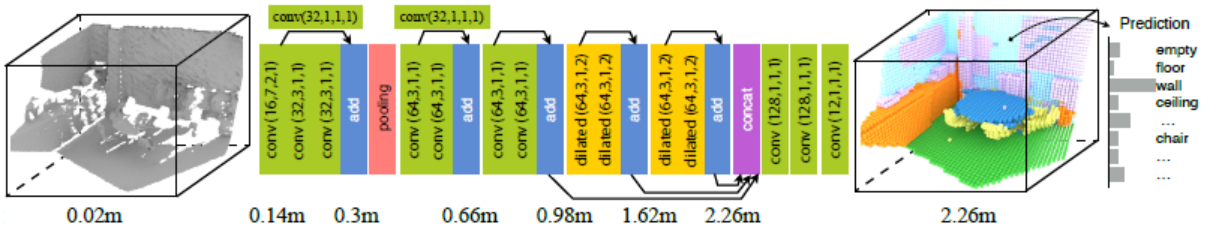

Figure 15: Architecture SSCNet [52]

Based on SSCNet, the work of Guedes [20] additionally uses the color information of RGB-D images instead of using only the depth information. This additional color information did not show better results when evaluating it using the NYUv2 dataset.

Another approach fusing the RGB and depth information is AM²FNet [4]. Figure 16 gives an overview of the structure of this network. In the depth branch, the information is converted into a flipped TSDF. At the same time, the RGB image is projected onto three volumetric data representations. Each representation contains areas of red, green and blue. Then, the integration module fuses the representations together, and the refinement module integrates high-level features into low-level features to retain more local details.

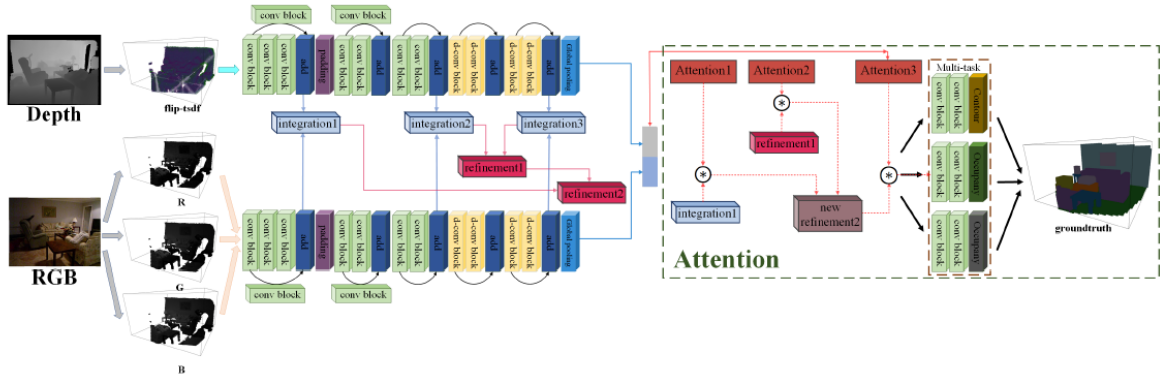

Figure 16: AM²FNet architecture [4]

A different idea is approached by Guo et al. with VVNet [22], a view-volume convolutional neural network. To reduce computational costs, the geometric features from a depth image are obtained with a 2D-view CNN. Afterwards, the features are projected into a 3D volume and processed with 3D CNN to learn the context information.

A similar approach is taken by SATNet [49], which first performs the semantic segmentation with a 2D convolutional network and then projects the 2D semantic features onto their corresponding 3D spatial positions. The approach completes the 3D scene using

an architecture consisting of two residual blocks and performing two Atrous spatial pyramid poolings and two 1*1 convolutions. The greatest benefit is that the completion process can take advantage of the semantic scene surface. Figure 17 gives an overview of the general structure, consisting of an encoder-decoder architecture for 2D segmentation, a 2D-3D reprojection layer for mapping the 2D semantic scene into a 3D volumetric voxel grid and a 3D convolutional network to complete the scene.

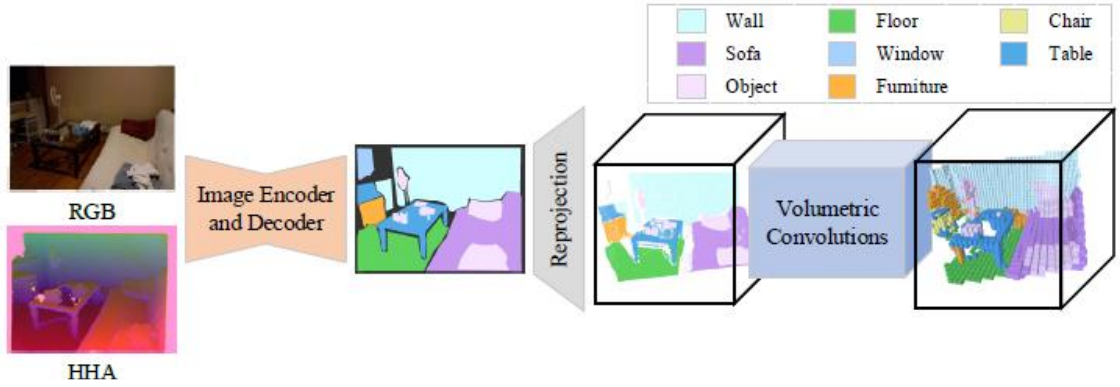

Figure 17: SATNet [49]

Garbade et al. [18] used a colored voxel grid in which the RGB image is first segmented with a 2D CNN. Each pixel in the depth map corresponds to a pixel in the 2D semantic segmentation map. Due to this, every class pixel can be projected into the 3D volume at the location of its correlating depth value. This gives an incomplete 3D semantic tensor that allocates to every surface voxel its corresponding class label. Afterwards 3D CNN completes the scene.

An approach to obtaining 3D SSC simultaneously from the 2D segmentation data was proposed using AMFNet [33], which has a similar structure to SATNet, but the SSC has a two-branch structure as shown in Figure 18: a 3D guidance branch and a 3D SSC branch. After the 2D segmentation, which involves the encoder-decoder structure and the 2D-3D projection layer that outputs a 3D semantic voxel grid, the 3D volume network predicts the complete scene.

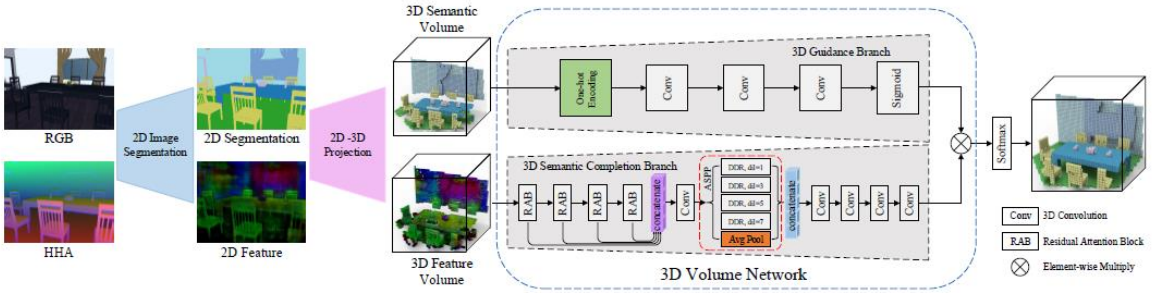

Figure 18: Architecture of AMFNet [33]

To reduce memory cost and increase inference efficiency, Wang et al. [60] introduced a semantic reconstruction method that uses octrees. The approach performs in a coarse-to-fine fashion. It predicts a semantic class for each voxel in every octree level and then decides which voxel is to be divided further to define the reconstruction. In this way, a reconstruction in a higher resolution compared to binary voxel grids and TSDFs is possible.

Another octree-based approach uses the O-CNN [57, 59] framework and was proposed in [58]. The network structure is displayed in Figure 19. Here the network has a U-Net design consisting of two deep residual networks for encoding and decoding.

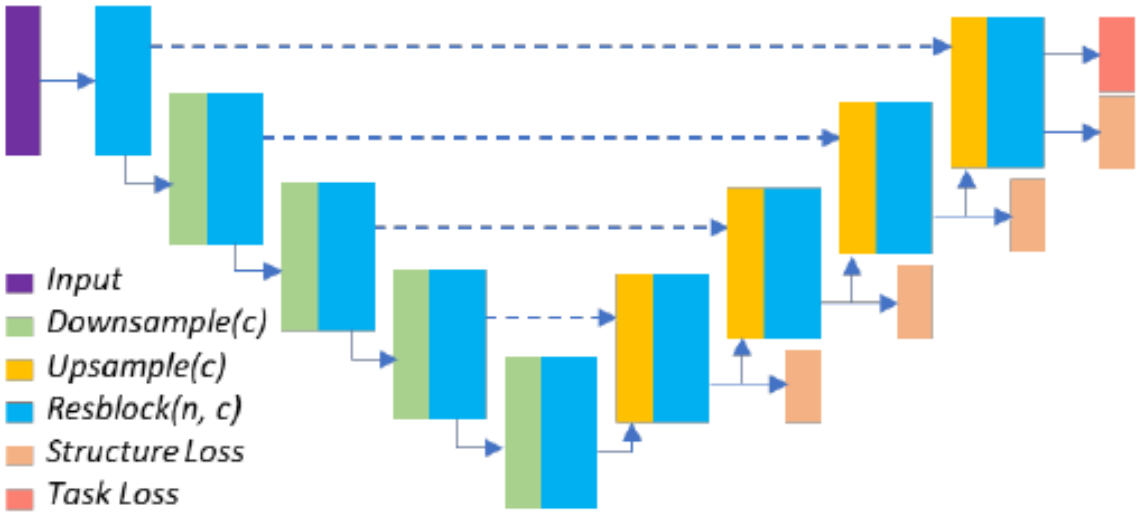

Figure 19: U-Net design with octrees. [59]

Since the network completes the scene further, the octree resolutions differ in some areas between input and output. To combat this, the authors proposed a novel output-guided skip connection, as shown in Figure 20, that adds skip connections between the created octree node and its corresponding octree node in the input. In Figure 20

(a) is displayed the input shape with additional noise in the top right corner. The target shape is represented by (b), and there are three skip connections: l_1, l_2 and l_3. The procedure can be summed up in four steps as shown in the figure. In step one, the node's status (empty, nonempty) is defined. In step two, the octree is categorized by the node status. In the third step, the features of (a) are multiplied with the map of (e). In the last step, (f) and (b) are summed up, and the result is presented in (d). Evident in this example is the robustness toward noise.

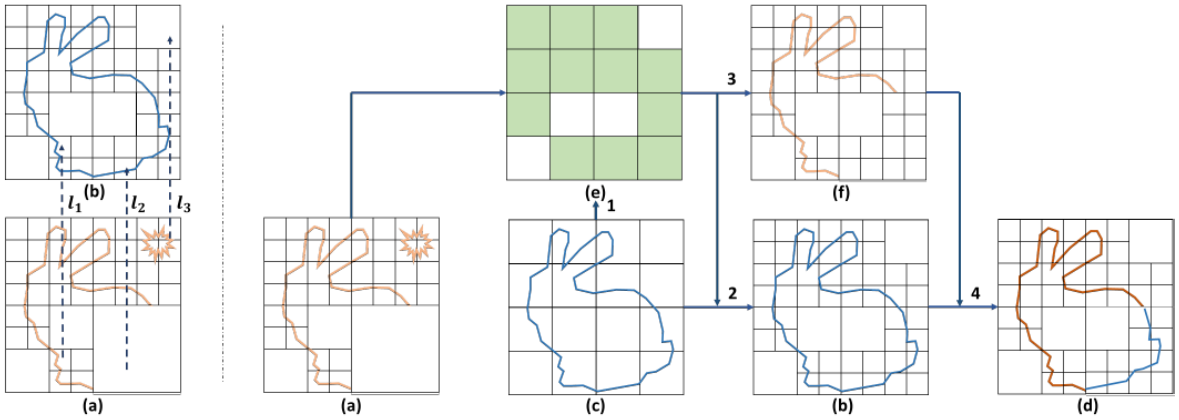

Figure 20: A octree output-guided skip connection. [59]

EdgeNet [15] represents a new end-to-end 3D CNN architecture that combines and represents information on color and depth. This contrasts to other methods that perform the 2D segmentation before training the 3D-CNN. EdgeNet use edge detection in the image. This gives a 2D binary representation of the scene which can be used to highlight objects that are hard to determine in depth maps. A picture on a wall is, for example, expected to be invisible in a depth map. EdgeNet tests three fusion schemes as shown in Figure 21: early fusion, middle fusion and late fusion.

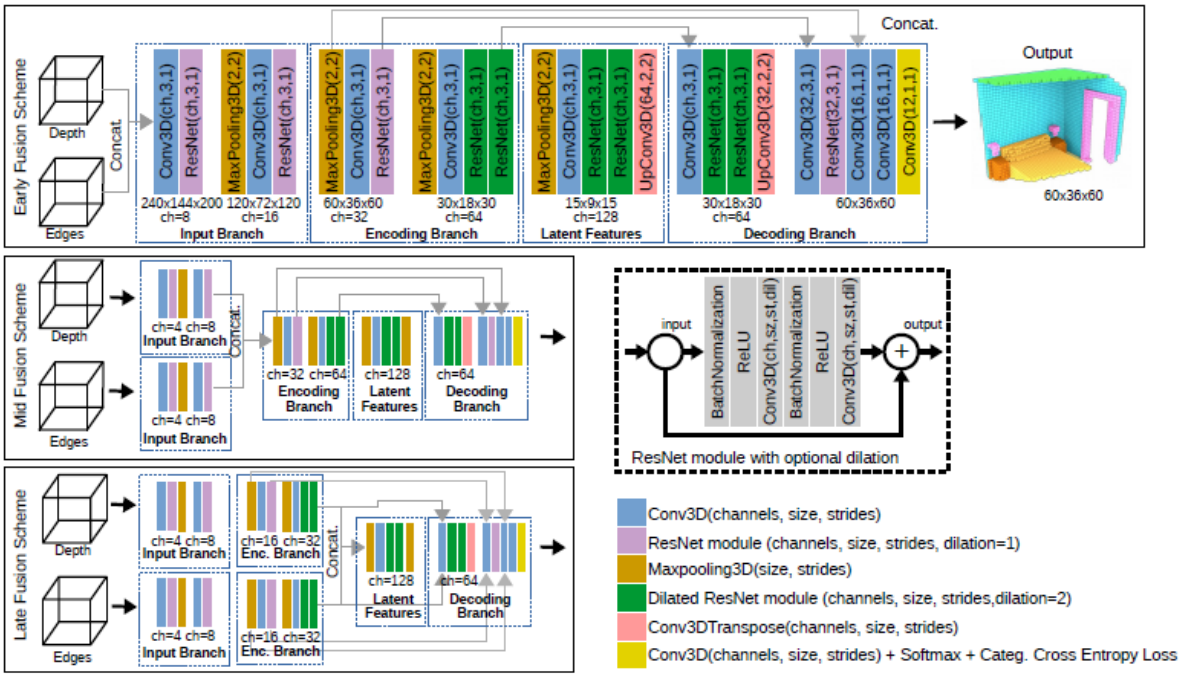

Figure 21: EdgeNet architecture and fusion schemes [15]

After testing the different fusion strategies, they concluded that a mid-level fusion strategy performs best. Since memory requirements increase as fusion occurs later, the batch size for training must be reduced. This in turn can negatively impact on learning.

Cherabier et al. [7] introduced a framework for multi-view 3D scene completion. Their network architecture consists of a three-layer approach with an encoder, unrolled primary dual optimization layers, and a decoder. The primal-dual algorithm uses variational optimization for 3D reconstruction as a lightweight regularizer. The advantage of the architecture is significantly reduced memory and computational need compared to a high-capacity 3D CNN. Figure 22 gives an overview of the architecture.

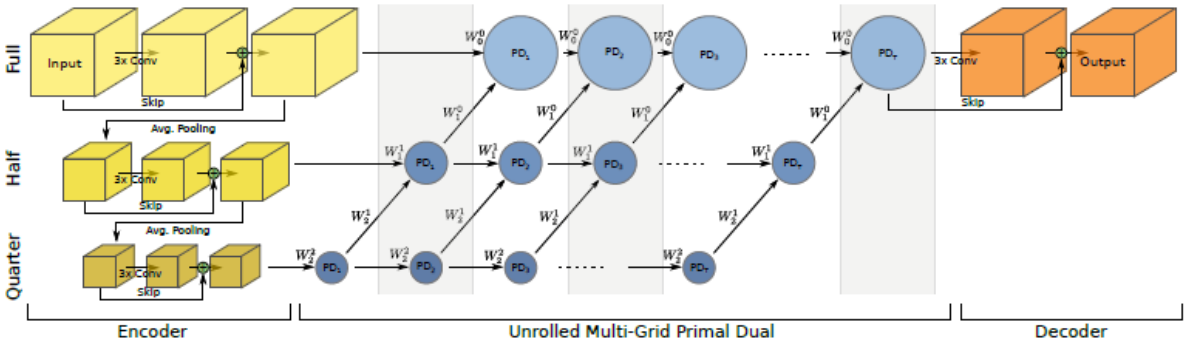

Figure 22: Architecture with primal dual optimization [7]

Another work addressing the problem of the cubical growth of the memory and computation needs of 3D CNN is that of Zang et al. [67]. They used spatial group convolution in which voxels are grouped together, and then 3D sparse convolution was performed on every group. The method cuts the computational effort considerably with only a small loss of accuracy since the voxels are only accounted for during the convolution.

Wang et al. [61] proposed an adversarial learning-based method for SSC. They used depth images directly as input and trained the depth information into the 3D volumetric space with semantic labels using two discriminators. One discriminator compares the reconstruction with the ground truth to optimize the overall architecture. The other discriminator optimizes the learned latent features.

Chen et al. [6] tried to improve the generative adversarial network by using a TSDF as input since Wang et al. discovered that the encoders discard too much information to match the different representations, resulting in a substantial loss of information. This results because the encoder of the depth image differs from the encoder of the voxelized ground truth.

Since 3D convolution is so computationally expensive, Li et al. [31] introduced an anisotropic convolutional network. Compared to normal 3D convolution, the introduced anisotropic convolution in AIC-Net is less computationally demanding and has a higher parameter efficiency. The proposed anisotropic convolution module allows 3D kernels with changing sizes since it modifies voxel-wisely to the dimensional anisotropy property. Figure 23 shows the structure of an anisotropic convolution module.

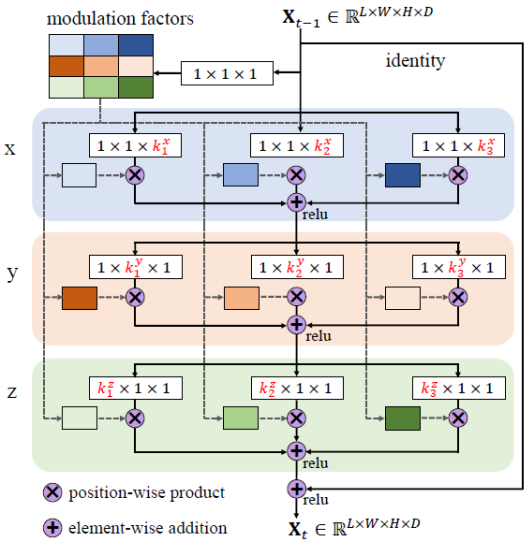

Figure 23: Anisotropic convolution [31]

With SPCNet [39], an approach to point clouds and RGB pictures was proposed to reduce memory consumption compared to grid-based solutions. SPCNet consists of two modules: an observer point encoder and an observed-to-occluded point encoder. The encoder projects the point cloud features stepwise into lower point resolution as shown in the Figure 24. Afterwards, the decoder projects the features to the occluded representation points and finally assigns semantic labels to every class. The texture of the RGB images is segmented in 2D, and afterwards the semantic label is projected to the corresponding point. This approach is promising in terms of efficiency but does not offer more accurate results than grid-based methods.

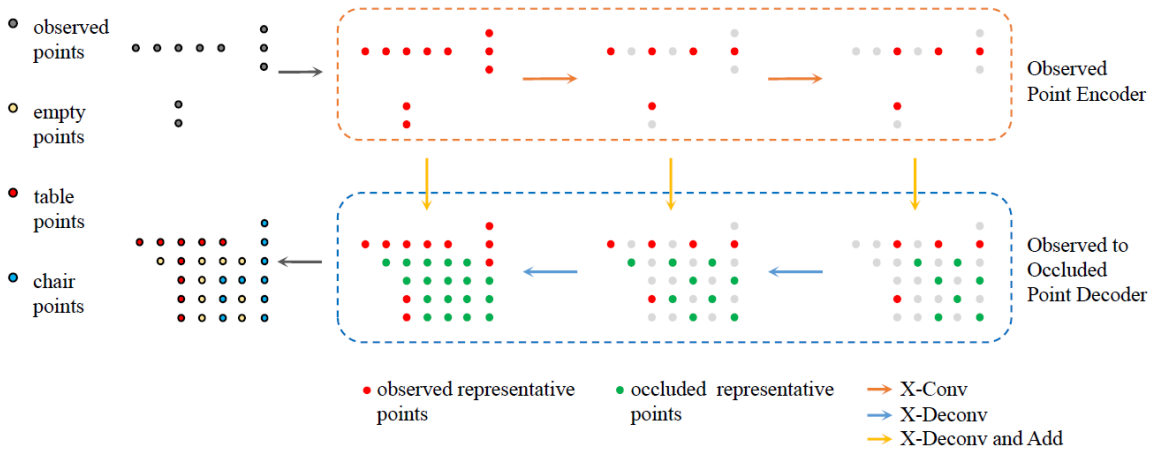

Figure 24: SPCNet architecture [39]

6 Methodology

The evaluations presented in Chapter 5 show that all the publicly available methods for 3D SC are based on a volumetric approach. Most methods use a TSDF as input since it has the greatest descriptive potential in characterizing a scene. Thus, better results can be achieved using a TSDF. Taking this into account, the goal of this thesis is to transform a point cloud into a TSDF and then run it on an existing SC algorithm.

Most SC algorithms are trained on synthetic data as it is difficult to obtain complete ground truth using real-world data. However, applying a model trained on synthetic data to real-world scans has its limitations and leads to inaccurate results. To overcome these limitations, this thesis uses an algorithm introduced by Dai et al. [12]. Their sparse generative neural network (SG-NN) for self-supervised scene completion of RGB-D scans is fully self-supervised and can be trained on real-world scans. It achieves this by removing information from a real 3D scan, making the scan less complete. In this way, the network can be trained on the differences between the less complete input scan and the real scan. After being trained, the algorithm can complete the scene even further than the real scan, thereby eliminating the gaps created by occlusions. Figure 25 shows the three stages of completion: on the left, the less complete input scan; in the middle, the real scan (also called the target scan); and on the right, the completed prediction scan.

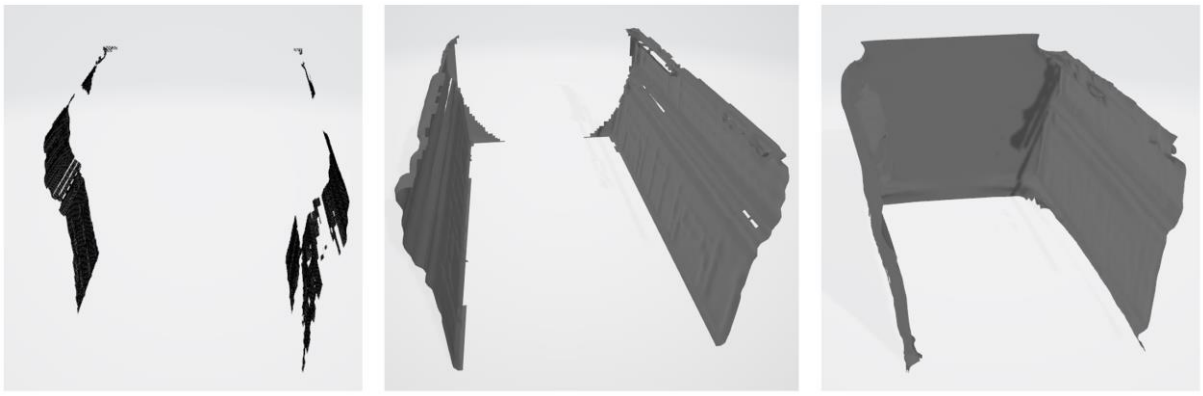

Figure 25: Input, Target and Prediction Scan

The first section in this chapter explains the architecture of an SG-NN. The next two sections describe two approaches to implementing an SG-NN on a point cloud. In the last section, data collection with a LiDAR sensor is described.

6.1 Baseline Architecture: Sparse Generative Neural Network

This section describes the SG-NN network. The input is explained in detail as this information is essential for the creation of TSDFs from point clouds described in Section 6.2. The code was developed using Pytorch 1.1.0 and Python 2.7, and it was updated by the author to run with Pytorch 1.8.1 and Python 3.8.

6.1.1 Input

SG-NN uses a truncated signed distance field (TSDF) as input and output. TSDFs are generated through volumetric fusion from RGB-D scans. The RGB-D scans were retrieved form the Matterport3D [2] dataset. The RGB-D camera system was mounted on a stative and consisted of three RGB-D cameras pointing slightly upwards, horizontally, and slightly downwards. The camera system was rotated on its vertical axis to capture six images; in total, there were 18 RGB-D images in a panorama from one camera standpoint. One RGB-D image had a resolution of 1280 × 1024. For an entire room, a set of panoramas were taken with an average distance of 2.25 m. In Figure 26, the different camera standpoints for a scene are shown as green dots. For every camera frame, 6-DoF camera poses are determined; 6-DoF indicates three domains of freedom in translation and three domains of freedom in rotation along the x, y and z axes. The camera poses are globally registered. In the Matterport3D dataset, the RGB-D data of complete levels with many different rooms is saved in one sequence, as can be seen in Figure 26. Since only one room is required at a time for training in SG-NN, only the images with camera positions in the same room are used for volumetric fusion.

Figure 26: Camera poses are on average 2,25 meters apart in the Matterport3D dataset [2].

SG-NN uses the fusion process introduced by Curless and Levoy [10] in 1996. As discussed in Section 2.5 a SDF stores the signed distance to the closest surface at each voxel. A TSDF is a SDF in which the voxels are truncated at a defined distance. To retrieve the distance, from a depth image Curless and Levoy approximated it, by using the distance along the sight of the camera. This is referred to in the literature as a projective SDF or TSDF. Figure 27 shows the projective distance when a TSDF is created from one depth image.

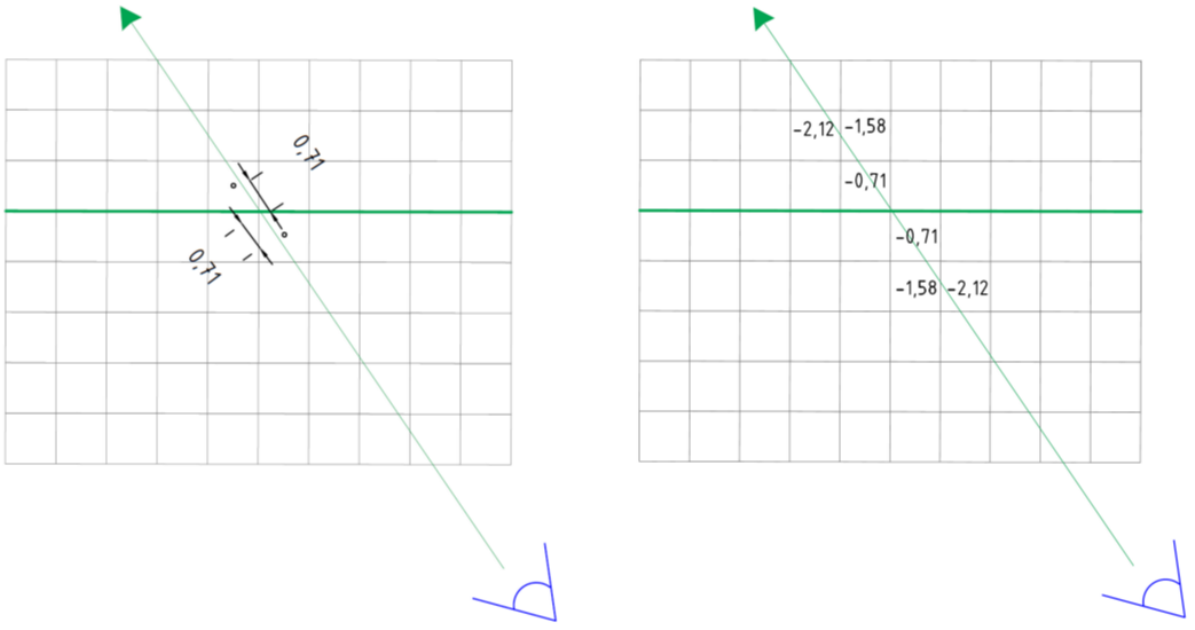

Figure 27: TSDF from one depth image

In SG-NN the TSDFs are created from multiple depth images. When using multiple depth maps the average of the distances is calculated. Figure 28 and Figure 29 visualizes this in a simplified way.

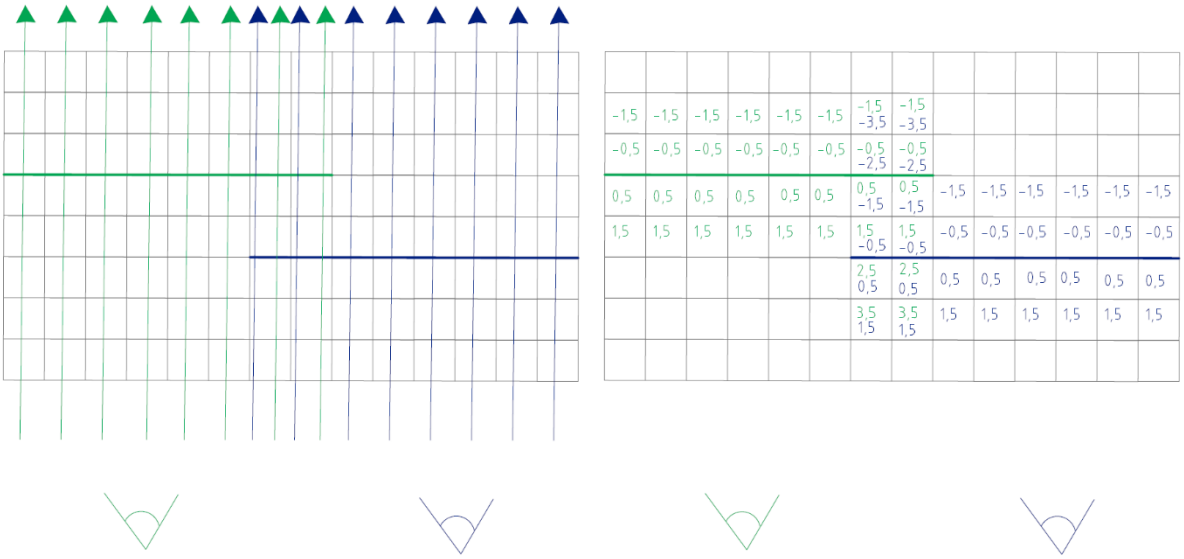

Figure 28: Two TSDFs from two depth images

Figure 10 shows the calculation of the distance for both images. For each voxel with two distances, in these two images, the average distance is calculated. This can be seen in Figure 29.

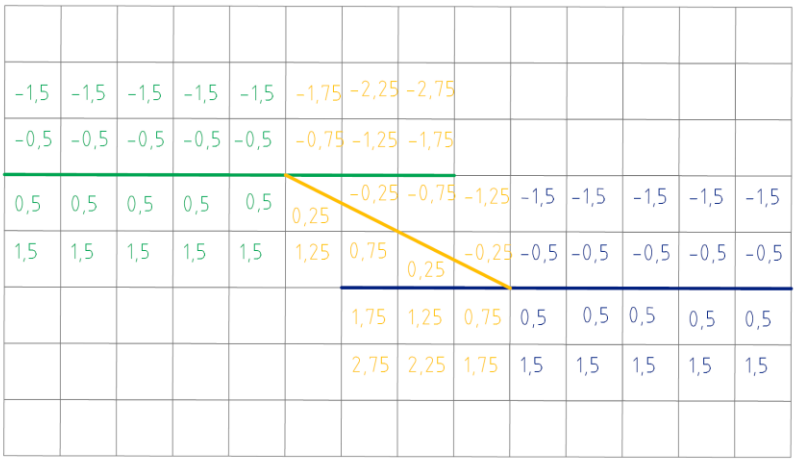

Figure 29: Two depth images averaged into one TSDF

The calculation of the fused distance, depicted in yellow in Figure 29, uses the following equations (6.1) and (6.2).

$$D(x) = \frac{\sum w_i(x)\, d_i(x)}{\sum w_i(x)} \tag{6.1}$$

$$W(x) = \sum w_i\,(x) \tag{6.2}$$

The weight of voxel *x* for camera *i* is described by $w_i(x)$. The weights are useful to down weight the SDF values behind the surface. This is helpful since in front of the surface the space is known, but behind the surface there is no information about the space. The distance along the ray, depicted in the Figure 28 with green and blue arrows., from voxel *x* for camera *i* is represented by $d_i(x)$. The fused weight and distance are described by $D(x)$ and $W(x)$.

SG-NN uses a TSDF with a voxel size of 2 cm and a truncation of 3 voxels. As described at the beginning of Chapter 6, the input consists of the real scan S_{target} and a less complete input scan S_{input}, as shown in Figure 25. The scan S_{input} is created by using approximately 50% of the frames for the volumetric fusion, and the scan S_{target} is obtained using all the frames. To train the model more efficiently, the scans are cropped to 64 × 64 × 128 voxels.

6.1.2 Architecture

The architecture is an encoder-decoder convolutional neural network and works in a cosrse-to-fine fashion generating a sparse TSDF prediction. Figure 30 visualizes the architecture and in Figure 32 the architecture is shown in detail.

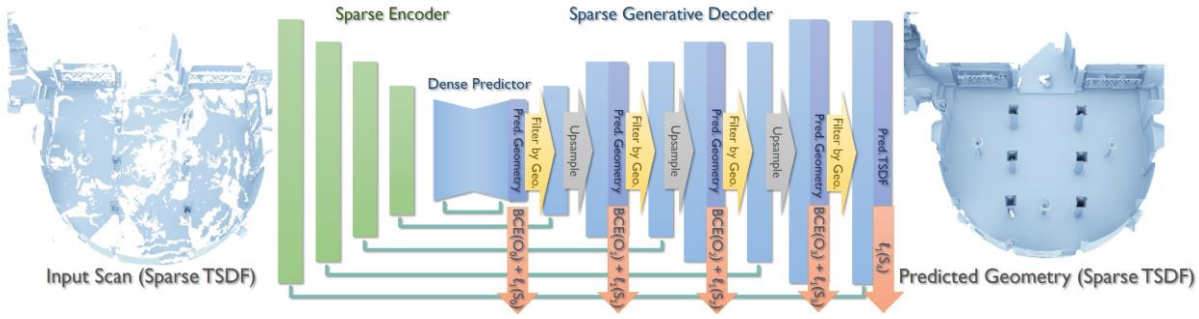

Figure 30: SG-NN architecture [12]

In the first step, the input scan, represented by a sparse TSDF, is encoded with several 3D sparse convolutions, depicted in Figure 32 as “SparseEncoder”. With each set of 3D convolutions, the spatial dimensions are reduced by half. After the encoding the features are transformed into a dense occupancy grid (in Figure 32 SparseToDense), to allow for a better prediction of the complete scene at a coarse level. The next step, shown in Figure 32 as DenseGenerator, predicts the geometry of the complete scene at a low resolution, by using several dense 3D convolutions. The output is a feature map F_0, the predicted coarse occupancy O_0 and the predicted TSDF S_0. Next, a sparse representation is calculated based on the occupancy O_0. These representations of F_0, O_0 and S_0 are then decoded using the SparseGenerator. In each step, the scene is predicted with a higher resolution. This hierarchical process of predicting occupancy and TSDF and continuously improving resolution is visualized in Figure 31.

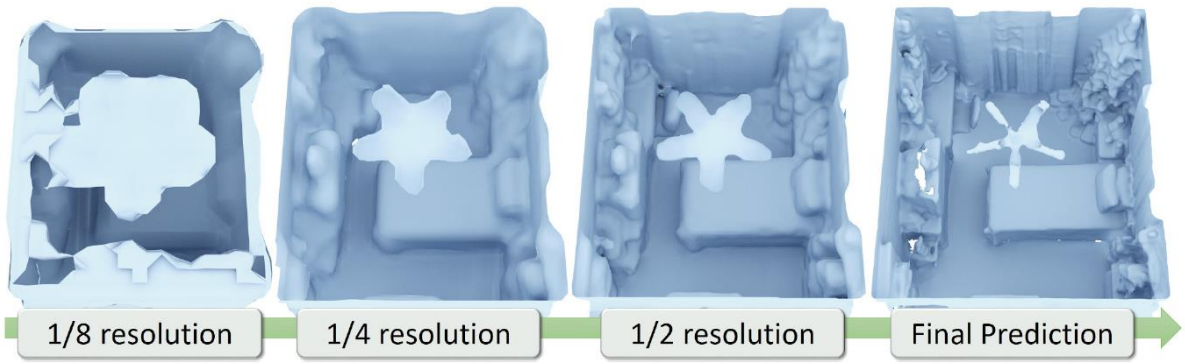

Figure 31: SG-NN: coarse-to-fine [12]

In the last step the final TSDF is predicted by module SurfacePrediction, as shown in Figure 32. As input the last predicted F_n, O_n and S_n are used. The network also works with sparse skip connections between the encoder and decoder. The features that are the same in the target and input destination are connected and when a target destination is not present in the destination zero feature vectors are used.

SG-NN uses a binary cross entropy loss in every level for the occupancy O_n and a l_1 loss with the target TSDF values.

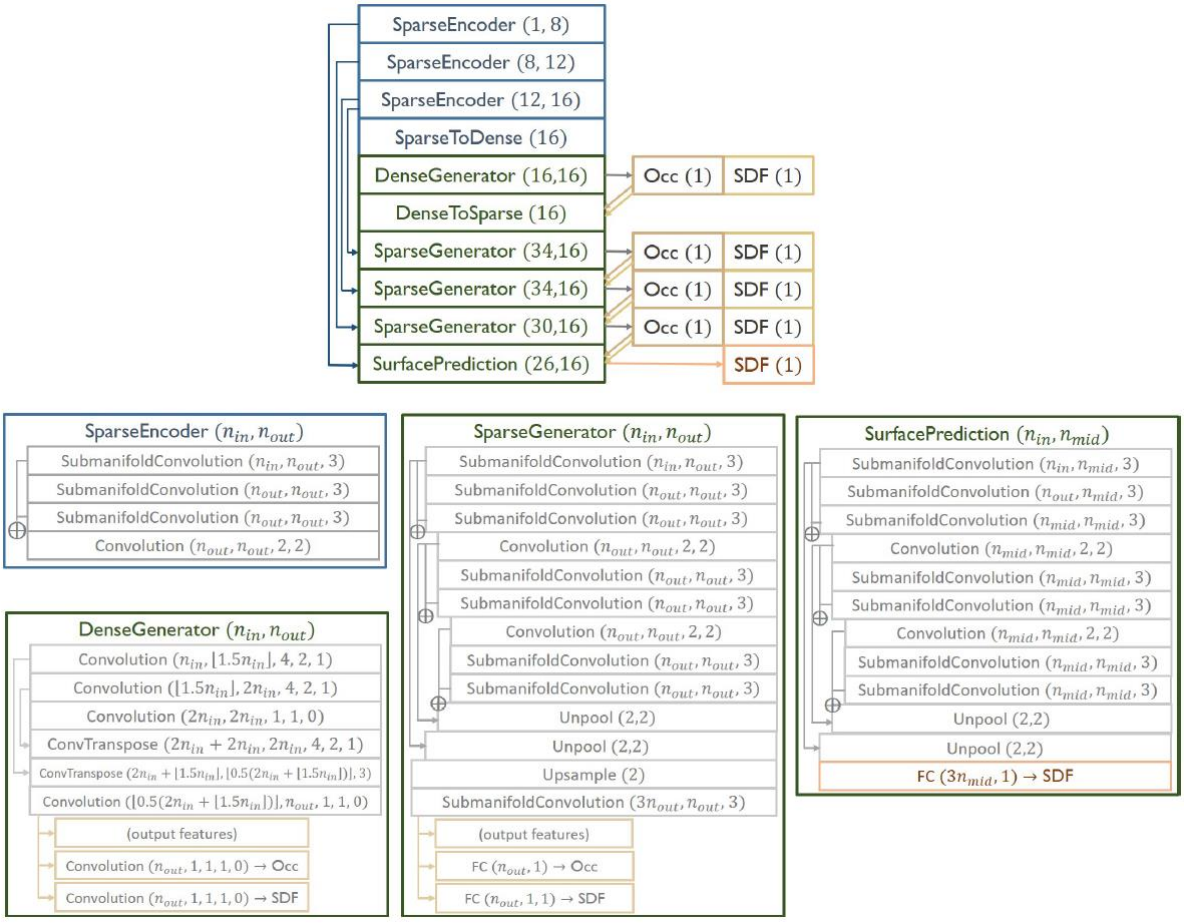

Figure 32: Detailed architecture of SG-NN [12]

In Figure 32, the architecture of SG-NN is shown in detail with its convolutional parameters given as nf in, nf out, kernel size, stride and padding, stride and padding defaulting to 1 and 0, respectively. An arrow indicates concatenation and a ⊕ addition.

6.2 Point Cloud to TSDF

In this section, a novel method for transforming a point cloud into a TSDF is proposed. The section is divided into three subsections. First, the requirements and challenges are explained. Then, in the second and third subsections, the algorithm is described.

6.2.1 Requirements and challenges

The first step in developing an algorithm from point cloud to a TSDF was to find the data representation of a TSDF and a .ply file. Table 3 shows the data representation of a TSDF file in SG-NN. A TSDF is defined by the number of voxels and a 3D array comprising the *x*, *y* and *z* coordinates as unsigned integers. The distance is stored as a float. The point cloud is defined by the number of points, and the coordinates of the points are saved as floats.

Table 3: Data representation of a TSDF file in SG-NN

Number of bytes	Data type	Meaning
Header data		
8	unsigned long long	dimx
8	unsigned long long	dimy
8	unsigned long long	dimz
4	float	Voxelsize
64 = 4 x 4 x 4	array 4 x 4 float	world2grid
Input data		
8	unsigned long long	num = number of voxels
4 x num x 3	unsigned int	input_locs = x,y,z coordinates of the voxels
4 x num	float	input_sdfs: distance for every voxel

To obtain more insight into the sparsity of the TSDF in SG-NN a small algorithm was written to visualize all voxels of a training TSDF as points. Figure 33 shows the input TSDF from a bathroom. On the left all negative voxels are represented as points and on the right all voxels are visible.

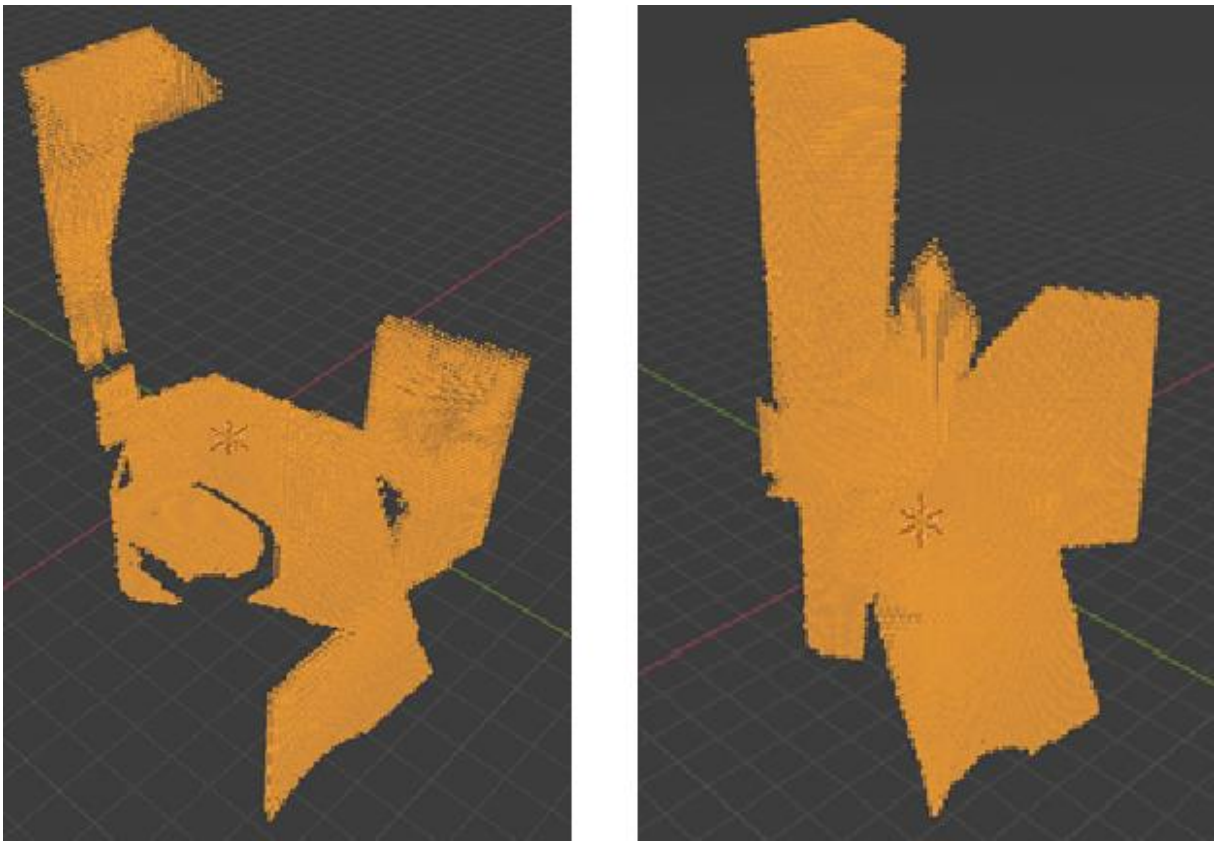

Figure 33: Voxels of a TSDF represented as points.

From the left image in Figure 33, it is evident that the negative TSDF values are always behind the scanned surface and are truncated after three voxels. The indication from the right image is that there are more positive TSDF values than untruncated SDF values. However, since the truncated voxels do not provide any additional important information, it is better to create a TSDF with only the untruncated voxels for efficiency reasons. Theoretically, the truncated voxels can be easily added if needed.

From the previous descriptions, it can be concluded that generating the sign of a TSDF is the most difficult challenge. Unlike deep images, a point cloud gives no indication of the location of a surface and which points are adjacent to each other. Above all, it is not possible to identify the front and back of a surface. However, since point clouds are usually scanned from a viewpoint in the center of a space (the coordinate origin), this information can be used to determine the sign. If the point cloud is composed of several point clouds, using point cloud registration and the origin coordinates of the scan points is no longer possible; it is only possible to obtain a truncated distance field (TDF). The approach for that scenario is elucidated in Section 6.3. Another point to keep in mind when designing an algorithm from a point cloud to a TSDF is that the coordinate system of the voxels contains only positive values.

6.2.2 Overview Method

In this subsection, the general idea of the proposed algorithm for generating a TSDF from a point cloud is presented. The following subsection explains the actual algorithm. Figure 34 visualizes the approach. The illustrated room is not true to scale, and the voxels are for illustrative reasons much larger than in reality. The point cloud was generated form one viewpoint *K*, which is also the coordinate origin. The positive voxels are visualized in blue and the negative voxels in white. The green line represents the occupied voxels that contain the point cloud.

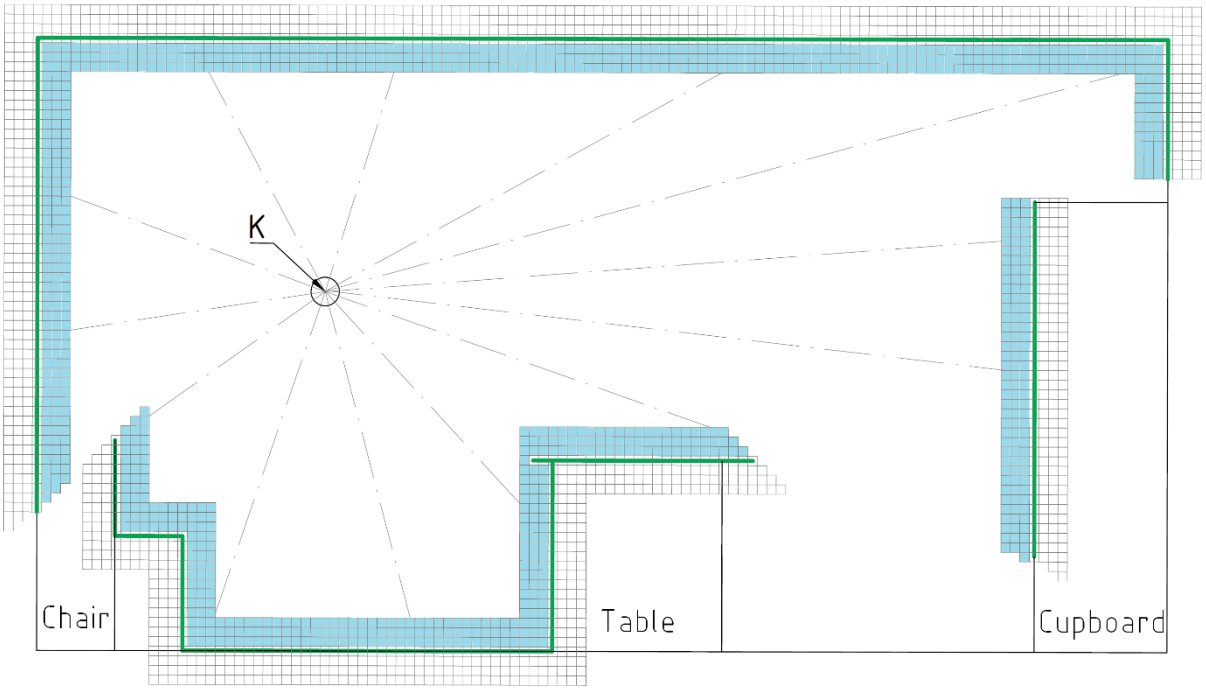

Figure 34: TSDF to point cloud

The basic idea is to project a voxel grid over the point cloud and then to save the occupied voxels. Afterwards, every voxel that lies on the line between *K* and an occupied voxel and that has a maximum distance of 3 voxels to the occupied voxel is calculated. This is illustrated in Figure 35. The occupied voxel has the distance 0, and the voxels in front of the occupied voxel on the line toward *K* have positive distances. Behind the occupied voxel, the voxels have negative distances. Euclidean distance is used in the calculation.

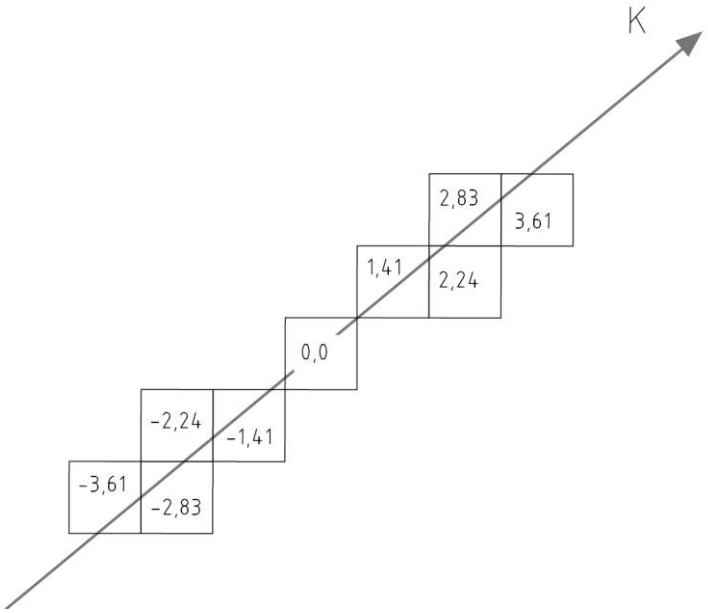

Figure 35: Occupied voxel with its adjacent signed distance voxels.

Instead of calculating the distance to the surface or in the case of a point cloud to the points, the algorithm calculates the distance to the occupied voxel center. In this way, the algorithm is more robust to noise and point density. At the same time, the level of detail is reduced due to the voxel size of 2 cm.

6.2.3 Algorithm

This subsection describes the proposed algorithm in detail. It is possible to change the voxel size, the number of voxels until truncation, whether the input point cloud should be visualized and by how many voxels the dimension space should be increased.

In the following, the steps of the algorithm are explained.

1) The algorithm employs the Open3D library to read and down-sample the point cloud to the desired voxel size.
2) If the input of show_visualization is set to *true*, the down-sampled voxel grid is visualized using the Open3D library.
3) The down-sampled voxel grid is converted to an array. The coordinates are saved with the voxel distance, not the real distance.
4) After acquiring the voxels that are occupied, **Algorithm 1** is used for every voxel. It creates the voxels that lie on the line to *K* before and behind the occupied voxel. To find these voxels, Bresenham's line algorithm was adapted.

The occupied voxel *(loc)* is the starting voxel, and the scan point *K* (origin) is the target voxel. At first, a distance of 0 is saved for the occupied voxel in line 2. Then, the driving axis is determined by the greatest absolute between the coordinates *(dx, dy, dz)* of the starting and target voxels.

For the driving axis, the slope error is calculated using the equations in Table 4. Afterwards, the algorithm always moves one voxel forward on the driving axis and one voxel on the other axes if the slope error is greater than zero. In every step, the new voxel is saved with its accompanying Euclidean distance *(sdfs)* and the voxel that lies on the opposite side of the starting voxel with the same negative distance.

Table 4: Slope-error equations

x-axis	y-axis	z-axis
$p_{xy} = 2 \times d_y - d_x$	$p_{yx} = 2 \times d_x - d_y$	$p_{zx} = 2 \times d_x - d_z$
$p_{xz} = 2 \times d_z - d_x$	$p_{yz} = 2 \times d_z - d_y$	$p_{zy} = 2 \times d_y - d_z$

With every step, the Bresenham algorithm moves forward one voxel on the drive axis and thus skips some voxels that lie on the projection line. In Figure 35, this would be for example the voxel with the distance 2.24. Further methods to solve this are discussed in Section 7.2.

5) Since the signed distances *(sdfs)* are calculated in voxels, they are multiplied by the voxel size to be converted into meters.
6) The voxels are stored in SG-NN as unsigned integers, so the voxel grid is shifted in this step. First, the maximum and minimum voxel coordinates are calculated, and the defined additional_shift is added into the input for padding on each side. Then, all voxels are shifted by the minimum coordinates of the voxel grid.
7) The dimensions are saved from the maximum coordinates in the voxel grid.
8) The array world2grid is created and filled.
9) With all necessary information in place, a TSDF file is generated with the structure shown in Table 3.
10) The necessary known file is written, which contains all voxels of the dimension with the value zero.
11) To verify the correctness of the data, all TSDF data is also written to a text file.

Algorithm 1: Calculate voxels till truncation with distance

```
    Input:   coordinate origin: origin,
             coordinates occupied Voxel coordinates: loc,
             number of voxels till truncation: number_of_voxels_front,
    Output:  voxels: locs,
             distance of voxel: sdfs
1   Save loc[x,y,z] in locs and distance 0.0 in sdfs
2   Store loc_start[x,y,z] = loc[x,y,z]
3   Calculate the constants dx,dy,dz = |origin[x,y,z]- loc[x,y,z]|
4   If origin[x] > loc[x]              #function three times for x,y,z
5   |       xs = 1
6   Else
7   |       xs = -1
8   End
9   If dx > dy and dx > dy             #function three times for x,y,z
10  |       p1 = 2 * dy - dx
11  |       p2 = 2 * dz - dx
12  |       While |loc[x]-loc_start[x]| < number_of_voxels_front
13  |       |       loc[x] = loc[x] + xs
14  |       |       If p1 ≥ 0
15  |       |       |       loc[y] = loc[y] + ys
16  |       |       |       p1 = p1 – 2 * dx
17  |       |       End
18  |       |       If p2 ≥ 0
19  |       |       |       loc[z] = loc[z] + zs
20  |       |       |       p2 = p2 – 2 * dz
21  |       |       End
22  |       |       p1 = p1 + 2 * dy
23  |       |       p2 = p2 + 2 * dz
24  |       |       Save loc_new[x,y,z] in locs
25  |       |       Save distance sdf = loc_new to loc in sdfs
26  |       |       Save loc_opposite_new[x,y,z] in locs
27  |       |       Save negative distance sdf in sdfs
28  |       End
29  End
```

6.3 Point Cloud to TDF

The approach in Section 6.2 does not include point clouds taken from multiple viewing angles as shown in Figure 36. Thus, this section describes an approach for converting such a point cloud to a TDF.

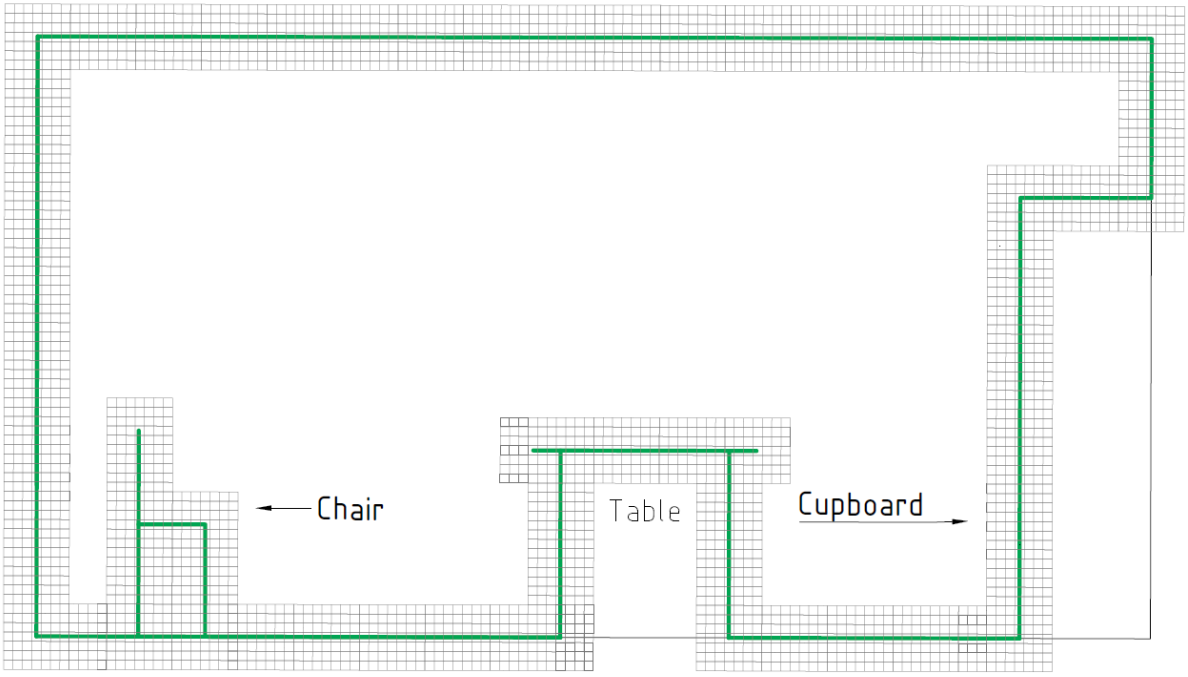

Figure 36: Truncated distance field

The calculation of a TDF has one major advantage. Using multiple point clouds combined with point cloud registration provides a more accurate ground truth for a scene. Training a SC neural network with this additional information should result in better prediction models. At the same time, there is less information about the fronts and backs of objects. At the start of this thesis, it was not certain how well the completion algorithms would work on a TDF compared to a TSDF.

The algorithm for a conversion to a TDF starts with the same steps as the proposed algorithm for the conversion from a point cloud to a TSDF. First, the point cloud is read, down-sampled and saved to an array. Here, the algorithm differs as it calculates all voxels surrounding the occupied voxel until truncation. Figure 37 shows in green two occupied voxels and the voxels with their distance until truncation at three voxels

3,0	3,0	3,0	3,0	3,0	3,0	3,0	3,0
3,0	2,8	2,2	2,0	2,0	2,2	2,8	3,0
3,0	2,2	1,4	1,0	1,0	1,4	2,2	3,0
3,0	2,0	1,0	0,0	0,0	1,0	2,0	3,0
3,0	2,2	1,4	1,0	1,0	1,4	2,2	3,0
3,0	2,8	2,2	2,0	2,0	2,2	2,8	3,0
3,0	3,0	3,0	3,0	3,0	3,0	3,0	3,0

Figure 37: Truncated distance field

6.4 Generation of point cloud dataset.

Based on the review of datasets presented in Chapter 3, it is evident that a large-scale dataset of private housing is still missing. To achieve this, the possibility of an iPhone and iPad crowdsourced dataset is explored in this thesis.

Since 2020, new iPhones and iPads from the pro product line have included LiDAR sensors since 2020. LiDAR stands for light detection and ranging and is in the category of time-of-flight sensors. The sensor sends a light impulse and measures the time it takes the light to return from the scanned object. The LiDAR sensor on the iPad is suitable for distances up to 5 m. This is sufficient for most indoor spaces but is not suited for large outdoor spaces.

There is little research yet on the accuracy of the LiDAR sensors on iPads and iPhones. Vogt et al. [37] investigated the potential of LiDAR and TrueDepth by using different Lego bricks and comparing the results with an industrial Artec Space Spider Handheld 3D scanner. The work concluded that the LiDAR sensors on iPads and iPhones are not suitable for scanning small objects such as Lego bricks. However, there has been no research on larger objects such as rooms. For this thesis, a small study was conducted (portrayed in Table 5) that included measuring distances with the LiDAR sensor on an iPhone 12 pro and comparing the measurements with the ground truth obtained by tape measurement.

Table 5: LiDAR sensor error

	Surface	Surface material	Ground truth [mm]	LiDAR distance [mm]	Error in distance [mm]	Error in percentage [%]
1	White ceiling	drywall	2534	2541	-7	0.28
2	White ceiling	drywall	2612	2633	-21	0.80
3	White wall	plaster	4234	4265	-31	0.73
4	White wall	plaster	3945	3978	-33	0.84
5	White wall	plaster	2817	2834	-17	0.60
6	White wall	plaster	3486	3505	-19	0.55
7	Cupboard	wood	565	570	-5	0.88
8	Table	wood	721	728	-7	0.97
9	Couch	fabric	2653	2667	-14	0.53
10	Carpet	fabric	1870	1880	-10	0.53

To place these errors of the LiDAR sensor into perspective, Figure 38 shows the error measured with Kinect sensors in the paper [42]. The figure illustrates that the Kinect v1 sensor showed exponential growth and the Kinect v2 sensor linear growth with higher accuracy.

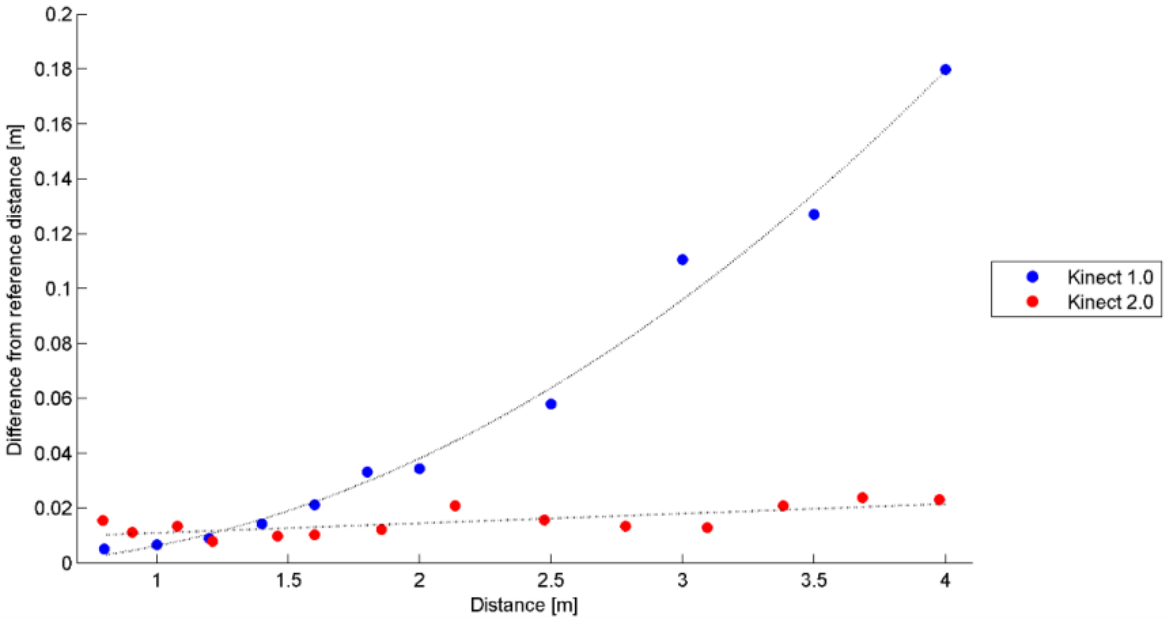

Figure 38: Error of Kinect v1 and v2 [42]

To better illustrate the errors of the Kinects sensors, Table 6 adopts the structure of Table 5 and provides the error percentages, which are calculated from the distances in Figure 38.

Table 6: Kinect error sensor

Distance	Kinect v1 Error distance [mm]	Kinect v1 Error in per-cent [%]	Kinect v2 Error distance [mm]	Kinect v2 Error in per-cent [%]
4.0	0.180	4.5	0.020	0.5
3.5	0.127	3.6	0.017	0.5
3.0	0.092	3.1	0.016	0.5
2.5	0.063	2.5	0.015	0.6
2.0	0.035	1.8	0.013	0.7
1.5	0.018	1.2	0.010	0.7

The Kinect sensor was chosen for comparison since it produced several datasets, as described in Chapter 3. The popular NYU dataset, for example, was created with a Kinect v1 sensor. Through comparing the numbers in Table 5 and To better illustrate the errors of the Kinects sensors, Table 6 adopts the structure of Table 5 and provides the error percentages, which are calculated from the distances in Figure 38.

Table 6, it becomes obvious that the accuracy of the iPhone's LiDAR sensor is suffi-cient for creating a dataset; the sensor is almost as accurate as the Kinect v2 and by far more accurate than the Kinect v1, especially when it comes to longer distances. The range of the sensors are similar, with 4.5 m for the Kinect sensors and 5 m for the LiDAR sensor.

After evaluating accuracy, the practicability of the scanning process was examined. The Polycam LiDAR 3D scanner application was used for scanning. For the output of meshes, the application supports the formats .obj, .glb, .dae, .stl and .usdz, and for point clouds the formats .dxf, .ply, .laz, .xyz and .pts are supported. The application allows a freehand scan that can be performed by walking around in a room.

During the extensive testing, some limitations became apparent. One of these was mentioned earlier in the chapter, namely that small objects such as Lego bricks cannot

be scanned accurately [37]. The other limitation results from the distortion caused by reflective objects such as mirrors, windows, glass doors and pictures with glass frames. These items must either be removed or covered up during the scanning process. In Figure 39, a distortion is shown on the right that was caused by a picture with a glass frame.

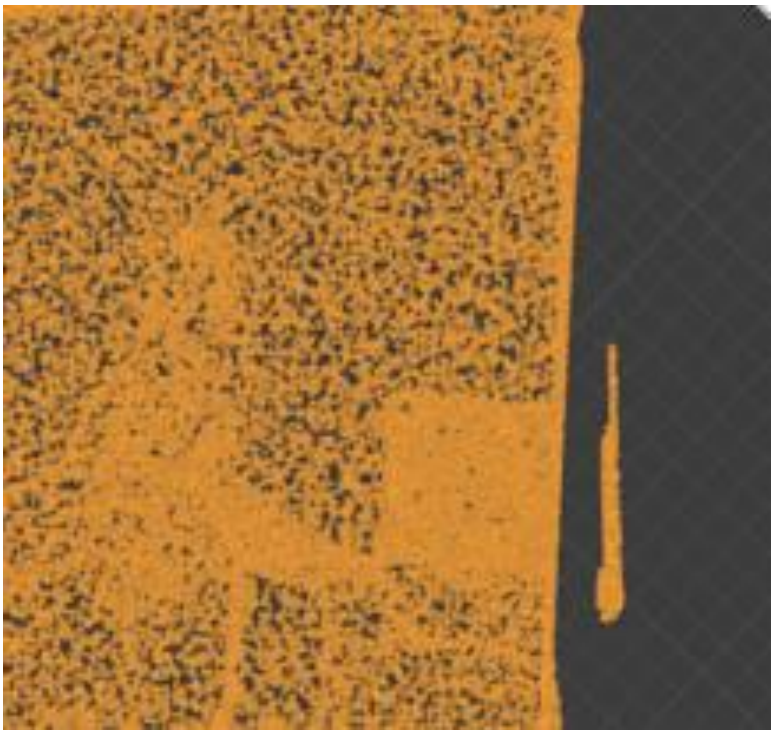

Figure 39: Distortion of a point cloud by an image through a glass frame

The original idea was that because an iPhone is mobile and small, it could be used to create a higher quality dataset. An iPhone can scan under furniture like tables and chairs, creating a more detailed dataset and more accurate ground truth for training. Unfortunately, this was not confirmed in reality. The scan registration process, which consists of aligning multiple scans on one iPhone, is not perfect. To capture all the details in a room, the scanner—in this case the iPhone—must be moved around quite a bit, thus increasing susceptibility to errors during the registration process. Especially when scanning detailed elements, for instance when placing the sensor under a table, distortions of the scan can occur. Such an error can be seen in Figure 40. After scanning a corner behind a table, the sensor experienced an error in its registration process. This caused the cupboard on the left side of Figure 40 to be distorted, thus making the scan unusable.

Figure 40: Scanning error caused through a wrong registration process.

Although a more detailed dataset with a more accurate ground truth is not possible, the results when scanning from one viewpoint are good and can be improved by using a tripod.

7 Discussion

This chapter first describes the difficulties with the SG-NN algorithm and then evaluates the results of the point cloud completion. At the end, other possible approaches are briefly discussed.

7.1 SG-NN

After debugging and adapting the algorithm of Dai et al. [12], it runs on Pytorch 1.8.1 and Python 3.8. The provided TSDFs, which were generated by RGB-D scans were tested on the trained model. In Figure 41 the input can be seen on the right and the predicted model on the left.

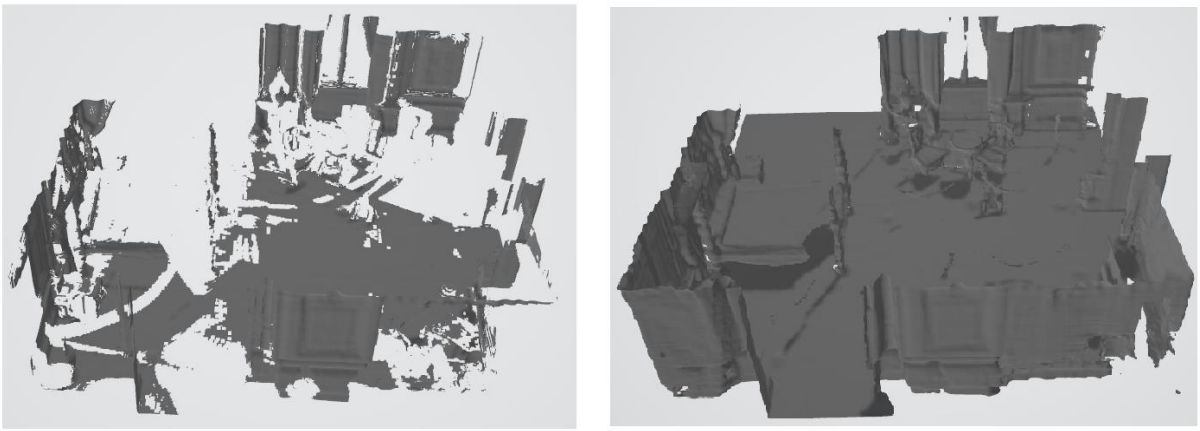

Figure 41: Input and prediction mesh

Since the source code of SG-NN is not documented, several short algorithms were written to reengineer the needed files for training. SG-NN uses an input TSDF, a target TSDF and a known file to train and complete scenes as already described in Subsection 6.1.2. The TSDF was explained in detail in Subsection 6.1.1 and will therefore not be further discussed in this chapter. Much more problematic was the known file needed to test and train the model. This file was not described in the corresponding paper nor on GitHub. The known file contains all voxels of the voxel grid, compared to a TSDF file, which only contains the voxels up until truncation. Additionally, there is an integer from 0 to 5 for every voxel. To determine how this file was created, the source code was analyzed in depth, and some small functions were written to display the binary data as text files and point clouds. In Figure 42, the voxels of the known file are represented as yellow points, and the corresponding TSDF is shown as mesh in gray. Each

of the six images contains a category or, respectively, one of the assigned numbers of the known file. The top three images show mainly but not exclusively voxels that are outside the room. The image at the bottom left displays voxels located near the surface both inside and outside the room, which are located near the surface. The pictures in the middle and bottom right illustrate the rest of the voxel grid.

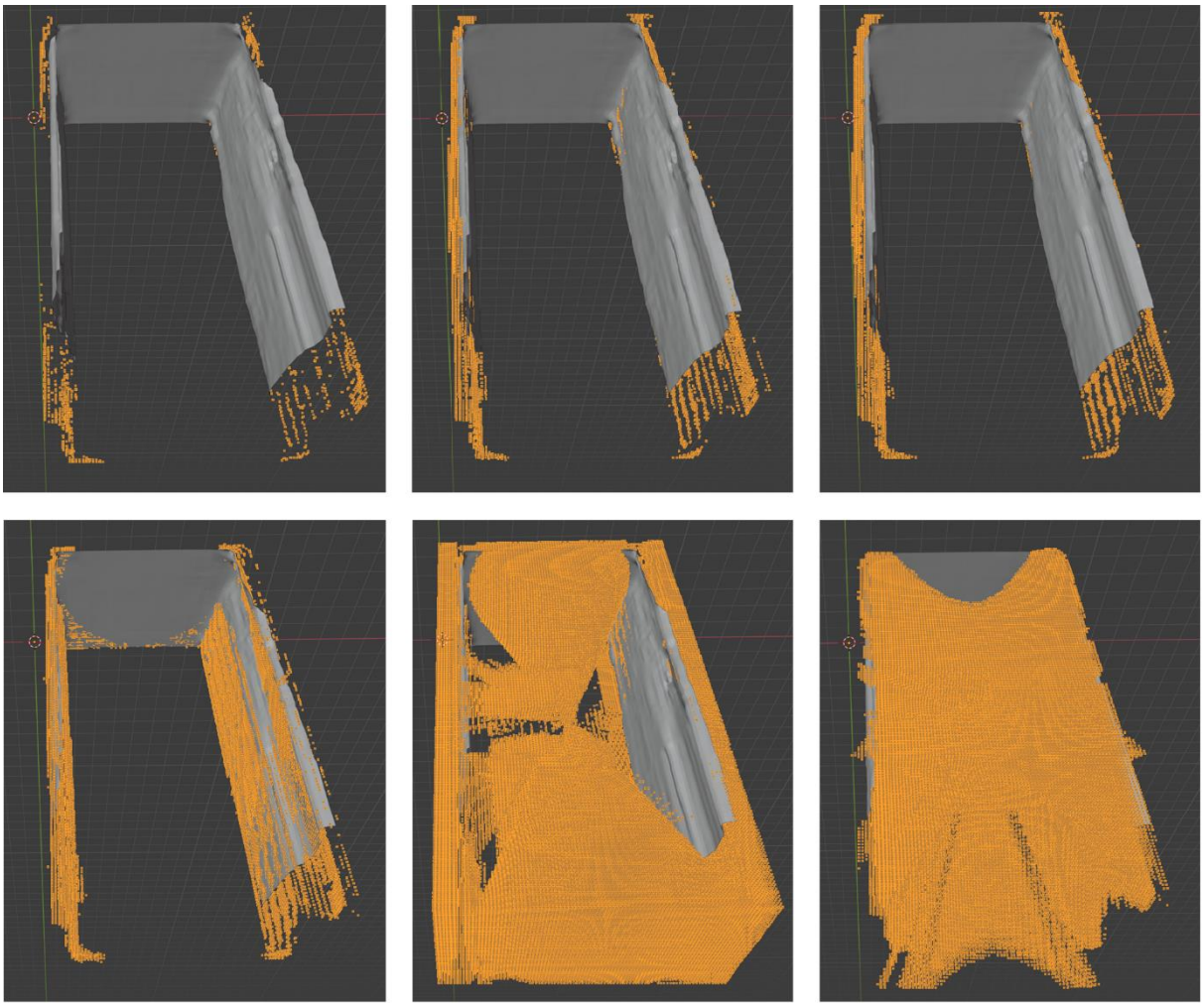

Figure 42: Known file with its different categories

Unfortunately, no clear scheme is visible from the images as to how the known file was created and what purpose it could fulfill. After an unsuccessful deep analysis of the source code and an attempt to adapt SG-NN to function without the known file, the importance of the categories was tested by assigning only one category to the whole known file. The result of the prediction is depicted in Figure 43. The image on the left shows the prediction with its original known file and the image on the right represents the completion with the modified known file with only one category. Since almost no difference is visible in their completeness, further investigation into the purpose of the

known file was not carried out. Thus, the created known files in this master thesis consist only of all voxels from the space of a scene with one assigned category.

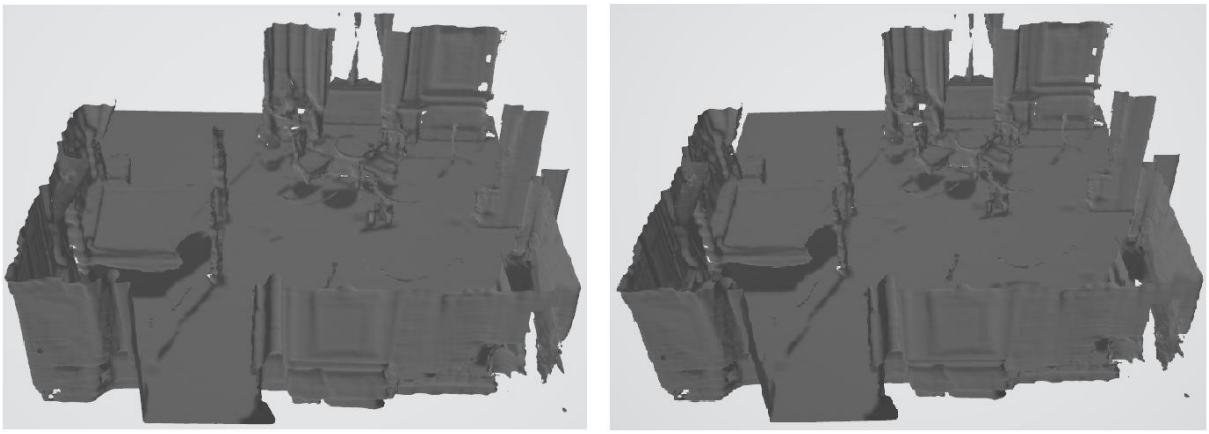

Figure 43: Known file

Since SG-NN uses an input and a target file to produce the predicted file, it was determined what happens if the already trained model only uses the target file for prediction. In Figure 44 on the right side, the prediction is shown with only the target file and on the left side with the input and target files. As can be seen in the figure, the results are nearly equally accurate. Therefore, in the following sections, only the real scan is used for the input and target file.

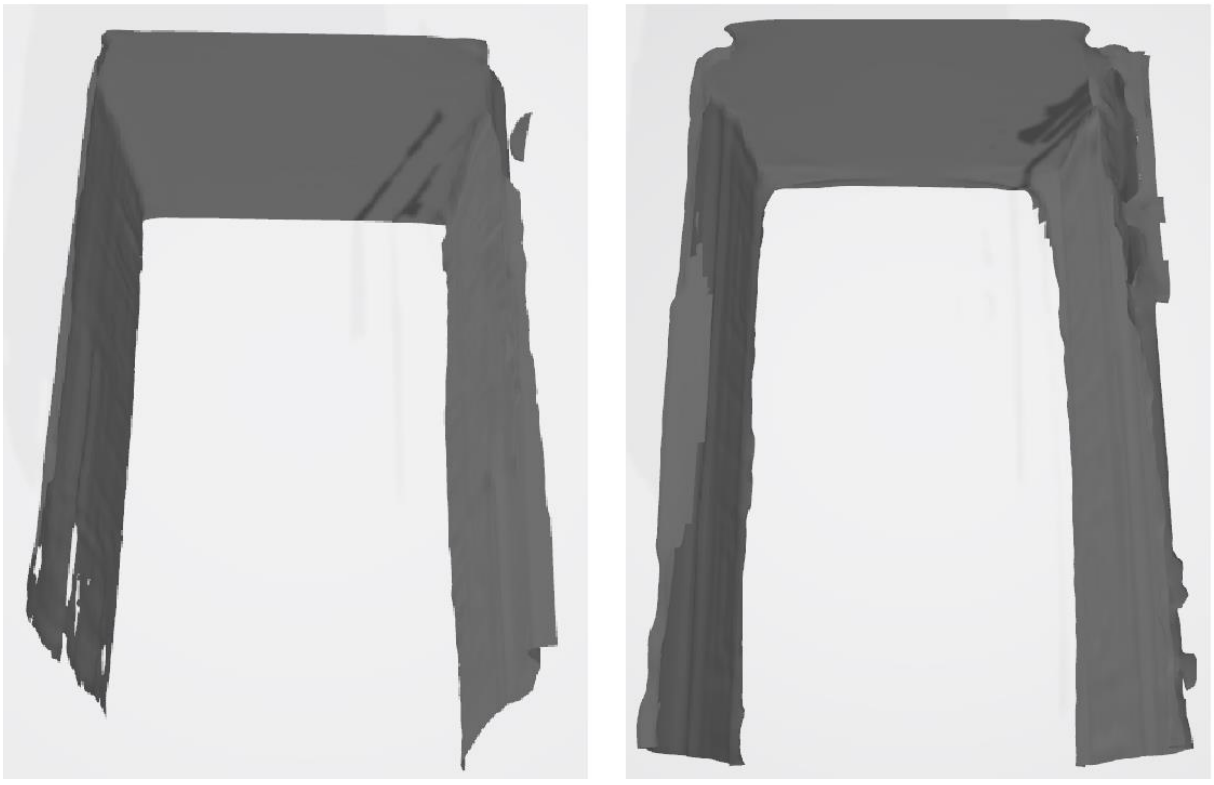

Figure 44: Prediction with two target TSDFs

Since the algorithm proposed in Section 6.2 always calculates the distance between two voxel centers, this results in an accumulation of equal distances. However, the model of SG-NN was trained with the distances to the real surfaces. To test whether this could lead to problems with the evaluation of the completion of point clouds in the next chapter, all distances of the input TSDF are rounded. Figure 45 displays on the left side the prediction with rounded numbers and on the right side the original prediction. As can be seen from the figure, the prediction is significantly worse. Thus, for an accurate completion the model must be retrained with the point cloud data. However, the accuracy is sufficient for a first evaluation of the approach.

Figure 45: TSDF with rounded distances

7.2 Point cloud to TSDF

Figure 46 shows on the left the input TSDF created from the point cloud on the right using the proposed algorithm of this thesis. The point cloud, containing the authors living room, was recorded with the LiDAR sensor on an iPhone. Unfortunately, only a few spots are visible in the input TSDF due to the fact that the TSDF is exceptionally sparse, and the marching-cubes algorithm that converts the TSDF into a mesh cannot handle this well.

Figure 46: Point cloud on the right with the corresponding TSDF on the left.

When using this very sparse TSDF from Figure 46 as input for SG-NN, the prediction result is not optimal, as can be seen in Figure 47. The scene has been significantly more completed compared to the input TSDF, but it does not contain as much information as the input point cloud. Moreover, it has defects at the four corners, in each of which an area on the ground is indicated even though there is no actual surface there. These areas on the ground extend to the respective end of the defined voxel space. Since this complete voxel space is only defined in the headers of the TSDF file and the known file, there is a strong assumption that this is related to the known file.

To optimize the completions, the input TSDF file must be less sparse. To verify this hypothesis, the number of voxels was increased before truncation in Figure 48.

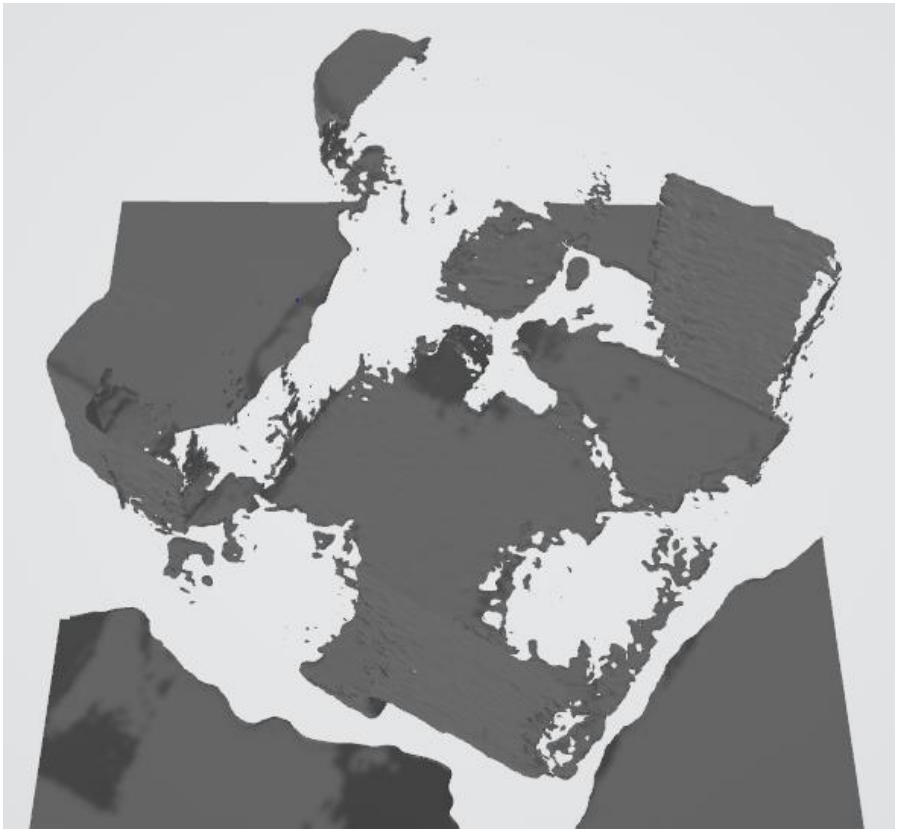

Figure 47: Prediction TSDF with truncation after 3 voxels

Figure 48 shows the prediction TSDF with a truncation after seven voxels. Although the algorithm is not optimized for this, a more complete picture is already visible.

Figure 48: Prediction TSDF with truncation after 7 voxels

Since every step of the Bresenham algorithm moves one voxel on the driving axis forward, some voxels that lie on the projection line are left out. To address this, the algorithm entitled "A Fast Voxel Traversal Algorithm for Ray Tracing" by Amanatides et al. [28] could be used. Another or additional possibility would be to adjust the algorithm so that the voxels directly next to the line from the occupied voxel to *K* are also saved and assigned the distance to the occupied voxel.

When following up on these approaches, it was found that the possibility of storing the same voxel multiple times is high. To avoid this, the algorithm must sort out multiple voxels and store only the voxels with the shortest distances. This problem is visualized in Figure 49.

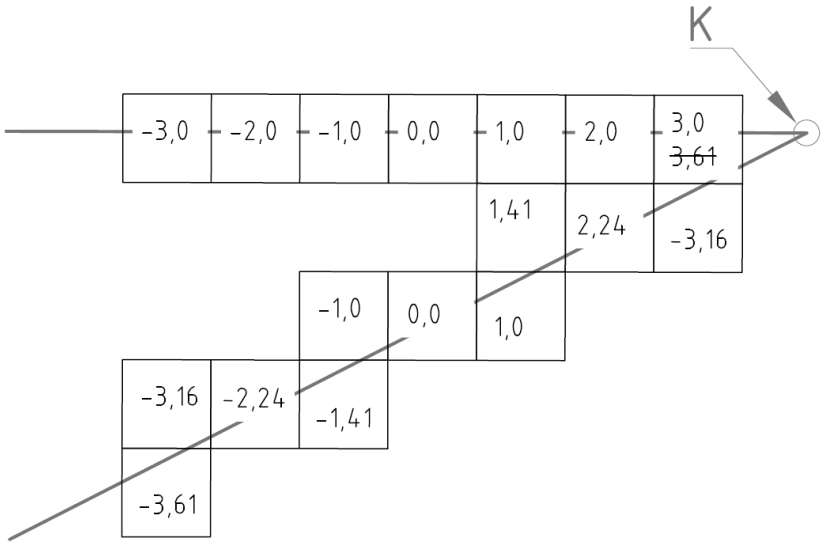

Figure 49: Multiple distances for one voxel

Based on the above discussion, the conclusion is that it is possible to convert a point cloud to a TSDF and that this can be a successful approach for completion if further adjustments are made.

7.3 Point cloud to TDF

In order to perform a first investigation into whether a TDF is at all suitable for a completion, the same algorithm as described in Subsection 6.2.3 was used, with the adjustment that the negative values were stored as positive values. The result of the completion is illustrated in Figure 50. It is obvious the result is worse compared to Figure 48. However, the level of completion is already informative and with the proposed approach of Chapter 6.3 the TDF would be much denser. Therefore, it can be assumed that the approach with a TDF instead of a TSDF is to be preferred. On the one hand, the low sparsity is eliminated; on the other hand, the point clouds for a TDF can be recorded from multiple viewing angles, thus making available a more accurate ground truth with which to train.

Figure 50: TDF with a truncation after 7 voxels

7.4 Point cloud generation through a LiDAR sensor

As discussed in Section 6.4, there is potential for a dataset generated with a LiDAR sensor on an iPhone or iPad. Especially, the widespread availability of iPhones and iPads has made possible a large-scale real-world dataset that could rival synthetic datasets, for example SUNCG with 45,000 scenes. Another issue that could be resolved using a crowdsourced dataset is the scarcity of rooms in private apartments in existing datasets. By having a dataset containing point clouds rather than RGB-D scans, the privacy issues would be far less severe.

Currently, the LiDAR sensors in iPhones and iPads only provide good results from one central scanning point. To obtain a more detailed point cloud dataset with fewer obscured areas, further research must be done into the registration process of the scans. In addition, the accuracy of the LiDAR sensor should be further tested since the results in this thesis only went in one direction. However, even with these flaws, a large-scale dataset from one standpoint would be an asset in training 3D deep learning models.

7.5 Possible other approaches

Other approaches were also considered in the context of this work but were not further pursued. In the following, these two approaches are briefly presented, and it is explained why they were not further pursued.

The first idea involved converting a point cloud into a grey scale depth image and then using the same volumetric fusion process as in SG-NN to create a TSDF. The depth images are taken from the center of the room similar to in Subsection 6.2 and they always contain the distance from the surface points to the viewpoint. Figure 51 visualizes this approach. For point clouds with color information, even an RGB-D image would be possible, although this is not required for volumetric fusion. Chmelar et al. [8] describes an approach for converting point clouds into depth images.

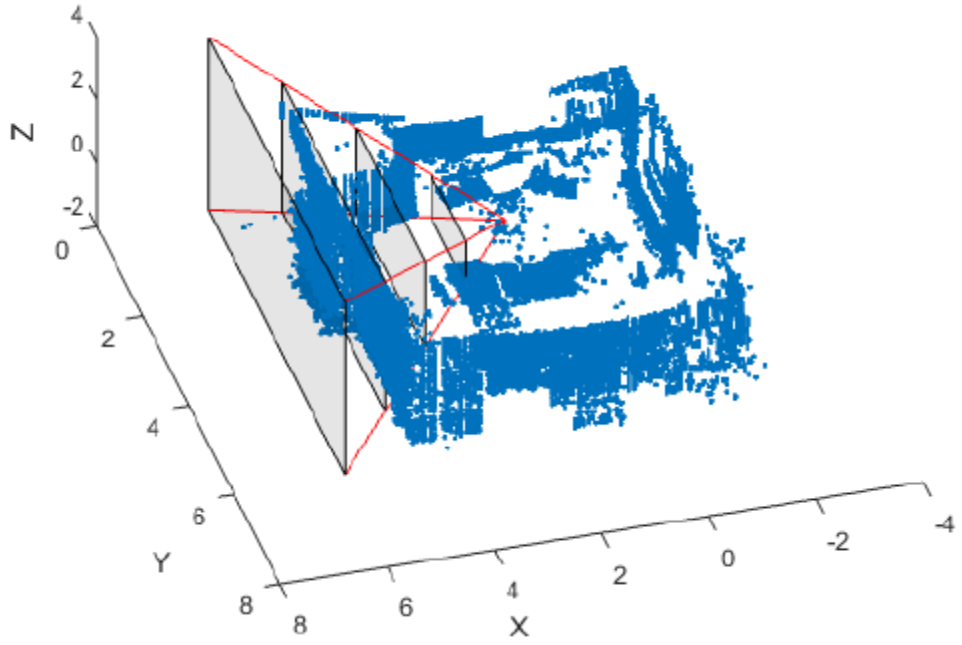

Figure 51: Point cloud to depth image [8]

Ultimately this approach was not chosen because the conversion from a point cloud to an RGB-D image to a TSDF requires additional computational need compared to the direct transformation from a point cloud to a TSDF and more spatial information is lost due to the additional transformations.

The second approach considered, was to rebuild the entire network and generate an encoder based on the PointNet [44] architecture. For example, through max pooling layers and convolution the input points could be compressed into a latent space vector, which then be transformed into a distance field through 3D convolutions. This approach would lie beyond the scope of this thesis.

8 Concluding Remarks

The objective for this master thesis was to develop an algorithm for the scene completion of point clouds. In Subsection 8.1 the methods developed and results are summarized. Subsection 8.2 describes possibilities for further development.

8.1 Conclusion

In this thesis, the scene completion network SG-NN by Dai et al. [12] was first updated to run on Pytorch 1.8.1 and Python 3.8. The results with the TSDFs from generated RGB-D images are as expected very accurate.

Afterwards a novel approach was proposed and implemented that generates a TSDF from a point cloud. Unfortunately, the approach was not entirely successful. It turned out that the original idea of creating the TSDF as sparse as possible for computational and memory reasons had a detrimental effect on the description of the scene. To address this, improvements were proposed in Subsection 7.2. With the implementation of these improvements and the training of SG-NN with a large-scale dataset of point clouds, it should be possible to achieve the same results as with an RGB-D dataset.

In addition, the thesis briefly investigated whether a TDF can also provide the necessary descriptive information to be considered as an input for a SC algorithm such as SG-NN. This analysis was performed using the same algorithm that transforms a point cloud into TSDF. The negative values of the TSDF were assigned positive values. This implies that the TDF is not as dense represented as defined. However, even with this sparse TDF, usable results could already be obtained. It can be assumed that when a dense TDF is used for training, the results provide sufficient descriptive information for a proper completion. The major advantage of a TDF is that it can also be generated from a point cloud, which has been taken from several scan points and fused through point cloud registration. Thus, this approach is more promising than the TSDF approach.

In the final step, the suitability of the LiDAR sensor of iPhones and iPads for generating a dataset was examined. At present, the sensors are useful for capturing a point cloud of a scene or room from one position. If a point cloud is to be created from many standpoints – for example, to obtain a better ground truth – the iPhone or iPad is not suitable

at this time. The automatic point cloud registration does not yet function seamlessly on the devices. When moving through the room to scan details such as hidden corners or the floor under a table, errors occur that are characterized by double-scanned objects. One way to fix this would be to manually register the point clouds. However, it is also expected that automatic registration will improve in the next years. The potential to generate large-scale crowd-source datasets is definitely existent.

8.2 Recommendations for Further Work

Recommendations for further development are provided below, ordered in decreasing order of importance.

- Since the study in this thesis is based on the accuracy of the LiDAR sensor of one device, a deeper analysis of the accuracy based on different devices would be beneficial. Based on this information, a crowd-source dataset of point clouds could be obtained. This would be extremely valuable for training deep learning models.

- As described in Subsection 7.3, the conversion of point clouds to TDFs is very promising when using volumetric scene completion methods. Further development and implementation of the approach in chapter 6.3 is recommended. To minimize the computational cost, the use of the rectilinear distance instead of the Euclidean distance could also be considered.

- A purely point-based approach, as described in Subsection 7.5, would be also suitable for further development. This would make it possible to draw a comparison between a point-based method and a volumetric method where the point cloud was previously converted into a TSDF or TDF.

References

[1] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. 2019. *SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences*.

[2] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. *Matterport3D: Learning from RGB-D Data in Indoor Environments*.

[3] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. *ShapeNet: An Information-Rich 3D Model Repository*.

[4] Rong Chen, Zhiyong Huang, and Yuanlong Yu. 122019. AM 2 FNet: Attention-based Multiscale & Multi-modality Fused Network. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 1192–1197. DOI: https://doi.org/10.1109/ROBIO49542.2019.8961556.

[5] Xiaokang Chen, Kwan-Yee Lin, Chen Qian, Gang Zeng, and Hongsheng Li. 2020. *3D Sketch-aware Semantic Scene Completion via Semi-supervised Structure Prior*.

[6] Yueh-Tung Chen, Martin Garbade, and Juergen Gall. 2019. *3D Semantic Scene Completion from a Single Depth Image using Adversarial Training*.

[7] Ian F. Cherabier, Johannes L. Schönberger, Martin R. Oswald, Marc Pollefeys, and Andreas Geiger. 2018. Learning Priors for Semantic 3D Reconstruction. In *2018 European Conference on Computer Vision*, 314–330.

[8] Pavel Chmelar, Ladislav Beran, and Lubos Rejfek. 2016. The Depth Map Construction from a 3D Point Cloud. *MATEC Web Conf.* 75, 3005. DOI: https://doi.org/10.1051/matecconf/20167503005.

[9] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 2016. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. In *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu

Sebe and Max Welling, Eds. Lecture Notes in Computer Science. Springer International Publishing, Cham, 628–644. DOI: https://doi.org/10.1007/978-3-319-46484-8_38.

[10] Brian Curless and Marc Levoy. 1996. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*. ACM Press, New York, New York, USA, 303–312. DOI: https://doi.org/10.1145/237170.237269.

[11] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. 2017. *ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes*.

[12] Angela Dai, Christian Diller, and Matthias Niessner. 62020. SG-NN: Sparse Generative Neural Networks for Self-Supervised Scene Completion of RGB-D Scans. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 846–855. DOI: https://doi.org/10.1109/CVPR42600.2020.00093.

[13] Angela Dai, Charles R. Qi, and Matthias Nießner. 2016. *Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis*.

[14] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. 2017. *ScanComplete: Large-Scale Scene Completion and Semantic Segmentation for 3D Scans*.

[15] Aloisio Dourado, Teofilo E. d. Campos, Hansung Kim, and Adrian Hilton. 2019. *EdgeNet: Semantic Scene Completion from a Single RGB-D Image*.

[16] Aloisio Dourado, Hansung Kim, Teofilo E. de Campos, and Adrian Hilton. 2272020. Semantic Scene Completion from a Single 360-Degree Image and Depth Map. In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 36–46. DOI: https://doi.org/10.5220/0008877700360046.

[17] Michael Firman, Oisin M. Aodha, Simon Julier, and Gabriel J. Brostow. 62016. Structured Prediction of Unobserved Voxels from a Single Depth Image. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 5431–5440. DOI: https://doi.org/10.1109/CVPR.2016.586.

[18] Martin Garbade, Yueh-Tung Chen, Johann Sawatzky, and Juergen Gall. 2018. *Two Stream 3D Semantic Scene Completion*.

[19] Franca Giannini and Alexander Pasko. 2004. *SMI 2004*. *Shape Modeling International 2004 7-9 June 2004, Genova, Italy*. IEEE Computer Society, Los Alamitos Calif.

[20] Andre B. S. Guedes, Teofilo E. d. Campos, and Adrian Hilton. 2018. *Semantic Scene Completion Combining Colour and Depth: preliminary experiments*.

[21] Ruiqi Guo and Derek Hoiem. 122013. Support Surface Prediction in Indoor Scenes. In *2013 IEEE International Conference on Computer Vision*. IEEE, 2144–2151. DOI: https://doi.org/10.1109/ICCV.2013.266.

[22] Yu-Xiao Guo and Xin Tong. 2018. *View-volume Network for Semantic Scene Completion from a Single Depth Image*.

[23] Xiaoguang Han, Zhen Li, Haibin Huang, Evangelos Kalogerakis, and Yizhou Yu. 2017. *High-Resolution Shape Completion Using Deep Neural Networks for Global Structure and Local Geometry Inference*.

[24] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. 2015. *SceneNet: Understanding Real World Indoor Scenes With Synthetic Data*.

[25] Yong He, Hongshan Yu, Xiaoyan Liu, Zhengeng Yang, Wei Sun, Yaonan Wang, Qiang Fu, Yanmei Zou, and Ajmal Mian. 2021. *Deep Learning based 3D Segmentation: A Survey*.

[26] Iro Armeni, Sasha Sax, Amir R. Zamir, Silvio Savarese. 2017. Joint 2D-3D-Semantic Data for Indoor Scene Understanding.

[27] Chiyu ". Jiang, Dequan Wang, Jingwei Huang, Philip Marcus, and Matthias Nießner. 2019. *Convolutional Neural Networks on non-uniform geometrical signals using Euclidean spectral transformation*.

[28] John Amanatides, Andrew Woo. 1987. A Fast Voxel Traversal Algorithm A Fast Voxel Traversal Algorithm for Ray Tracing.

[29] John McCormac, Ankur Handa, Stefan Leutenegger, Andrew J. Davison. SceneNet RGB-D 5M Photorealistic Images of Synthetic Indoor Trajectories with Ground Truth.

[30] Michael Kazhdan and Hugues Hoppe. 2013. Screened poisson surface reconstruction. *ACM Trans. Graph.* 32, 3, 1–13. DOI: https://doi.org/10.1145/2487228.2487237.

[31] Jie Li, Kai Han, Peng Wang, Yu Liu, and Xia Yuan. 2020. *Anisotropic Convolutional Networks for 3D Semantic Scene Completion*.

[32] Jie Li, Yu Liu, Dong Gong, Qinfeng Shi, Xia Yuan, Chunxia Zhao, and Ian Reid. 2019. *RGBD Based Dimensional Decomposition Residual Network for 3D Semantic Scene Completion*.

[33] Siqi Li, Changqing Zou, Yipeng Li, Xibin Zhao, and Yue Gao. 2020. *Attention-based Multi-modal Fusion Network for Semantic Scene Completion*.

[34] Chen Liu, Jiaye Wu, and Yasutaka Furukawa. 2018. *FloorNet: A Unified Framework for Floorplan Reconstruction from 3D Scans*.

[35] Yu Liu, Jie Li, Qingsen Yan, Xia Yuan, Chunxia Zhao, Ian Reid, and Cesar Cadena. 2020. *3D Gated Recurrent Fusion for Semantic Scene Completion*.

[36] Yu Liu, Jie Li, Xia Yuan, Chunxia Zhao, Roland Siegwart, Ian Reid, and Cesar Cadena. 2020. *Depth Based Semantic Scene Completion with Position Importance Aware Loss*.

[37] Maximilian Vogt and Adrian Rips and Claus Emmelmann. Comparison of iPad Pro®'s LiDAR and TrueDepth Capabilities with an Industrial 3D Scanning Solution.

[38] Michael Kazhdan, Matthew Bolitho, Hugues Hoppe. Poisson surface reconstruction.

[39] Gang Z. Min Zhong. Semantic Point Completion Network for 3D Semantic Scene Completion 2020.

[40] Niloy J. Mitra, Leonidas J. Guibas, and Mark Pauly. 2006. Partial and approximate symmetry detection for 3D geometry. In *ACM SIGGRAPH 2006 Papers on - SIGGRAPH '06*. ACM Press, New York, New York, USA, 560. DOI: https://doi.org/10.1145/1179352.1141924.

[41] Duc T. Nguyen, Binh-Son Hua, Minh-Khoi Tran, Quang-Hieu Pham, and Sai-Kit Yeung. 62016. A Field Model for Repairing 3D Shapes. In *2016 IEEE Conference*

on Computer Vision and Pattern Recognition (CVPR). IEEE, 5676–5684. DOI: https://doi.org/10.1109/CVPR.2016.612.

[42] Diana Pagliari and Livio Pinto. 2015. Calibration of Kinect for Xbox One and Comparison between the Two Generations of Microsoft Sensors. *Sensors (Basel, Switzerland)* 15, 11, 27569–27589. DOI: https://doi.org/10.3390/s151127569.

[43] Ameya Phalak, Zhao Chen, Darvin Yi, Khushi Gupta, Vijay Badrinarayanan, and Andrew Rabinovich. 2019. *DeepPerimeter: Indoor Boundary Estimation from Posed Monocular Sequences*.

[44] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2016. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*.

[45] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*.

[46] Ran Cheng, Christopher Agia, Yuan Ren, Xinhai Li, Liu Bingbing. S3CNet A Sparse Semantic Scene Completion Network for LiDAR Point Clouds.

[47] Luis Roldão, Raoul d. Charette, and Anne Verroust-Blondet. 2020. *LMSCNet: Lightweight Multiscale 3D Semantic Completion*.

[48] Muhammad Sarmad, Hyunjoo J. Lee, and Young M. Kim. 2019. *RL-GAN-Net: A Reinforcement Learning Agent Controlled GAN Network for Real-Time Point Cloud Shape Completion*.

[49] Shice Liu, YU HU, Yiming Zeng, Qiankun Tang, Beibei Jin, Yinhe Han, and Xiaowei Li. See and Think: Disentangling Semantic Scene Completion.

[50] Nathan Silberman and Rob Fergus. 112011. Indoor scene segmentation using a structured light sensor. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 601–608. DOI: https://doi.org/10.1109/ICCVW.2011.6130298.

[51] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. 2012. Indoor Segmentation and Support Inference from RGBD Images. In *Computer Vision – ECCV 2012*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Andrew Fitzgibbon, Svetlana Lazebnik, Pietro

Perona, Yoichi Sato and Cordelia Schmid, Eds. Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg, 746–760. DOI: https://doi.org/10.1007/978-3-642-33715-4_54.

[52] Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser. 2016. *Semantic Scene Completion from a Single Depth Image*.

[53] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. 2017. *Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs*.

[54] Lyne P. Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. 62019. TopNet: Structural Point Cloud Decoder. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 383–392. DOI: https://doi.org/10.1109/CVPR.2019.00047.

[55] University of Zurich. 2014. *UZH 3D dataset* (2014). Retrieved March 8, 2021 from https://www.ifi.uzh.ch/en/vmml/research/datasets.html.

[56] Jacob Varley, Chad DeChant, Adam Richardson, Joaquin Ruales, and Peter Allen. 92017. Shape completion enabled robotic grasping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2442–2447. DOI: https://doi.org/10.1109/IROS.2017.8206060.

[57] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-CNN. *ACM Trans. Graph.* 36, 4, 1–11. DOI: https://doi.org/10.1145/3072959.3073608.

[58] Peng-Shuai Wang, Yang Liu, and Xin Tong. 2020. *Deep Octree-based CNNs with Output-Guided Skip Connections for 3D Shape and Scene Completion*.

[59] Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. 2019. Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes. *ACM Trans. Graph.* 37, 6, 1–11. DOI: https://doi.org/10.1145/3272127.3275050.

[60] Xiaojuan Wang, Martin R. Oswald, Ian Cherabier, and Marc Pollefeys. 2019. Learning 3D Semantic Reconstruction on Octrees. In *Pattern Recognition*, Gernot A. Fink, Simone Frintrop and Xiaoyi Jiang, Eds. Lecture Notes in Computer Science. Springer International Publishing, Cham, 581–594. DOI: https://doi.org/10.1007/978-3-030-33676-9_41.

[61] Yida Wang, David J. Tan, Nassir Navab, and Federico Tombari. 2018. Adversarial Semantic Scene Completion from a Single Depth Image, 426–434. DOI: https://doi.org/10.1109/3DV.2018.00056.

[62] Yida Wang, David J. Tan, Nassir Navab, and Federico Tombari. 2019. *ForkNet: Multi-branch Volumetric Semantic Completion from a Single Depth Image*.

[63] Shun-Cheng Wu, Keisuke Tateno, Nassir Navab, and Federico Tombari. 2020. SCFusion: Real-time Incremental Scene Reconstruction with Semantic Completion, 801–810. DOI: https://doi.org/10.1109/3DV50981.2020.00090.

[64] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 62015. 3D ShapeNets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1912–1920. DOI: https://doi.org/10.1109/CVPR.2015.7298801.

[65] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. 122013. SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels. In *2013 IEEE International Conference on Computer Vision*. IEEE, 1625–1632. DOI: https://doi.org/10.1109/ICCV.2013.458.

[66] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. 2018. *PCN: Point Completion Network*.

[67] Jiahui Zhang, Hao Zhao, Anbang Yao, Yurong Chen, Li Zhang, and Hongen Liao. 2018. Efficient Semantic Scene Completion Network with Spatial Group Convolution. In *Computer Vision – ECCV 2018*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu and Yair Weiss, Eds. Lecture Notes in Computer Science. Springer International Publishing, Cham, 749–765. DOI: https://doi.org/10.1007/978-3-030-01258-8_45.

[68] Pingping Zhang, Wei Liu, Yinjie Lei, Huchuan Lu, and Xiaoyun Yang. 2019. *Cascaded Context Pyramid for Full-Resolution 3D Semantic Scene Completion*.

[69] Wei Zhao, Shuming Gao, and Hongwei Lin. 102007. A Robust Hole-Filling Algorithm for Triangular Mesh. In *2007 10th IEEE International Conference on Computer-Aided Design and Computer Graphics*. IEEE, 22. DOI: https://doi.org/10.1109/CADCG.2007.4407836.

[70] Matt Zucker, J. A. Bagnell, Christopher G. Atkeson, and James Kuffner. 052010. An optimization approach to rough terrain locomotion. In *2010 IEEE International Conference on Robotics and Automation*. IEEE, 3589–3595. DOI: https://doi.org/10.1109/ROBOT.2010.5509176.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Master-Thesis selbstständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ich versichere außerdem, dass die vorliegende Arbeit noch nicht einem anderen Prüfungsverfahren zugrunde gelegen hat.

München, 16. Juli 2021

Vorname Nachname

[redacted]