Deep Learning-based 3D Surface Reconstruction – A Survey

Anis Farshian, Markus Götz, *Member, IEEE*, Gabriele Cavallaro, *Senior Member, IEEE*,
Charlotte Debus, *Member, IEEE*, Matthias Nießner, Jón Atli Benediktsson, *Fellow, IEEE*, and Achim Streit

***Abstract*—In the last decade, deep learning has significantly impacted industry and science. Initially largely motivated by computer vision tasks in two-dimensional imagery, the focus has shifted towards three-dimensional data analysis. In particular, 3D surface reconstruction, i.e., reconstructing a three-dimensional shape from sparse input, is of great interest to a large variety of application fields. Deep learning-based approaches show promising quantitative and qualitative surface reconstruction performance compared to traditional computer vision and geometric algorithms. This survey provides a comprehensive overview of these deep learning-based methods for 3D surface reconstruction. To this end, we will first discuss input data modalities, such as volumetric data, point clouds as well as RGB, single-view, multi-view, and depth images, along with corresponding acquisition technologies and common benchmark datasets. For practical purposes, we also discuss evaluation metrics enabling to judge the reconstructive performance of different methods. The main part of the document will introduce a methodological taxonomy ranging from point- and mesh-based techniques, to volumetric and implicit neural approaches. Recent research trends, both methodological and for applications, are highlighted, pointing towards future developments.**

***Index Terms*—machine learning, 3D deep learning, 3D surface reconstruction, geometric deep learning, geometry processing**

I. Introduction

In the last decade, advances in artificial intelligence, in particular in deep learning (DL) [1–3], has been adopted by a multitude of fields and have thus led to major breakthroughs in science and industry alike. One of the major driving forces behind these developments is the field of computer vision, and its desire to "teach" machines how to recognize patterns within image and video data. Initially, a strong emphasis was placed on the interpretation of 2D information; however, recent advances in cost-effective scanner-based data acquisition and the establishment of large-scale shape repositories have brought the analysis of 3D data into focus. Still, complexity, variety and irregularities in three-dimensional shape representations pose significant methodological challenges.

The reconstruction of 3D surfaces of objects from different types of input data formats, such as point clouds, depth maps, single-view or multi-view images, is fundamental to a number of application fields such as computer vision, robotics, CAD, medicine, city planning, disaster prevention, and archaeology. One of the special use cases of 3D reconstruction is human shape reconstructions and pose estimation from images or videos, which is addressed by some other works [4, 5]

Despite a long research history for 3D surface reconstruction, the precise representation of three-dimensional geometrical objects remains an unsolved problem, usually requiring the reconstructed 3D surfaces to be: 1) highly resolved and smooth, 2) water-tight, i.e., "without gaps", 3) in accordance with possible ground-truth, 4) robust against noisy or incomplete input, and 5) simultaneously densely and compressibly represented.

Classical approaches for addressing these problems encompass geometrical or simplistic machine-learning-based algorithms [6, 7]. Most of these methods are not able to comprehensively and consistently reconstruct arbitrary detailed 3D surfaces. Well-known techniques, such as (screened) poisson surface reconstruction [8, 9], the ball-pivoting algorithm (BPA) [10] as well as delaunay triangulation [11, 12] still suffer from scalability issues and struggle to reconstruct fine details for large-scale data.

The recent successes of deep neural networks (DNNs) in other data-driven computational problems such as classification [13, 14], object detection [14, 15], and segmentation [13, 14, 16], have sparked interest in utilizing deep learning for 3D surface reconstruction. Partially overlapping with the latter is the task of shape completion, i.e., enhancing the input data with (partially) occluded shape information.

Several reconstruction-related surveys [17, 18] present early approaches, with [17] providing an overview of the classical and non-deep learning-based surface reconstruction methods from point clouds with respect to priors, and [18] reviewing RGB-D scene reconstruction approaches. There is another deep learning-based surface reconstruction survey [19] with the focus on image-based methods. Our paper, however, covers broader data modalities and reviews recent trends in 3D surface reconstruction including implicit neural representation and neural radiance fields thoroughly.

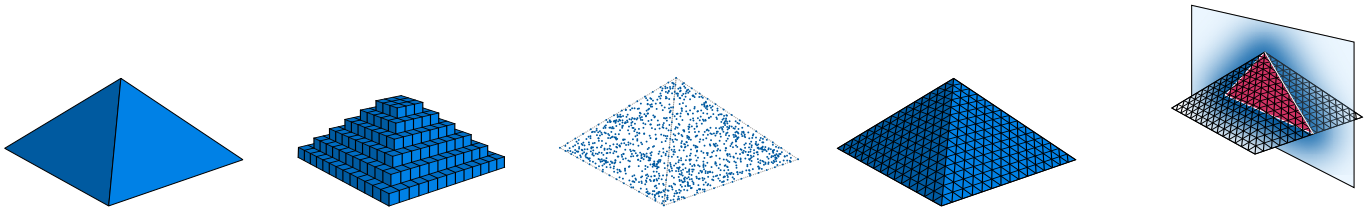
Therefore, the fast-paced development of the field makes it, however, necessary to revisit up-to-date research frequently. The current landscape of deep learning-based 3D reconstruction can be broadly classified into four main categories according to their representation, as depicted in Fig. 1: 1) volumetric, i.e., representing a surface with small cuboids, either a dense 3D voxel grid [20–25], or an octree [26–29],

A. Farshian, M. Götz, C. Debus and A. Streit are with Helmholtz AI and the Steinbuch Centre for Computing, Karlsruhe Institute of Technology, 76344 Eggenstein-Leopoldshafen, Germany, e-mail: {anis.farshian, markus.goetz, charlotte.debus, achim.streit}@kit.edu

G. Cavallaro is with the Jülich Supercomputing Centre, Forschungszentrum Jülich, Wilhelm-Johnen-Straße 52428 Jülich, Germany, and with the University of Iceland, 107 Reykjavik, Iceland (e-mail: g.cavallaro@fz-juelich.de)

Matthias Nießner is with the Technical University Munich, 80333 Munich, Germany, e-mail: niessner@tum.de

Jón Atli Benediktsson is with the University of Iceland, 102 Reykjavík, Iceland, e-mail: benedikt@hi.is



(a) Object (b) Voxelized (c) Point cloud (d) Mesh (e) Implicit

Fig. 1: Output representations of various 3D surface reconstruction approaches. DL-based 3D surface reconstruction approaches can be broadly classified into four main categories according to their representation: volumetric, point cloud, mesh, and an example of implicit neural representation based on signed distance function (SDF)

.

2) point-based [30–33], i.e., using points to present a surface, 3) mesh-based [34–44], i.e., describing an object with vertices, edges, and faces , and 4) implicit neural representation [45–56], i.e., representing a shape as a neural network that takes any (x, y, z) coordinate as input and maps it to an occupancy or signed distance of the shape at that coordinate, or modeling radiance or appearance properties of an object such as NeRF-based approaches [57–68].

In this survey, we present a comprehensive overview of these state-of-the-art deep learning-based approaches to 3D surface reconstruction. Our main goal is to provide method researchers with a guide to current work and applied researchers with a toolbox for their domain challenges. Towards this end, we first provide a broad introduction to input data formats (Section II), acquisition technologies (Section III) as well as widely used benchmarking datasets (Section IV). Section V covers evaluation metrics enabling to quantitatively judge the reconstructive performance of a method, independent of being classical or learning-based. The main part of the survey (Section VI) highlights deep learning methods to reconstruct 3D surfaces using volumetric, point- and mesh-based as well as implicit neural representation. We assume that the reader has a general grasp of neural networks and deep learning concepts to thoroughly follow the content. Discussion, current trends, and challenges are highlighted in section VII. Finally, section VIII summarizes and concludes the survey.

II. Input Data

Various types of data representations can be used as input for 3D surface reconstruction task. Conventional representations of 3D inputs can be divided into Euclidean and non-Euclidean data. Examples of non-Euclidean data representations are point clouds or meshes, while Euclidean data representations can be volumetric, RGB-D data, or multi-view images.

Point clouds are currently the most common format of raw 3D sensor data. With the improvement of scanning devices, leading to enhanced capabilities for capturing the surrounding 3D environment in various applications and representing it with points, point clouds are becoming increasingly important and available. Thus, processing this type of representation using neural networks and deep learning techniques has attracted considerable attention. From a mathematical point of view, point clouds comprise an irregular data structure in the form of an unordered set of points. Each point on a 3D surface of an object can basically be defined by a vector of its (x, y, z) coordinates, which can be inferred by various 3D data acquisition techniques. Hence, the size of the representation matrix of a 3D object is initially $N \times 3$ for N points. The matrix may also contain different properties including color, transparency, surface normals, and other scanner information. However, pure point clouds do not include the interconnections between vertices. Since a point cloud is a set, its elements are orderless, a characteristic that causes many challenges for surface reconstruction methods. Point clouds can be easily converted to/extracted from other data representations, such as voxels, depth maps, or meshes, and vice versa. Furthermore, they can be extracted from depth images by projecting the depth value of each pixel into 3D space.

Meshes are another highly popular type of representation for 3D objects providing detailed and connected geometries in an efficient way. They are an irregular data embedding in continuous space. Their basic components are vertices, edges, i.e., pairs of vertices, and (triangluar) faces, i.e., n-tuples of edges, forming an undirected graph.

In volumetric representations, the basic element is a voxel. A voxel in a 3D grid is a cuboid equivalent to a pixel in 2D space. The 3D grid, regardless of being sparse or dense, can be fed to a neural network as the input.

An RGB-D image is the combination of an RGB image and a depth image. It not only has RGB information for each pixel but also includes depth information.

Multi-view images are a collection of (single-view) images taken from different angles from an object. By putting these images together, 3D information can be partly retrieved.

On the other hand, 2D data such as single-view RGB images can also be considered as the input to a network for surface reconstruction individually, in which the method is called single-view reconstruction [69–71], or in conjunction with another 3D input mentioned before.

III. Data Acquisition

As explained in the previous section, point clouds are the most common format of raw 3D sensor data. 3D point cloud data are acquired through sensing technologies that measure distance (i.e., 3D laser scanning also known as light detection

and ranging (LiDAR)) or generated with stereo- and multi-view image-derived systems that can be based on red, green, blue-depth (RGB-D) cameras, stereo cameras and multiple synthetic aperture radar (SAR) image pairs [72, 73]. High-quality 3D point clouds can capture the 3D surface geometries of target objects (e.g., physical features that occupy the earth's surface and ocean bottom) with a spatial accuracy up to the millimeter-level and a point density of a few thousand points per square meter (pts/m^2).

A. 3D laser scanning (LiDAR)

LiDAR is a remote sensing (RS) active technology that uses light in the form of a pulsed laser to measure the distance between the sensor and the object under study [74]. By measuring the time that emitted pulses take to travel to a target, LiDAR derives 3D representations of objects. LiDAR can also operate at different wavelengths (i.e., multispectral LiDAR [75, 76]) to discriminate the different spectral reflectance of land-cover classes [77, 78].

Depending on the platform on which the LiDAR sensor is mounted, a 3D laser scanner is classified as a terrestrial laser scanner (TLS or ground LiDAR), airborne laser scanner (ALS), mobile laser scanner (MLS) and unmanned laser scanner (ULS) [72, 73].

A TLS uses ground-based RS systems (e.g., tripods) to cover middle- or close-range areas with scans performed in all directions, including upwards [79]. Once scans of a single zone are completed, the tripod is moved to another location to scan from another angle or capture data from a new area. As TLS systems are static during the acquisition process, they reach the highest point cloud density and can prodce high-quality 3D models of interiors of building and heritage sites.

Nevertheless, TLS systems cannot always be used, especially for scanning restricted locations that are not safe or accessible for teams (e.g., areas of dense vegetation, unsafe building sites, etc.). In these cases, LiDAR sensors can be mounted on airborne platforms. ALS systems are also used to acquire point cloud data over large areas (e.g., for 3D building reconstruction [80]).

When target regions are directly accessible, their structures and objects can be reconstructed from data acquired by MLS systems, i.e., LiDAR sensors mounted on moving vehicles (e.g., to derive high-resolution 3D city models [81]).

Since drones and other unmanned vehicles have become cheaper and autonomous navigation more reliable [82], ALS and MLS are often operated as ULS systems. Their platforms are compact and lightweight, which enables them to be exploited as first responders for disaster management. ULS systems can make a first scan of the terrain to track movements and changes and deliver 3D mapping of the most affected locations [83, 84]).

B. Photogrammetry

While LiDAR performs a direct measurement of the target object, i.e., by physically hitting a feature with light and measuring the reflection, approaches based on photogrammetry or computer vision theory [85] use a set of overlapping images taken from different locations to identify isolated points within a target. This includes airborne photogrammetry but also satellite stereo systems, which can map larger regions quickly. Image-based reconstruction algorithms can estimate the relative locations of these points and eventually convert the overlapping images into a 3D point cloud. For instance, the structure from motion (SfM) algorithms [86] can process multi-view images simultaneously through estimating camera positions and orientations automatically, while dense matching and multi-view stereovision (MVS) algorithms [87] can generate a large volume of point clouds (e.g., large-scale scenarios, crowded environments, etc.).

C. RGB-D Camera

Similar to LiDAR, RGB-D cameras measure the distance between the sensor and the objects. Depth information of each RGB pixel of the image is retrieved via a depth sensor. An RGB-D camera generates a colored point cloud by mapping RGB images with depth information (i.e., images include the (x, y, z) spatial coordinates and RGB colors). In this case, the point cloud is not the direct result of RGB-D scanning [88, 89], since the camera generates pixel-wise depth data rather than unstructured points. RGB-D cameras are generally cheaper than LiDAR systems and are mostly used in indoor environments for close-range applications [90].

Structured light and Time of Flight (ToF) [91], which are active imaging systems, serve as depth cameras and calculate the distance from the sensor to an object, consequently providing 3D information. The depth of an object can be determined using ToF sensors by measuring the duration of light travel from the sensor to the object and back. By determining the time of flight of light, these sensors can calculate the object's distance and create a detailed depth map, which can be directly used or easily converted to a point cloud for instance. Structured light sensors employ the deformation of a projected pattern to determine the distance. By emitting a known light pattern onto a scene and examining how the pattern changes as it interacts with objects in the scene, these sensors are able to accurately measure the depth information of the objects. Structured light technology-based 3D scanners are comparatively more affordable, being lighter in weight than their laser-based counterparts as well. Due to their higher degree of sensitivity to lighting conditions, they may not operate well in outdoor environments or in challenging conditions such as dusty rooms. For black or glossy surfaces, a specific spray should be applied before 3D scanning.

D. SAR Point Cloud

Synthetic aperture radar (SAR) is an active RS system that can operate day and night and can penetrate clouds and smoke. Interferometric SAR (InSAR) extends the principle of SAR to the 3D domain [92] by taking advantage of the physical properties of microwaves [93]. An InSAR system compares the phase of multiple SAR image pairs acquired from slightly different viewing angles to generate InSAR-based point clouds. The SAR tomography (TomoSAR) and

persistent scatterer interferometry (PSI) are two major techniques that generate point clouds with InSAR [73]. They are used to monitor terrain changes (e.g., surface deformations, human-made structures [94]).

E. Videogrammetry

3D point clouds can also be reconstructed using video frames (i.e., the input data are video streams instead of a collection of images). This approach is referred to as videogrammetry [95] and is based on the principles of photogrammetry. It can reconstruct point clouds from the frames of a video since their information is sequentially interconnected. Videogrammetry approaches provide a valuable alternative to camera images. They can be semi-automatic since the search for target points in different images can be achieved by measuring or tracking features of interest between consecutive video frames. However, the reconstruction needs to be coupled with effective frame selection algorithms (e.g., video frames are selected based on the surveyed geometry) and robust 3D processing methodologies [96].

IV. Datasets

Deep learning approaches are data-demanding; thus they require large amounts of data with high-quality 3D shapes and ground truths. Recent developments in scanning and sensing technologies have led to the collection of various widely used and openly accesible benchmarking datasets. These datasets are used to train and evaluate the performance of deep learning methods for different tasks, including 3D reconstruction. In this section, we summarize some of the most popular datasets, which can be used by different 3D deep learning approaches, with a focus on 3D reconstruction. Table I offers a comparative overview of these datasets.

- ShapeNet [97] is a richly-annotated, large-scale synthetic dataset of 3D shapes represented by 3D computer-aided design (CAD) models of objects, providing roughly 3,000,000 shapes. This dataset has been used for computer graphics and vision purposes. The full ShapeNet dataset is not yet publicly available. It consists of several subsets, including ShapeNetCore and ShapeNetSem. ShapeNetCore contains single clean 3D shape that covers 55 common object categories with about 51,300 unique 3D shapes. ShapeNetSem is a smaller, more densely annotated subset, containing 12,000 shapes of a broader set of 270 categories. For each shape in ShapeNet, annotations such as its geometry, texture, parts, symmetry planes, voxelization, screenshot, category, alignment, and size are available. The final representation of an object in this dataset can be a 3D mesh. The 3D shapes are stored in the Wavefront object file format (`.obj`), which describes the surface geometry of a 3D shape and includes vertices and faces, along with material template library (`.mtl`) files used to store material definitions. An `.mtl` file is a companion file for one or more `.obj` files which describes some surface appearance properties.

- PartNet [98] is a dataset of 3D objects, built on top of ShapeNet with fine-grained, hierarchical, and instance-level 3D part annotations. The dataset comprises 573,585 part instances of 26,671 ShapeNet 3D shapes in 24 indoor object categories in an attempt to enable part-level understanding of 3D objects.

- ModelNet [99] is a large-scale CAD model synthetic dataset. It includes a comprehensive and clean collection of 127,915 CAD models with 662 object categories and consists of two subsets, ModelNet10 and ModelNet40 with 10 and 40 classes respectively. ModelNet10 has also been annotated with the orientation of the CAD models, which are given in the Geomview object file format (`.off`). The final representation of this dataset can be a mesh.

- KITTI [100, 101] is a real-world urban scene dataset composed of images and point clouds. The dataset was acquired by the autonomous driving platform Annieway while driving around the city of Karlsruhe. Evaluation benchmarks were developed for several computer vision and robotic tasks such as stereo, optical flow, visual odometry, SLAM, 3D object detection and 3D object tracking. Semantic KITTI [102], which is based on KITTI, provides point-wise annotations for semantic segmentation and semantic scene completion purposes. The dataset comprises 28 classes including classes for non-moving and moving objects.

- ScanNet [103] is a 3D reconstruction dataset of indoor scenes consisting of 2.5 million frames (views) derived from more than 1500 RGB-D scans. 3D camera poses, surface reconstructions, and instance-level semantic segmentations are also provided. All scans are reconstructed into 3D mesh models. The data is stored in polygon file format (`.ply`).

- Matterport3D [104] is another dataset facilitating RGB-D scene understanding. It captures 10,800 panoramic views from 194,400 RGB-D images of 90 building-scale scenes. The dataset is annotated with surface reconstructions as textured meshes, camera poses, and 2D/3D semantic segmentations.

- NYU depth v2 [105] introduced an annotated dataset of 1,449 RGB and depth images, consisting of 464 diverse indoor scenes. These images were acquired by RGB and depth cameras from Microsoft Kinect.

- Sun3D [106] is a real-world large-scale dataset of RGB-D frames with semantic object segmentations and camera pose used for scene understanding. It consists of 415 sequences captured for 254 different indoor spaces in 41 different buildings.

- SUN RGB-D [107] is a dataset containing over 10,000 RGB-D images from NYU depth v2 [105], Berkeley

B3DO [108], and SUN3D [106] datasets. These images are annotated with 2D segmentations (146,617 2D polygons), 3D object box (64,595 3D bounding boxes), 3D room layout, 3D object orientation, and scene category for each image.

- Sydney Urban Objects dataset [109] is a point cloud dataset that contains 631 scans of 26 different object classes including vehicles, pedestrians, trees, and signs taken in the city of Sydney. Each object's information is available in three file formats, ASCII CSV format (`.csv`), binary-packed CSV (`.bin`), and meta information files (`.meta`).

- ABC dataset [110] is a CAD model dataset with one million 3D models. Koch *et al.* [110] offered a pipeline that is able to convert these CAD models into other representations in order to be processable by deep learning techniques. These models are provided in `.obj` and 3D Systems' stereolithography CAD file format (`.stl`).

- Semantic3D.net [111] is a large labelled 3D point cloud dataset of natural scenes with over four billion points in eight class labels. These dense point clouds, which were recorded by terrestrial laser scanners, depict urban and rural outdoor terrestrial scenes.

- H3D [112] is a high-resolution real-world dataset containing both point clouds (H3D(PC)) and meshes (H3D(Mesh)) of airborne LiDAR data, and can be used for semantic segmentation in geospatial applications. The point clouds are classified in eleven classes and labeled 3D textured meshes can be derived from them.

- 3D Furnished Rooms with layOuts and semaNTics (3D-Front) [113], is a synthetic dataset of indoor CAD model scenes, containing 18,968 rooms with 3D objects. The individual objects are taken from 3D-FUTURE [114]. The CAD models are stored in `.obj` and `.mtl` file formats.

- 3D Furniture shape with TextURE (3D-FUTURE) [114] is a repository of 3D furniture shapes in the household scenario enriched with 3D and 2D annotations. It includes 20,240 synthetic images of 5,000 different rooms. Stylistic and texture details of individual objects are provided. The 3D models are stored in `.obj` file format.

- SensatUrban [115] is a dataset for urban-scale point cloud understanding. It covers 7.6 km^2 of urban areas in Birmingham, Cambridge, and York cities. The point clouds are obtained from high-resolution aerial images which are captured by the UAV mapping system.

- Stanford 3D Scanning Repository [116] is a surface reconstruction repository containing some famous 3D models such as the Stanford bunny, happy Buddha, dragon, and armadillo in `.ply` format. These 3D models and some others also exist in the Large Geometric Models Archive [117].

V. Evaluation Metrics

Evaluation metrics are used to assess the performance of deep learning models [1–3]. Various metrics have been proposed for testing deep geometric learning methods. Some of the common distance metrics used for surface reconstruction methods are Chamfer Distance, Earth Mover's Distance, and Hausdorff Distance, that all measure the discrepancy between two sets, as illustrated in Fig. 2. Another common metric for evaluating 3D reconstruction solutions is the Intersection over Union (IoU). Furthermore, the formulas in this section denote false positives, false negatives, true positives, and true negatives as FP, FN, TP, and TN respectively.

- The Chamfer Distance (CD) [30] measures the distance between two different surfaces or sets of points by first calculating the distances between predicted points and their ground truth nearest neighbors, and then averaging all of these distances. The calculated value represents the dissimilarity between predicted output and ground truth. The lower the value, the better the result. Let S_1 and S_2 be two point clouds that represent the predicted and ground truth shapes, and x and y be two points that belong to these point clouds respectively. Then, the Chamfer Distance is defined as:

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} ||x - y||_2^2 + \sum_{y \in S_2} \min_{x \in S_1} ||x - y||_2^2 \tag{1}$$

- The Earth Mover's Distance (EMD), also known as the Wasserstein distance in mathematics and optimization theory) [30, 118, 119], is based on solving an optimization problem, called the transportation problem. The transportation problem attempts to find the least-expensive flow of goods from suppliers to consumers, while satisfying the consumers' demand. For calculation of the EMD of two point sets, each point in one set should be assigned to a unique point in the other set to fulfill optimal assignment. EMD uses bijection between the points that minimizes the total sum of the pair-wise distances. Consider $S_1 \subseteq R^3$ and $S_2 \subseteq R^3$ to be two point sets of equal size, representing the predicted and ground truth shapes, respectively. The EMD [30] is defined as:

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} ||x - \phi(x)||_2 \tag{2}$$

where $\phi : S_1 \to S_2$ is a bijection.

- The Hausdorff Distance (HD) considers the farthest and largest dissimilarity between predicted output and ground

TABLE I: A comparison of benchmark datasets

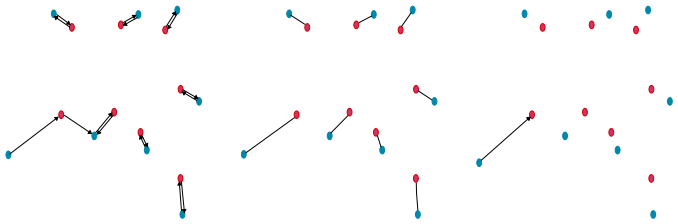
Name	Count/Size	Dataset Type	Representation	Scene Type	Source	DL Tasks
ShapeNetCore	51,300 3D models from 55 object categories	Synthetic	Mesh	Indoor and outdoor objects	CAD model	Shape recognition, reconstruction, retrieval
ShapeNetSem	12,000 3D models of 270 object categories	Synthetic	Mesh	Indoor and outdoor objects	CAD model	Shape recognition, reconstruction, retrieval
PartNet	573,585 part instances of 26,671 3D ShapeNet models in 24 object categories	Synthetic	Mesh and point cloud	Indoor object parts	CAD model	Part-level understanding
ModelNet	127,915 3D models with 662 object categories	Synthetic	Mesh	Indoor and outdoor objects	CAD model	Recognition, reconstruction, generation, and completion
KITTI	Around 49,000 frames from 5 categories	Real-world	Image and point cloud	Outdoor	RGB and LiDAR	Stereo, optical flow, visual odometry, SLAM, 3D object detection, and object tracking
Semantic KITTI	23,201/20,351 scans with 4549 points from 28 classes	Real-world	Point cloud	Outdoor	LiDAR (MLS)	Semantic segmentation and scene completion
ScanNet	2,5 million frames from 1500 RGB-D scans	Real-world	Image and mesh	Indoor	RGB-D Sensor	Object classification, voxel labeling, model retrieval, and reconstruction
Matterport3D	10,800 views from 90 scenes	Real-world	Image and mesh	Indoor	RGB-D Sensor	Scene understanding, normal prediction, classification, semantic segmentation, and reconstruction
NYU depth v2	1449 RGB-D images consisting 464 diverse scenes across 26 scene classes	Real-world	Image	Indoor	RGB-D	Segmentation
Sun3D	415 sequences captured for 254 different spaces in 41 different buildings	Real-world	Image and point cloud	Indoor	RGB-D sensor	Scene understanding, reconstruction, and segmentation
Sun RGB-D	10,335 RGB-D images from 47 scene categories consisting about 800 object categories	Real-world	Image	Indoor	RGB-D sensor	Scene understanding, semantic segmentation, object detection, orientation, and classification
Sydney urban objects	631 scans from 26 object categories	Real-world	Point cloud	Outdoor	LiDAR	Classification and recognition
ABC	1 million 3D models	Synthetic	Mesh	Indoor	CAD model	Feature detection, shape reconstruction and surface normal estimation
Semantic3D	4 billion points in 8 class labels	Real-world	Point cloud	Outdoor	LiDAR (TLS)	Classification and semantic segmentation
H3D	Around 73 million points and 3,5 million faces in 11 classes	Real-world	Point cloud and mesh	Outdoor	LiDAR	Semantic segmentation

3D-Front	18,968 rooms with 13,151 furniture objects from 31 scene categories	Synthetic	Mesh	Indoor	CAD model	3D scene understanding, reconstruction, and segmentation
3D-FUTURE	20,240 images of 5,000 different rooms	Synthetic	Mesh	Indoor	CAD model	2D instance segmentation, 3D object pose estimation, image-based 3D shape retrieval, 3D reconstruction from a single image, and texture recovery for 3D shape
SensatUrban	4 billion points in 13 semantic class labels	Real-world	Image and point cloud	Outdoor	UAV Photogrammetry	Urban-scale point cloud understanding

truth. A point in one set that has the worst mismatch and maximum distance from its nearest point in the other set, determines Hausdorff Distance.

$$d_{HD}(S_1, S_2) = \max\{ \max_{x_i \in S_1} \min_{y_j \in S_2} ||x_i - y_j||, \max_{y_j \in S_2} \min_{x_i \in S_1} ||x_i - y_j|| \} \quad (3)$$

The metric is, however, not very robust towards outliers.



(a) CD (b) EMD (c) HD

Fig. 2: Visualization of Chamfer Distance, Earth Mover's Distance, and Hausdorff Distance metrics. Red dots and blue dots belong to two different point sets and each of these metrics measures the distance between these two sets in a unique way.

- The Intersection over Union (IoU), also known as the Jaccard Index, is often used as quality measure in object detection and semantic segmentation. As illustrated in Fig. 3, it is defined as the overlap between the prediction and the ground truth, divided by their union. The lower the IoU, the worse the prediction result.

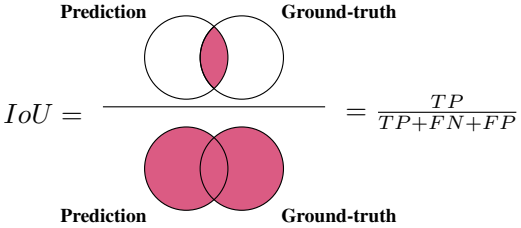


Fig. 3: Visual intuition of the IoU metric.

IoU can also be easily utilized for evaluating voxel-based representations and specifying the overlap between a reconstructed 3D voxel and its voxelized ground truth. For volumetric approaches, IoU can be formulated as [20]:

$$IoU = \frac{\sum_{i,j,k}[I(p_{(i,j,k)} > t) I(y_{(i,j,k)})]}{\sum_{i,j,k}[I(I(p_{(i,j,k)} > t) + I(y_{(i,j,k)}))]} \quad (4)$$

where $I(.)$ is an indicator function, $p(i,j,k)$ is the predicted voxel occupancy probability, t is a voxelization threshold, and $y(i,j,k)$ is the ground truth occupancy probability.

- In classification problems, precision is the number of predictions correctly assigned to one label, i.e., true positives, divided by the number of all predictions assigned to that label, including those identified incorrectly, i.e., false positives (Fig. 4).

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

The Average Precision (AP) is computed by averaging all precision values of all positively labeled samples [99]. The mean Average Precision (mAP) is the average of AP calculated over all classes. For point clouds, precision is calculated as the percentage of predicted points that are close to the ground truth surface, i.e., with the distance less than a specific threshold [120].

- Recall or sensitivity denotes the ratio between the number of predictions correctly assigned to one class (TP) and the actual number of elements in that class, including those that are incorrectly assigned to the other label (FN) (Fig. 4). It is a measure of how well a DL model can find all labels of one class:

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

For point clouds, recall is calculated as the percentage of points on the ground truth that are close to the predicted surface, i.e., having a distance less than a specific threshold [120].

- The F_1 score, also known as balanced F-score, F-measure or Dice Similarity Coefficient (DSC), is the harmonic mean of precision and recall. The higher the value, the better the result.

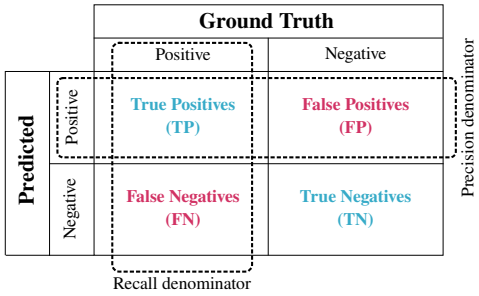


Fig. 4: Confusion matrix for binary classification.

$$F_1 score = 2 * \frac{(Recall * Precision)}{(Recall + Precision)} \tag{7}$$

For point clouds, precision and recall can be calculated by checking the percentage of points in one point cloud, for instance the predicted point cloud or the ground truth, that can find a neighbor from the other point cloud within a threshold [38]. Intuitively, F-score can be interpreted as the percentage of points that were reconstructed correctly [120].

- In classification problems, the Accuracy (Acc) is the ratio between correct predictions and all predictions, i.e., it shows how much of the data is labeled correctly.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{8}$$

However, it is not an appropriate metric for imbalanced datasets as it does not take into account the distribution skew [121].

- Normal Consistency [45] is defined as the mean absolute dot product of the surface normal of each point, i.e., a perpendicular vector to the surface at the given point, in one mesh and the surface normals of its nearest neighbors in the other mesh.

$$\begin{aligned} NC(\hat{M}, M) &= NormalConsistency(\hat{M}, M) \\ &= \frac{1}{2|\partial\hat{M}|} \int_{\partial\hat{M}} |\langle n(p), n(\pi_2(p))\rangle| dp \\ &+ \frac{1}{2|\partial M|} \int_{\partial M} |\langle n(\pi_1(q)), n(q)\rangle| dq \end{aligned} \tag{9}$$

where $\partial\hat{M}$ and ∂M are predicted and ground truth mesh surfaces, $n(p)$ and $n(q)$ are unit normal vectors on these mesh surfaces respectively, $\pi_2(p)$ and $\pi_1(q)$ indicate the projections of p and q on the aforementioned surface meshes respectively, and $\langle .,. \rangle$ implies the inner product. The higher the normal consistency, the better the result.

- The Jensen-Shannon Divergence (JSD) [31] measures the similarity between marginal point distributions. It is mainly based on the Kullback-Leibler (KL) divergence [122]. Considering two point clouds and a voxel grid that discretizes 3D space, the number of points within each voxel from the predicted point set P and the ground truth point set G are counted. The JSD between the obtained empirical distributions (P_P,P_G) is calculated as:

$$\begin{aligned} JSD(P_P||P_G) &= \frac{1}{2} D_{KL}(P_P||M) \\ &+ \frac{1}{2} D_{KL}(P_G||M) \end{aligned} \tag{10}$$

where $M = \frac{1}{2}(P_P + P_G)$.

- Coverage [31] quantifies the fraction of points in the ground truth set S_2 that are matched to points in the predicted set S_1. A match happens when a nearest neighbor in the ground truth set is found for each point in the predicted set.

$$Coverage(S_1, S_2) = \underset{Y \in S_2}{arg\,min} \frac{D(X, Y)|X \in S_1|}{|S_2|} \tag{11}$$

where $D(.,.)$ or "nearness" is measured using distance metrics such as CD or EMD. High coverage indicates that most of the points in S_2 are roughly present within S_1. However, this does not assess the quality of the predicted set. Achieving a perfect coverage is possible, despite large distances between the predicted point set and the ground truth set [33].

- Minimum Matching Distance (MMD) [31, 33] is a complement to the coverage metric. It measures the distance between every point in the ground truth set S_2 and its nearest neighbor in the predicted set S_1 and averages these distances in order to evaluate the quality of the predicted set.

$$MMD(S_1, S_2) = \frac{1}{|S_2|} \sum_{Y \in S_2} \min_{X \in S_1} D(X, Y) \tag{12}$$

where $D(.,.)$ is measured using distance metrics such as CD or EMD.

- Light Field Descriptor (LFD) [123] measures visual similarity between 3D shapes. In short, LFD assumes that a 3D object can be represented as a number of 2D views; therefore, if two 3D models are similar, they also look alike from all views. A light field, which is used in image-based rendering, is defined as a five dimensional function that represents the radiance at a given 3D point along a given direction. To extract LFD for a 3D model, a set of image renderings (silhouettes) are obtained from different angles. These rendered images are acquired using cameras located on the vertices of a fixed regular dodecahedron, i.e., 20 vertices, that surrounds the 3D model. Each of these silhouettes is then encoded both by a region shape descriptor (Zernike moments descriptor) and a contour shape descriptor (Fourier descriptor) for similarity comparisons. A visual representation can be found in Fig. 5. LFD is a good visual similarity metric for

3D surfaces; however, by rendering merely the silhouette of the shape without lighting, LFD can only observe the condition of this shape on the edge of the silhouette [49]. D_A, which is the dissimilarity between two 3D models, is calculated as:

$$D_A(L_1, L_2) = \min_i \sum_{k=1}^{10} d(I_{1k}, I_{2k}), \quad i = 1..60 \quad (13)$$

where i indicates different rotations between camera positions for two 3D models and I_1k and I_2k are corresponding images for i-th rotation. The dissimilarity between two images is denoted by d.

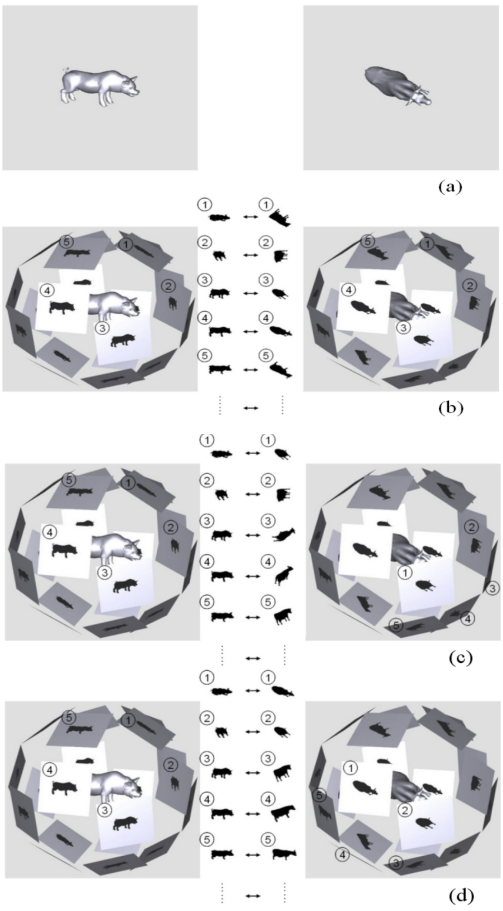


Fig. 5: Comparison of Light Field Descriptors between two 3D models, a pig and a cow (a). First, rendered images are extracted for both 3D models. Then, as illustrated in (b), all 2D images from the same views are compared and a similarity value for this camera angle is obtained. Next, a different mapping between rendered images of the two 3D models is chosen and thus, another similarity value is extracted, as illustrated in (c). Eventually, the rotation of camera positions with the best similarity is found, as shown in (d). The similarity between the two 3D models is attained by summing up the similarities from all the corresponding images [123].

VI. Deep Learning-based 3D Surface Reconstruction

Deep learning-based 3D surface reconstruction approaches can be broadly classified into four main categories according to their representation, as illustrated in Fig. 6:

- Volumetric representations define a surface via small cuboids, either a dense 3D voxel grid [20–25], or an octree [26–29]. Dense voxels are the 3D analogue of a pixel in 2D space, i.e., a cubical element in a regularly spaced 3D grid. Therein, octrees are obtained by recursively splitting 3D space into octants, i.e., eight equally-sized cells. In this data structure, only cells containing information by being close to the surface boundary, are subdivided. Neighboring cells that have the same value do not need to be subdivided and all of these areas can be represented by a single large octree cell. In order to achieve finer details, the space can be further partitioned into smaller octree cells, that is the main difference between a regular voxel grid and an octree.
- Point-based representations utilize the constituting surface points to mark a shape [30–33]. The entire surface is described through an unordered set of (x, y, z) coordinates.
- Mesh-based representations describe an object through vertices, edges, and faces [34–44]. Existing approaches can be mainly divided into three categories:
  - Patch-based approaches attempt to reconstruct the final shape by learning a group of mappings from 2D squares to 3D patches and putting together these small patches.
  - Deformable template-based approaches deform the vertices of a template mesh with predefined interconnections and predict the final shape based on it.
  - Other mesh generation methods are so unique, yet singular, that they are sorted into a catch-all category.
- Implicit neural representations describe a shape as a neural network that takes any (x, y, z) coordinate as input and maps it to occupancy or signed distance value [45–56], or model radiance or appearance properties of an object such as NeRF-based approaches [57–68]

Accordingly, we summarize and discuss the existing literature for deep learning-based 3D surface reconstruction methods based on these categories in the following sections. Furthermore, we depict the architecture of different approaches with a unique color scheme in these sections. In the figures, data units are represented in red, trainable units in blue, and computing units in orange.

A. Volumetric Representations

Volumetric approaches in neural networks for 3D surface reconstructions rely on describing the object through a grid. By extending the concept of 2D convolutions to 3D, a grid can be easily processed using learning-based approaches such as neural networks.

Volumetric methods characterize 3D object data using 1) a regular 3D voxel grid, i.e., dense voxels, 2) or an octree, i.e., sparse voxels.

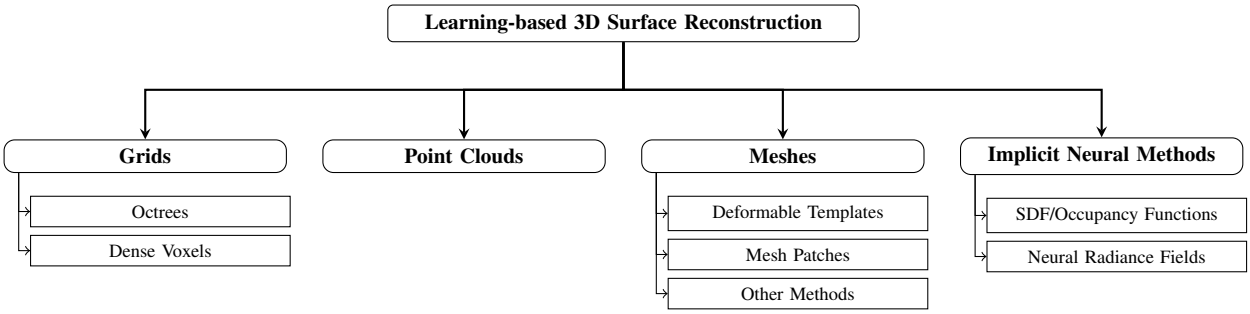


Fig. 6: A taxonomy of learning-based reconstruction approaches based on 3D shape representation.

Analogously to the concept of a pixel in the 2D world, a voxel is a cubical element in a regularly spaced 3D grid. An octree can be built by recursively subdividing the space into octants until a pre-defined maximum depth is reached. Additional information can be stored in cubic cells (both in dense and sparse voxels) to help reconstruct surfaces, as follows:

1) signed distance functions (SDF) express the distance between the center of each voxel and the closest point on the surface of an object. They can be stored in a cuboid by calculating distance functions (DF) [25, 124]. SDFs, a variation of distance functions, purely calculate the signed distance value for each cell. Truncated signed distance functions (TSDF) [125] go beyond the SDF definition by specifying a truncation threshold for SDF values stored in cuboids, i.e., assigning a fixed value to voxels that are not near enough to the surface and their signed distance values exceed the defined threshold.
2) occupancy or indicator functions indicate whether a cuboid is occupied by the surface of an object or not.

Learning voxel-based SDF representations is usually rather complicated compared to occupancy representations, since dealing with distance functions in 3D space is more difficult than simply classifying a voxel as occupied or unoccupied [45]. However, voxel-based SDF approaches provide the advantage of generating smoother surfaces compared to occupancy grid-based approaches. A general disadvantage of voxel-based methods is their resolution limitation by the underlying 3D grid. Mesh extraction approaches, such as the classical Marching Cubes algorithm [126], can be used to infer a mesh from the final output of these methods.

1) Dense Voxels:

Majority of approaches with dense voxel-based representation voxelize the 3D space in order to apply 3D Convolutional Neural Networks (CNNs) on a grid directly. In this section, we first present pioneer studies that applied CNNs to a 3D representation, i.e., dense voxels, for shape classification and then introduce 3D surface reconstruction and shape completion approaches that use dense voxels.

Volumetric CNNs for 3D Shape Classification

Several studies have focused on solving shape classification and recognition tasks using dense voxels [21, 23, 99, 127–130]. One of the pioneers in building deep learning models in 3D world is 3D ShapeNets, as proposed by Z. Wu *et al.* [99]. They were among the first authors to show the application of CNNs to a 3D representation. The introduced architecture uses a convolutional deep belief network for representing a 3D shape as a probabilistic distribution of binary variables on a 3D voxel grid. 3D ShapeNets is able to conduct several tasks, from shape recognition to reconstruction and completion, as well as next-best-view prediction. The DL model takes a single-view depth map of the physical object as input, and converts it into a volumetric representation. The occupancy status of each cell is specified by classifying it as either free space, unknown space or observed surface. Next, a deep belief network is trained on this grid of size 30^3. In terms of accuracy, precision, and recall metrics, 3D ShapeNets outperforms several baseline methods for 3D shape classification and retrieval, such as Light Field descriptor (LFD) approach [123] and Spherical Harmonic descriptor (SPH) [131], even though it utilizes a mesh at lower resolution. It was further shown that the DL model is able to automatically learn general 3D features.

Maturana *et al.* introduced VoxNet [127] which voxelizes input point cloud data and processes the grid with a 3D CNN for object recognition tasks. The authors utilized a volumetric grid for representing the estimated spatial occupancy, and a 3D CNN for extracting features and predicting class labels directly from the occupancy grid of size 32^3. Each point in the input point cloud is mapped to discrete volume coordinates. The resulting voxel volumes are fed to the proposed shallow neural network. VoxNet has fewer parameters compared to 3D ShapeNets [99], i.e., less than one million vs. over 12.4 million parameters, while achieving 8% and 6% higher average accuracy for ModelNet10 and ModelNet40 datasets respectively. However, in both these methods, the memory and computational costs increase cubically with respect to the input resolution.

ORION [128], which is based on VoxNet [127], studies the importance of object orientation in 3D object recognition results. Unlike VoxNet and 3D ShapeNets [99], which augment training data with rotations of the objects to achieve rotational invariance of the network, ORION seeks to predict object orientation. The proposed network uses 3D convolutional net-

works for 3D recognition and adds an auxiliary orientation loss for better classification performance. By forcing the network to predict object orientation in addition to class label during training, more accurate classification results can be achieved at test time. The ORION network is shallower than the proposed method by Brock *et al.* [21] that we discuss further down this survey, leading to fewer trainable parameters.

Some studies utilize multi-view CNNs for analyzing a 3D shape. Multi-view CNNs work in three steps: 1) rendering a 3D shape as a collection of images from different viewpoints, 2) inferring features for each viewpoint, and finally 3) fusing these features across various views. In order to minimize the performance gap between multi-view CNNs and volumetric CNNs, Qi *et al.* [129] suggested two new network architectures of volumetric CNNs. One architecture focuses on local regions, while the other uses anisotropic probing kernels for convolving a 3D cube, then projecting 3D volumes to a 2D image and afterwards applying image-based CNNs for classification. The proposed CNNs surpass volumetric CNN-based methods, such as 3D ShapeNets [99] and VoxNet [127]. Moreover, their classification accuracy competes with some multi-view-based methods, such as MVCNN [132], LFD approach [123], and SPH [131], given the same 3D resolution of 30^3.

3D Surface Reconstruction and Shape Completion using Volumetric Representation

In this section, we review the studies that leverage dense voxel representations for 3D surface reconstruction [20–23] and 3D shape completion [24, 25]. C. B. Choy *et al.* [20] introduced a framework, 3D Recurrent Reconstruction Network (3D-R2N2), for both single- and multi-view 3D reconstruction. This method takes one or more RGB images of an object from arbitrary viewpoints as input and outputs a 3D occupancy grid. The proposed network is composed of three main modules, as shown in Fig. 7: 1) a 2D convolutional neural network (2D-CNN), which encodes input into a low-dimensional feature vector, 2) a 3D convolutional long short-term memory (LSTM) [133], in which the 3D-LSTM units keep their previous cell states or update them, whenever there are more observations, i.e., multi-view images, available, and 3) a 3D deconvolutional neural network (3D-DCNN) that decodes the 3D-LSTM hidden states into a higher resolution and produces final occupancy grid.

In the LSTM module, 3D-LSTM units are located in a grid structure in such a way that each of them focuses on reconstructing a particular part of the output. Two versions of 3D-LSTMs, 3D-LSTMs without output gates and 3D gated recurrent units (GRUs), were tried out in 3D-R2N2, in which the latter achieved better results. The output size is 32^3. Although the generation of detailed and thin parts of the objects and reconstruction of objects with high texture levels are very challenging, 3D-R2N2 performs better than the category-specific approach proposed by Kar *et al.* [134], which learns 3D shapes using camera viewpoint estimations together with object silhouettes, in single-view reconstruction using real-world images. 3D-R2N2 is also able to produce accurate outputs compared to the Multi View Stereo (MVS) method [135] in multi-view reconstruction.

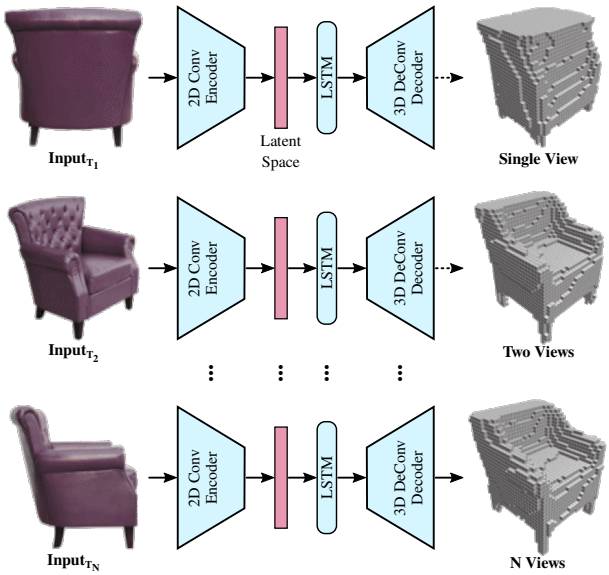


Fig. 7: Overview of 3D-R2N2 network [20]. The input to this network is one or more RGB images from arbitrary viewpoints and the output is a 3D occupancy grid. The main modules of 3D-R2N2 are an encoder, a 3D LSTM, and a decoder.

Brock *et al.* [21] inverstigated generative and discriminative voxel modeling with deep ConvNet architectures. In short, their method presents a voxel-based variational autoencoder (VAE) [136, 137] for reconstruction and interpolation, a graphical user interface for investigating the latent space of autoencoders, and a deep voxel-based convolutional neural network for object classification. The output size of the network is 32^3. The voxel-based VAE learns to reconstruct features of an object, attaining acceptable reconstruction accuracy. It further facilitates the transition from one object to another by interpolating between their reconstructions. The neural model has significantly fewer parameters than FusionNet [130], i.e., 18 million as opposed to 118 million. Nevertheless, it achieves competitive results compared to ORION [128] considering that ORION uses orientation augmentations to improve classification.

The TL-embedding network [22] learns a vector representation of an object, which is both generative in 3D, i.e., able to reconstruct objects in 3D space from this representation, and predictable from 2D images, i.e., able to extract this representation from images. As shown in Fig. 8, this architecture is composed of a convolutional network, which brings about the predictability, and an autoencoder, that results in generativeness. It generates outputs with 20^3 resolution. This method captures stylistic details better than the method proposed by Kar *et al.* [134].

J. Wu *et al.* introduced a framework, called 3D generative adversarial network (3D-GAN) [23], which generates novel volumetric 3D objects from a probabilistic latent space. 3D-VAE-GAN, an extension of 3D-GAN, provides the ability to reconstruct surfaces from input images. For generation and recognition of 3D objects, this method

utilizes both general-adversarial modeling [138, 139] and volumetric convolutional networks [99, 127], as illustrated in Fig. 9. Furthermore, it fuses 3D-GAN with a variational autoencoder [136] for 3D object reconstruction from a single 2D image. The resolution of its final output can reach up to 64^3. The classification accuracy of this network is roughly similar to volumetric learning-based approaches such as VoxNet [127] and ORION [128], but is lower than the method proposed by Qi *et al.* [129]. It shows higher average precision for voxel prediction compared to the work by Girdhar *et al.* [22] in single image 3D reconstruction task. However, 3D-VAE-GAN usually creates a noisy and incomplete output from an input image. Studies conducted by J. Wu *et al.* [140] showed that ultimately, training GANs together with recognition networks can lead to high instability.

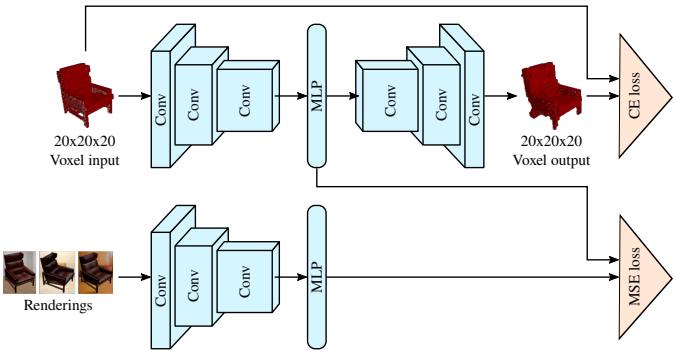


Fig. 8: The TL-embedding network [22]. During training, two types of input are fed to the network: 2D RGB images as the input to ConvNet at the bottom, and 3D voxel maps as the input to the autoencoder on the top. The network outputs a 3D voxel map.

Stutz *et al.* [25] introduced a learning-based approach with weak supervision for 3D shape completion. It takes a 3D bounding box and an incomplete point cloud as input and predicts the complete object shape. The completion process is done in two steps: 1) a shape prior is learned, i.e., a variational auto-encoder (VAE) is employed to learn a 3D shape model on synthetic data, encoding shape models in a dataset using occupancy grids and SDFs at $24 \times 52 \times 24$ resolution, and 2) shape inference is performed. For this, 3D shape completion is considered as a maximum likelihood (ML) problem. The authors used the amortized maximum likelihood (AML) approach that works over the lower-dimensional latent space z from the first step. It keeps the pre-trained decoder from the previous step fixed and adds a new encoder. The encoder is trained without supervision, i.e., without using explicit labels, and learns to directly predict ML solutions from incomplete input observations using maximum likelihood loss. The presented method was shown to be faster than a fully-supervised baseline while using 9% or less supervision, while being able to produce competitive results.

Dai *et al.* [24] fused a volumetric deep neural network with a 3D shape synthesis procedure to complete partial 3D inputs. Their approach generates the output in two major stages: 1) a shape prediction method, which predicts a volumetric grid with 32^3 resolution as a low-resolution global structure of the input. The proposed network, 3D-Encoder-Predictor Network (3D-EPN), consists of 3D convolutional layers and attempts to predict distance field values for missing data, and 2) a patch-based 3D shape synthesis method, which employs a synthesis procedure to improve local details and create a high-resolution output using CAD model priors. Given the predicted coarse output from the first stage, the authors carried out a search for similar 3D shape models in the ShapeNet [97] database. Based on the results, they sought to find similar local patches in these shape models for the purpose of local detail synthesis. The resolution of the final voxel grid is 128^3. Without the synthesis step, 3D-EPN provides only low resolution and is unable to predict local details and fine structures. Nevertheless, it outperforms 3D ShapeNets [99] as well as poisson methods [8, 9].

In another approach, Dai *et al.* suggested Sparse Generative Neural Networks (SG-NN) [141], which is a self-supervised scene completion approach that accepts an incomplete RGB-D scan as input and predicts a high-resolution 3D reconstruction while also inferring unseen, missing geometry. The self-supervised nature of this technique allows for training entirely on real-world, partial scans. This eliminates the requirement for synthetic ground truth. Self-supervision is achieved by removing some frames from a given (incomplete) RGB-D scan, resulting in an even more incomplete input; this input is used to create an input-target pair (original scan is considered as the target scan). The difference in partialness is then correlated in this input-target pair while regions that have never been observed are masked out during training. Despite the fact that fully complete scenes are not used as samples during training, this approach generates high levels of completeness by learning to generalize completion patterns across the training set. Dai *et al.* also proposed a sparse generative neural network, a fully-convolutional encoder-decoder architecture, capable of predicting high-resolution final geometry as a sparse TSDF representation. This end-to-end formulation generates a 3D scene in a coarse-to-fine manner. SG-NN is built upon sparse convolutions [142] that operate only on surface geometry. This self-supervised approach produces more accurate and complete scenes in comparison to a fully-supervised approach such as 3D-EPN [24].

In general, voxel-based methods encounter a number of difficulties. Information loss may occur due to discretization and transformation of input data to coarse voxels. Moreover, cubic growth in memory limits the resolution and the overall computational demands bring about coarse final outputs. Generating higher resolution surfaces requires deeper networks. However, the network depth is constrained by the available GPU memory. Therefore, it may affect the ability of CNNs with volumetric decoders in producing high resolution outputs [143].

2) Octrees:

Dense voxel representations are associated with a number of challenges regarding resolution, memory, and computational complexity. In many cases though, the 3D shape surface

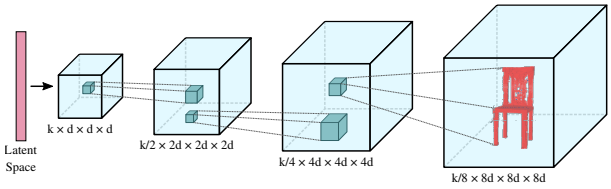


Fig. 9: The generator architecture in 3D-GAN [23]. Five volumetric fully convolutional layers of kernel sizes $4 \times 4 \times 4$ and strides 2 make up the generator. The discriminator architecture is usually mirroring the generator architecture.

occupies only a small portion of 3D space. Hence, octrees mark a popular approach for partitioning space, as they allow for the 3D data to be stored in a sparse structure [144, 145]. For octree construction of a 3D shape, a bounding cube is created around the entire shape. This bounding cube will be recursively subdivided. In each step, all cuboids which are occupied by a shape boundary, are traversed and each of them is divided into eight smaller, equally-sized cuboids. However, in order to enable CNN operations on an octree, this data structure needs to be updated and slightly changed, which leads to complex implementations while the resolution is still limited by the underlying 3D grid [45]. Hence, convolutions and pooling to octrees are applied similarly to CNN operations on dense voxels with the main difference being that the elementary operand is an octant.

OctNet

Riegler *et al.* presented OctNet [146], which enables the usage of high-resolution inputs for deep learning purposes. OctNet is based on a 3D convolutional neural network that can be applied to a special form of octree data structure to learn representations from high-resolution 3D data. Vanilla octree implementations might encounter data access speed issues in high resolution (high recursion-depths) octrees. On the other hand, for convolutional network operations such as convolution or pooling, it is crucial to have frequent access to different data elements, such as cell neighbors. In order to provide faster data access and reduce cell traversal time, the authors proposed a hybrid grid-octree data structure. They used a shallow octree, which is an octree with maximum depth $D = 3$, as a basic building block. Several of these shallow octrees are stacked in a regular grid structure to cover the whole volume. Input resolution effects of this representation were evaluated on three different task: 3D classification, 3D orientation estimation of unknown object instances, and semantic segmentation of 3D point clouds. For high resolution inputs in the 3D shape classification task, OctNet runs faster and requires less memory as opposed to DenseNet, a densely voxelized version of OctNet. In general, both OctNet and DenseNet perform better than a shallow network such as VoxNet [127], verifying that network depth is of great importance.

OctNet does not generate an octree structure, and this structure has to be known in advance for both input and output. In classification and semantic segmentation tasks, this does not comprise a problem. However, learning volumetric structure of objects and scenes and being able to construct them is crucial in generative tasks such as reconstruction, generation, and completion, since the input and output partitioning structure might be different. OctNetFusion [26] proposes a learning-based approach, which learns to partition the space and can predict a SDF or a binary occupancy map. The network takes one or more 2.5D depth maps as input. To reconstruct precise and complete 3D outputs, it fuses depth information from different viewpoints into a coarse volumetric grid. Then, this volumetric grid (grid-octree structure) is fed to the OctNet-Fusion network architecture, consisting of encoder-decoder modules. The network determines whether a cell should be subdivided or not in a coarse-to-fine manner. The output resolution can be up to 256^3. This approach performs qualitatively and quantitatively better than traditional volumetric fusion approaches, such as vanilla TSDF fusion [125] and TV-L1 fusion [147] for volumetric fusion tasks and Voxlets [148] for volumetric shape completion from a single image.

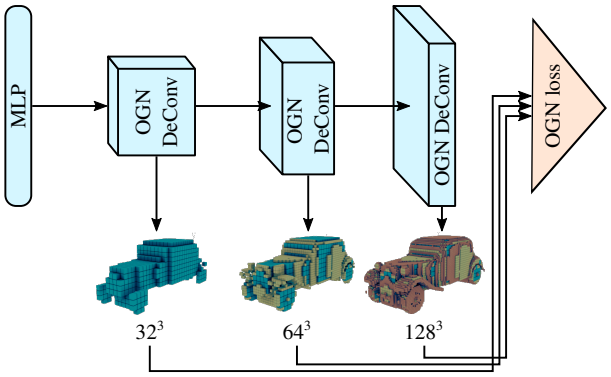


Fig. 10: Octree Generating Network (OGN) [29] takes an input 3D shape and gradually reconstructs octrees as the output in different resolutions.

O-CNN

Another concurrent work in the scope of octree-based CNNs for 3D shape analysis is the Octree-based Convolutional Neural Network (O-CNN) [149]. The authors' main idea is to represent 3D objects with octrees and execute 3D CNNs only on nodes or cuboids, which are occupied by boundries of the 3D object, instead of sliding the convolutional kernel over the whole voxel grid, as done for the standard convolution computation in full voxel grids. The network constructs an octree from an input oriented 3D model, e.g., an oriented triangle mesh or a point cloud with oriented normals, and enriches each octant of this data structure with meta-information, such as shuffle key vectors, label vectors, and input signal, which are needed for the convolution operations. Furthermore, a hash table is built to accelerate neighborhood search in the convolution. By storing the octree data structure into the graphical memory, O-CNN can be easily and efficiently trained and evaluated on GPUs. To demonstrate the efficiency of their network, the authors evaluated it on three shape analysis tasks: object classification, shape retrieval, and shape segmentation. In terms of classification accuracy, O-CNN performed better than VoxNet [127], slightly worse than the

method proposed by Brock *et al.* [21], and competitive to non-voxel based methods such as PointNet [150]. Additionally, the impact of different input representations on the same network architecture (O-CNN) was investigated. Results showed that an octree input achieves higher accuracy compared to full voxel structures. For object part segmentation, O-CNN yields better or comparable performance than other methods such as PointNet [150].

To improve computation and memory efficiency of O-CNN, P.-S. Wang *et al.* proposed the extension "Adaptive O-CNN" [27], which consists of an encoder-decoder structure and uses patch-guided adaptive octree shape representations. Contrary to approaches like volumetric-based CNNs, where the output is gerenated as voxels of the same resolution, this method can generate adaptive octrees based on a patch-guided partitioning strategy and with differently-sized planar patches. The underlying assumption is the subdivision rule, which states that splitting all octants to the finest level is not necessary. The process can be stopped early for some of the octants and the local shape inside these octants can be represented by simple patches, e.g., planar patches. However, this approach limits the quality of the output and may encounter some difficulties in generating watertight and curved surfaces. Adaptive O-CNN obtains better or comparable classification accuracy than PointNet [150], OctNet [146], and O-CNN [149], yet it performs worse than PointNet++ [151], Kd-Network [152], and the method proposed by Brock *et al.* [21]. For the task of shape reconstruction from a single image, Adaptive O-CNN surpasses PointSetGen (PSG) [30] and AtlasNet [34] in generating more detailed geometry.

Other Octree Prediction Approaches

Häne *et al.* introduced a hierarchical surface prediction (HSP) [28] framework for high resolution voxel grid prediction in 3D object reconstruction. The main idea boils down to generating and predicting high resolution voxels around the predicted surface and coarse resolution voxels for interior and exterior parts of an object. The high resolution voxels are not predicted directly, but instead, a coarse-to-fine approach is used to create smoother 3D models hierarchically and in a multi-resolution fashion. Starting with approximating the coarse geometry of the output, more finely resolved details are added step by step by refining the surface. This process finally results in a voxel grid with up to 256^3 resolution. The proposed method is based on an encoder-decoder architecture. A convolutional encoder encodes input to a feature vector and then an up-convolutional decoder predicts the voxel grid or final data structure (called voxel block octree data structure in the paper). Classifying each voxel as boundary, free space, or occupied space, only voxels with a boundary label require high resolution prediction, since they cover the actual surface. The major difference between HSP and OctNet [146] is that OctNet takes the structure of the shallow octrees as input, while HSP predicts the structure of the tree together with its content. HSP produces more accurate surfaces with higher resolutions compared to low resolution baselines predicting dense voxels.

In a similar approach [29], Tatarchenko *et al.* suggested Octree Generating Network (OGN) which is a convolutional decoder that can generate and predict octree structure of 3D shapes, along with the occupancy value of each cell. It operates on octrees and reconstructs 3D shapes in a multi-resolution manner, as illustrated in Fig. 10. This method generates results up to a resolution of 512^3. The network gradually reconstructs a high-resolution surface from the initial, low-resolution dense voxel grid using hash-table-based octree blocks. If the reconstructed surface has not yet reached the final output resolution, cells with "mixed" state, i.e., undetermined state, will be passed to the next layer of the network for further subdivision. Providing the same accuracy as dense voxel grids in low resolutions, OGN offers less memory consumption and shorter run-time in higher resolutions in comparison to voxel grid-based networks. In particular, it is 20 times faster and requires two orders of magnitude lower memory usage at 512^3 resolution.

B. Point-based Representations

These days, point clouds are becoming increasingly important and available due to the improvements in scanning devices in recent years. A point cloud is a set of points in 3D space, inferred by various 3D data acquisition techniques. It is an irregular data format, since there is no canonical order between the points in a set. Each point can be defined by its (x, y, z) coordinates. Therefore, the size of the matrix representing a 3D object is initially $N \times 3$ for N points. The number of columns in this matrix representing the features might be extended, if other information such as color, normal, etc. exist. Considering the irregular and unordered nature of point clouds, it is difficult to apply deep learning techniques, such as CNNs, directly on them. Consequently, in order to process a point cloud with neural networks, it was common to transform them to voxel grids or collections of images. These transformations usually present numerous challenges such as information loss, voluminous data, resolution constraints, and high computational cost. To reduce the overhead of data transformation to other data formats, different methods for effectively processing point clouds with neural networks have been proposed, which will be discussed in the following sections.

PointNet and PointNet++

Pioneer works in the field of learning global features directly on point clouds are PointNet [150] and PointNet++ [151]. PoinNet as proposed by Qi *et al.* directly consumes a raw point cloud as an input and uses it for discriminative deep learning tasks, e.g., object classification, semantic segmentation, and part segmentation. As illustrated in Fig. 11, each of the points in the input set is processed by a small neural network individually and independently based on its own coordinates, resulting in a high dimensional embedding of the points. Following the embedding step, a simple symmetric function, such as max pooling, is utilized to aggregate the encodings

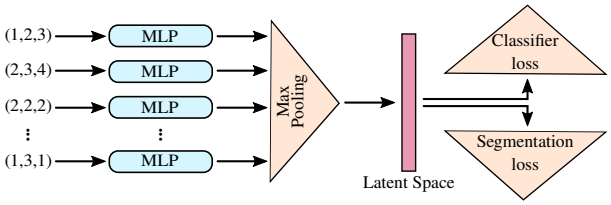


Fig. 11: PointNet architecture [150], which is used for classification and segmentation tasks, directly accepts a point cloud as input. Each of the points in the input point cloud is processed by a small neural network individually and independently. Then point features are aggregated by max pooling, a simple symmetric function that respects the permutation invariance of the input points. The aggregation step creates a global feature vector that encodes the entire shape.

from each of the points. The symmetric function is chosen such that it pays attention to the permutation invariance of the input points. The aggregation step brings about a global feature vector, which encodes the whole shape and can be fed to different neural networks for recognition purposes. PointNet achieves higher classification accuracy compared to LFD approach [123], which is a 3D model retrieval method, Spherical Harmonic descriptor (SPH) [131], and other methods with volumetric representation, such as 3D ShapeNets [99], VoxNet [127], and another method previously proposed by Qi *et al.* [129]. Although it has around 17 times fewer parameters than multi-view-based methods like MVCNN [132], its performance is only slightly lower compared to these methods. PointNet provides linear complexity $O(N)$ in both spatial and temporal domains, where N is the number of input points, while the complexity grows squarely with respect to image resolution for multi-view methods, and cubically with respect to the volume size for volumetric methods. More importantly, due to it satisfying the permutation invariance condition, PointNet cannot capture local information and thus, lacks generalization.

In order to resolve the issues of PointNet, Qi *et al.* introduced the extension PointNet++ [151], which pays more attention to local features and combines them with global features to infer better results. The architecture is built on top of PointNet, enriching it with a hierarchical feature learning approach. The whole process, which is done recursively, can be summarized as follows: 1) specifying centroids of local regions by sampling a subset of the input point cloud using the farthest point sampling (FPS) algorithm, 2) finding local neighborhoods of these centroid points using radius-based ball query, and 3) applying a mini-PointNet in each neighborhood to mimic the concept of a convolution kernel and conduct convolution-like operations in point space for the purpose of local feature extraction. The presented method proved to be robust towards non-uniform sampling density, which might occur due to perspective effects, variations in radial density, motion, etc. Compared to PointNet, PointNet++ has an improved classification accuracy for the ModelNet40 dataset.

Point Cloud Reconstruction and Generation

PointNet was mainly implemented for discriminative tasks, such as classification and segmentation. The first approach for reconstructing a 3D point cloud of an object from a single (monocular) RGB or RGBD image was proposed by Fan *et al.* [30] and is based on a generative learning-based approach. The main contributions of this work were: 1) designing a point set generator network, 2) proposing two proper loss functions for the comparison of the ground truth with the network's predictions for point sets, i.e., Chamfer Distance and Earth Mover's Distance, and 3) modeling uncertainty and ambiguity of the ground truth. The proposed network is composed of an encoder and a predictor part. The encoder transforms the input into an embedding space. The predictor is divided into two parallel branches: a deconvolution (deconv) branch and a fully-connected (fc) branch. The deconvolution branch learns the smooth parts and main body of the object, while the fully-connected branch learns non-smooth parts and details. The results of these branches are then concatenated to create the final point set. In comparison to 3D-R2N2 [20], which generates a volumetric representation from single or multi-view images, this method produces better results on CD, EMD, and IoU metrics. In addition, it is able to reconstruct thin structures more accurately.

Achlioptas *et al.* [31] proposed a solution for generative tasks and unsupervised representation learning, based on an end-to-end pipeline that can reconstruct point clouds using deep autoencoders (AE) and GANs. The autoencoder extracts features by learning a lower-dimensional representation of the input, based on which the GAN[138] generates point clouds. In the autoencoder architecture, the authors exploited a PointNet-like encoding scheme to learn compact representations. The encoder generates a latent code which is invariant to the order of input points. The latent code is converted back to a point cloud using a standard deep network with three fully connected layers as decoder. The authors further investigated three different approaches for point cloud generation: 1) GAN operating on raw point cloud, 2) latent-GAN, which is a plain GAN being trained on the latent space of the pre-trained AE, and 3) Gaussian mixture models operating on the latent space learned by AE. The study indicated that the proposed AE provides good generalization capacity towards unseen data. However, the output of the proposed DL model architecture is limited to 2048 points and generating high-quality surfaces with such a small number of points is challenging.

Another closely related approach that attempts to solve unsupervised learning challenges using deep auto-encoders is FoldingNet [32]. The presented architecture, as illustrated in Fig. 12, utilizes a simple graph-based scheme as the encoder part (similar to the method proposed in [153], an improved and generalized version of PointNet) in order to encode local neighborhood structure information. Since applying convolution operations on graphs is difficult, the authors suggested building the k-nearest neighborhood graph (K-NNG) and repeatedly applying max-pooling operations on each node's neighborhood. This way, the DL model is able to capture locality and extract features of neighboring points. For the decoder

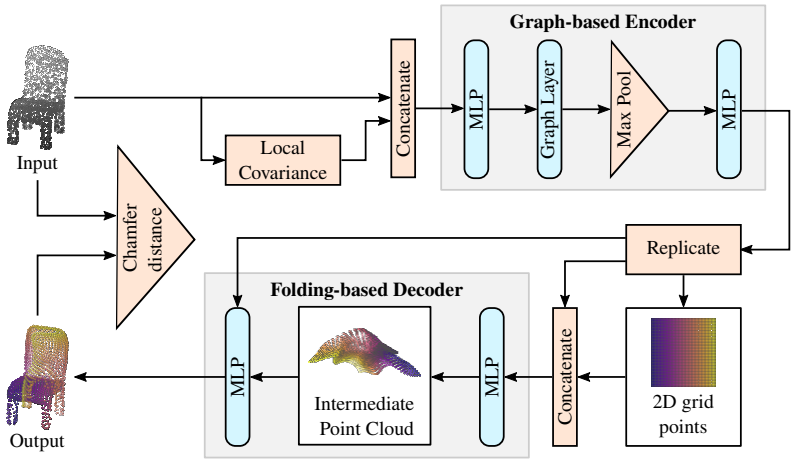


Fig. 12: FoldingNet architecture [32] consists of a graph-based encoder (an improved and generalized version of PointNet), that encodes local neighborhood structure information, and a folding-based decoder, that reconstructs the point cloud from a 2D grid template deformation process.

part, a folding-based scheme is proposed to reconstruct the point cloud from a 2D grid template deformation process. Due to the fact that 3D point clouds are often sampled from object surfaces, one can make the assumption that any 3D object surface can be converted and squeezed into a 2D plane. It is also possible to reverse this process, i.e., wrapping 3D shapes by a fixed 2D paper (plane). This property builds the foundation of the proposed method. The decoder maps 2D points from a 2D template grid to the surface of the 3D object using folding operations. The definition of the folding operations, i.e., 2D-to-3D mapping, is the main contribution of the paper , making it the first single learned parametric function embedding from a (grided) 2D (point) manifold into 3D space and a fundamental building block for other surface reconstruction approaches. FoldingNet's decoder requires about 7% of the parameters of the fully connected decoder proposed by Achlioptas *et al.* [31], which is significantly smaller than the latter. However, it was shown to perform better at feature extraction in terms of classification accuracy and reconstruction loss. Overall, FoldingNet achieves higher classification accuracy than other unsupervised methods, such as LFD approach [123], SPH [131], TL-embedding network [22], and 3D-GAN [23].

PointFlow [33] is a 3D point cloud generation framework that learns a distribution of distributions, i.e., distribution of shapes and its respective points. A variational auto-encoder (VAE) is applied to transform sampled 3D points from the point prior into a realistic point cloud conditioned on a shape vector. The distributions is modeled in two steps: First, the distribution of the latent space of shapes is learned. To enable the method to sample multiple shapes, PointFlow extracts latent vectors of different shapes. A sampled Gaussian vector (a shape prior) is transformed into a shape latent vector using a continuous normalizing flow (CNF) [154–156]. In the second step, the distribution of points on a specific shape is learned for shape generation. Given a sampled 3D Gaussian point cloud (point prior) and a shape latent vector inferred from the first step, a CNF is used to move input points to their new location and transform them into the target shape. For generative tasks, PointFlow outperforms the methods proposed in [31] in terms of 1-nearest neighbor accuracy (1-NNA) metric, while having fewer parameters. With respect to the EMD score, it achieves a better auto-encoding performance compared to Achlioptas' method [31] for point cloud reconstruction from inputs.

Several recent studies have investigated point cloud upsampling [157–159], normal and curvature estimation from point clouds [160, 161], classification [162–168], segmentation tasks [162–164, 166, 168–170], object detection [165, 171], and point cloud denoising [172]. Although point-based representation approaches discretize the surface of the shape into a set of 3D points, they do not model the corresponding connectivity. Thus, additional post-processing steps are needed to generate final high quality 3D mesh. On the other hand, existing approaches are very limited in terms of the number of generated points leading to limited output quality.

C. Mesh-based Representations

Meshes are irregular type of data that are difficult to predict by neural networks. Their components are vertices, edges, i.e., pairs of vertices, and (triangular) faces, i.e., triplets of vertices. Therefore, researchers have investigated different paths to address mesh-based representations, namely patch-based approaches [34, 37], deformable template-based approaches [38–41, 70, 173], and other mesh generation methods [35, 36, 42–44, 174]

1) Patch-based Approaches:

Groueix *et al.* introduced a method for 3D surface generation, called AtlasNet [34], as illustrated in Fig. 13. They suggested to generate a 3D surface and represent it as a set of folded 2D squares. The input shape can be either a 2D image or a 3D point cloud. The method outputs the corresponding 3D mesh and its atlas parameterization. The main steps of the approach include encoding an input 3D point cloud into a 3D shape and reconstructing the 3D shape from an input

RGB image. 3D point clouds are encoded using a PointNet-based encoder, which transforms the input point cloud into a 1024-dimensional latent vector. Input images are encoded using ResNet-18 [175]. The decoder consists of four fully-connected layers, which extract the final surface. The target 3D surface is estimated using multi-layer perceptrons (MLPs), which learn the local mapping of 2D-points to 3D-surface points. Therefore, by transforming the 2D squares to the 3D surface using learnable parametrizations, i.e., MLPs or patches, the final surface is covered in a way similar to putting paper strips on a shape to make a papier-mâché. The difference between the proposed method and FoldingNet [32], which is a folding-based method, is that FoldingNet deforms just one 2D square or patch while AtlasNet investigates varying number of 2D squares. Results from AtlasNet showed that usage of multiple patches improves 3D reconstruction. For single-view reconstruction from a 2D RGB image, AtlasNet yields qualitatively better performance compared to the dense voxel-based method 3D-R2N2 [20], the octree-based method HSP [28] and a point-based method [30]. Furthermore, it was shown that AtlasNet provides good generalization properties, however, it generates artifacts such as self-intersecting parts and overlapping patches.

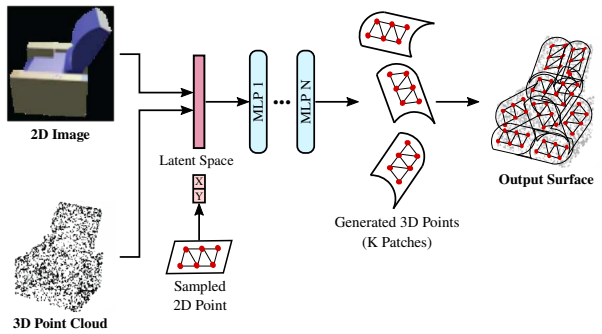


Fig. 13: AtlasNet [34] is a patch-based approach that takes either a 2D image or a 3D point cloud as input and outputs a 3D mesh. Multi-layer perceptrons (MLPs) are used to estimate the target 3D surface, which learn the local mapping of 2D-points to 3D-surface points.

Badki *et al.* proposed an approach [37] to extract a 3D mesh from a noisy, sparse, unordered and non-oriented set of points. Instead of learning shape priors at the object level, the method learns them locally while enforcing global consistency. In order to represent these priors and local features, small mesh patches, called meshlets, were used. These meshlets can be interpreted as a dictionary of local features and learned priors. The final mesh is the union of all meshlets. The authors used a variational autoencoder for learning the priors by using a very large dataset of meshlets, that was extracted from objects in the ShapeNet dataset. During training, the local priors are learned with meshlets. At inference, meshlets are deformed to match the input point cloud via distance minimization. Since individual meshlets are updated independently in order to adapt to the points, the overall mesh extracted from their union is not watertight. Therefore, a global consistency step is performed to eliminate inconsistencies across all meshlets, as illustrated in Fig. 14. Compared to Occupancy Networks [45] and AtlasNet [34], which are class-specific algorithms that learn priors at the object level, and Deep Geometric Priors [176], this method produces better quantitative results in terms of CD and Hausdorff Distance metrics. It also performs qualitatively well at reconstructing objects from unseen classes during training, coping with noise, and being robust to dramatic changes of the object's pose.

For all the aforementioned methods, mesh patches and the tessellation process may affect the quality of final surface, especially for complex shapes. Therefore, these approaches may generate self-intersecting meshes and might be unable to generate closed surfaces.

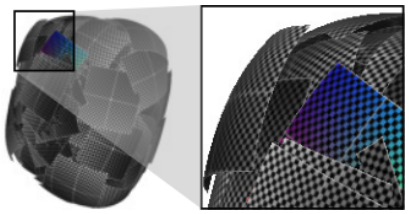


Fig. 14: Meshlet inconsistencies adapted from the patch-based approach paper [37].

2) Deformable Template-based Approaches:

Deformable template-based approaches take a template mesh with predefined interconnections as input, deform the vertices and predict the final shape based on this. These approaches can generally reconstruct meshes and shapes with simple topology, however, they struggle to generate complex structures with a lot of details. N. Wang *et al.* [38] designed Pixel2Mesh, an end-to-end reconstruction pipeline for extracting a 3D triangular mesh from a single RGB image. Taking an input image and an ellipsoid with fixed numbers of edges and vertices as initial mesh, it gradually deforms the mesh using a graph-based convolutional neural network (GCN) to generate the final 3D shape. As illustrated in Fig. 15, the overall method is composed of two main parts: 1) an image feature network (2D CNN), which is used to infer perceptual features using an input color image, and 2) a three block cascaded mesh deformation network (graph-based ResNet), that takes care of initial mesh deformation in a coarse-to-fine manner. Each graph-based ResNet block takes the perceptual feature concatenated with 3D feature encoding of the input mesh as input. In their study, the authors showed that Pixel2Mesh outperforms 3D-R2N2 [20] and the point-based method proposed by Fan *et al.* [30] in terms of mean of F-score, Chamfer Distance, and Earth Mover's Distance metrics. Quality-wise, it produces smoother surfaces with local details. Nevertheless, the approach shows generalization issues and can only generate meshes and objects of topologies similar to the initial mesh.

Pixel2Mesh++ [39] works along with Pixel2Mesh to produce 3D meshes from multi-view images. The main idea is that adding more images (3-5) of an object as input provides more information for a shape generation method and thus results in more accurate and detailed reconstructions. Pixel2Mesh++

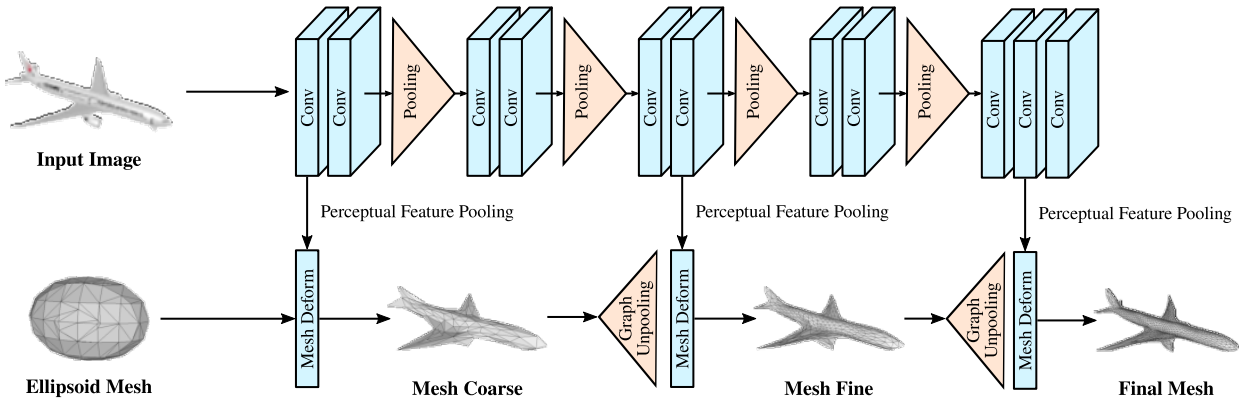


Fig. 15: Pixel2Mesh network [38] is a deformable template-based approach that reconstructs a 3D triangular mesh from a single RGB input image. It consists of three mesh deformation blocks used for mesh resolution enhancement and vertex location estimation.

consists of a multi-view deformation network (MDN), which processes cross-view information for the prediction of optimal deformations. First, a coarse mesh is produced by Pixel2Mesh, which is then fed to the MDN part to be refined progressively by adding details. With regard to the F-score metric, Pixel2Mesh++ generates better results than 3D-R2N2 [20], Learnt Stereo Machine (LSM) [177], and two other baselines that the authors implemented using Pixel2mesh [38]. In addition, it generalizes well across various semantic categories and produces high quality outputs with accurate details.

Recent efforts by Kanazawa *et al.* [40] utilized a CNN image encoder followed by three modules for 3D shape generation, camera pose estimation, and texture prediction. The CNN acts as encoder, producing a latent representation of a single input image, which is fed to the three prediction modules. The 3D structure of a shape is generated by deforming a learned category-specific mean shape with instance-specific predicted deformations. Texture is parameterized as an UV image which is predicted using texture flow. This mechanism enables the method to transfer the texture of one instance onto another. However, it cannot produce the detailed structure of input shape. The presented approach obtains comparable results to the one proposed by Kar *et al.* [134] in terms of the intersection over union (IoU) metric. Kar *et al.* [134] exploited segmentation masks and optionally a set of keypoints as annotations during inference to generate 3D rigid objects. Contrary to that, the method of Kanazawa *et al.* only utilizes these annotations during training and directly predicts a 3D structure form an unannotated input image at inference time.

Hanocka *et al.* introduced Point2Mesh for reconstructing meshes from point clouds [41]. The core idea is a mesh fitting process for reconstruction of the final mesh. In addition to the input point cloud, an initial watertight mesh is fed to the network. This initial mesh represents a coarse approximation of the point cloud, which is iteratively deformed from outside-in using a convolutional neural network to fit to the input point cloud, as illustrated in Fig. 16. Accordingly, a network learns displacement and deformation of the mesh vertex positions. The optimization of Point2Mesh is based on MeshCNN [178], which is a CNN-based pipeline applied on triangular meshes. Unlike Screened Poisson Surface Reconstruction, Point2Mesh is agnostic to normal orientation and ensures watertight reconstructions from noisy input with missing parts and unoriented normals. It also achieves a higher F-score compared to Screened Poisson Surface Reconstruction [9] and Deep Geometric Priors [176] for shape denoising and completion. However, Point2Mesh requires a large amount of compute time and memory, possibly alleviated by data- or model-parallelism [179].

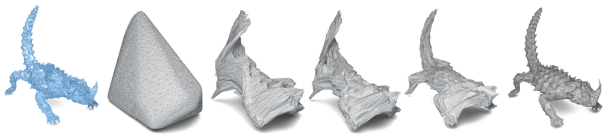


Fig. 16: Point2Mesh [41] takes a point cloud (in blue) and a deformable initial mesh as input and gradually reconstructs the final output shape.

3) Other Mesh Generation Methods:

Liao *et al.* investigated end-to-end 3D surface prediction using a Differentiable Marching Cubes algorithm (DMC) [42]. In previous research, surface prediction was solved in two steps: first, predicting an intermediate SDF/occupancy representation using an auxiliary loss, and second, taking a post-processing step for 3D mesh extraction separately, such as the Marching Cubes (MC) algorithm. On the other hand, applying backpropagation to the Marching Cubes algorithm is intractable due to non-differentiability. Hence, in order to unite these steps to create an end-to-end framework, the authors inserted a differentiable formulation as a final layer into a 3D convolutional neural network. A point cloud, which is used as input, is directly converted into a volumetric representation using a grid pooling operation, e.g., max pooling in each cell. An encoder-decoder network with skip connections is then used to process pooled features, with the decoder operating in volumetric space. That way, it not only estimates occupancy probabilities, but also predicts the vertex displacement field

for a surface mesh. When compared with baseline methods that infer occupancy or TSDF first and then apply marching cubes as a post-processing step, DMC achieves superior results with respect to Chamfer Distance, accuracy and completeness metrics. Nevertheless, difficulties may arise while reconstructing very thin surfaces and disconnected parts can become connected.

Scan2Mesh [43] is a generative model that combines convolutional and graph neural network architectures to predict a complete, lightweight and structured 3D mesh representation from an unstructured and incomplete range scan of an object. The aim is to predict both vertex location and edge. Initially, the features space is computed through a set of 3D convolutions from input TSDF. The vertices are then predicted based on the extracted features. A fully connected graph is generated from the predicted vertices and all of the vertices are connected to each other via edges. Next, a graph neural network is used to classify edges and extract the ones that belong to the mesh graph structure. Using this intermediate graph of predicted edges and vertices, a dual graph is created which comprises a set of valid potential faces. Finally, another GNN is applied to predict the final face structures from the dual graph. Scan2Mesh offers better qualitative and quantitative performance compared to 3D ShapeNets [99], 3D-EPN [24], and Poisson Surface Reconstruction [8, 9]. However, it depends on fully connected graphs for predicting edges, which leads to limitations in model size.

Mesh R-CNN [44] is an approach that unifies both 2D perception and 3D shape prediction. It takes a single RGB image as an input, detects 2D object instances in the image, and creates a category label, bounding box, segmentation mask, and 3D mesh predictions of the detected objects as the outputs. Mesh R-CNN utilizes Mask R-CNN [180], an end-to-end region-based 2D object detector, for the detection of 2D objects. The 3D shape prediction step depicted in Fig. 17 is based on a hybrid approach which primarily produces a coarse voxel representation of a detected object, transforms this voxelization into an initial 3D triangular mesh, and finally refines this mesh by modifying the vertex positions using a graph convolution network. This approach achieves better results compared to a voxel-based method such as 3D-R2N2 [20], a point-based method [30], and a mesh-based method such as Pixel2mesh [38] in single-image shape prediction considering Chamfer Distance and F1-score metrics.

M. Liu *et al.* [35] attempted mesh reconstruction from input point clouds by fully utilizing the input and simply adding connectivity to the existing points. Towards this end, they introduced a deep point cloud network which proposes candidate triangles and predicts faces. This information is provided as input to a mesh generation module. First, a k-nearest neighbor (k-NN) graph is build for each point in the input point cloud, in order to decide which three points should form a triangle face and infer candidate triangles proposals. Next, a MLP network is employed to classify candidate triangles and filter out incorrect triangles, such as the ones that connect two independent but spatially adjacent parts of the shape, using Intrinsic-Extrinsic Ratio (IER). To infer the local connectivity between vertices comprising a triangle, the ratio of geodesic distance (intrinsic metric) and Euclidean distance (extrinsic metric) was proposed. Finally, in a post-processing step the remaining candidate triangles are sorted and merged in a greedy way to generate the final mesh. The approach outperforms several learning-based methods, such as AtlasNet [34], Deep Geometric Priors [176], Deep Marching Cubes [42], and DeepSDF [50], as well as traditional reconstruction methods, such as Poisson Surface Reconstruction (PSR) [8, 9], Marching Cubes [126], and Ball-Pivoting Algorithm (BPA) [10] in terms of F-score, Chamfer Distance, Normal Consistency metrics. Moreover, it generates higher quality outputs with fine-grained structures than the aforementioned methods and offers the capability to be transferred to unseen categories.

Daroya *et al.* [36] proposed a recurrent neural network (RNN)-based method, called Recurrent Edge Inference Network (REIN), to produce triangulated surface meshes from sparse input point clouds using a bottom-up approach. The network tries to predict edges sequentially and generates a mesh by processing points one at a time from a queue of points. The latent vector of the input point cloud, which is inferred by a PointNet-based [150] autoencoder, is also used to enrich the data with global structure information of an object. For edge prediction, the authors relied on the application of recurrent networks, inspired by GraphRNN [181]. An RNN can be a good choice for inferring sequential predictions based on previous states [182]. To tackle memory issues of processing large point clouds, small sections of the input point cloud are fed into the network one at a time, instead of processing all of it at once. In each small section, points in the queue are processed consecutively by REIN in two steps: 1) edge prediction: REIN tries to predict connections, i.e., edges, between the new vertex (which was chosen from the queue) and the current partially predicted mesh. Two RNNs are used for edge prediction, State RNN and Edge RNN. State RNN encodes the current state of the graph with its nodes and edges, given a point cloud and its latent vector as input. Edge RNN attempts to predict sequence of edges considering the current state. 2) face generation: all of the vertices and predicted edges are investigated to form faces. However, the face generation module encounters problems generating surfaces from edge predictions, especially for non-manifold surfaces. Qualitatively and quantitatively, REIN produces better mesh surfaces than Ball-Pivoting Algorithm (BPA) [10] and Poisson Surface Reconstruction (PSR) [8].

D. Implicit Neural Representation

Neural networks are universal function approximators [183], hence they can be used to approximate any measurable function, including signed distance function (SDF) and occupancy/indicator function, or to model other properties such as radiance fields. Neural networks that parameterize such implicitly defined functions, without explicitly parameterizing the surface or properties of interest, are considered implicit neural representations [51].

Similar to implicit functions stored in discretized voxel grids, different functions can provide geometric information

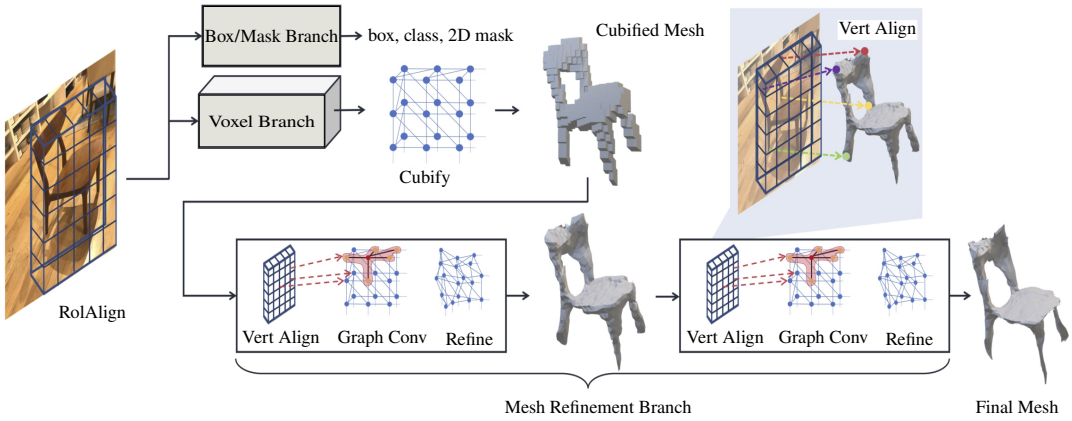


Fig. 17: Mesh R-CNN [44] architecture. After the object detection step, the voxel branch predicts a coarse voxel representation for each object detected by Mask R-CNN [180]. Then in the mesh refinement branch, the cubified object is transformed into the mesh after a series of refinement steps.

for parametrizing a surface by a neural network [124]. There are also other functions that focus on capturing surface-related properties such as appearance, texture, or reflectance properties. In particular, these functions can be as follows:

- Level set methods define a distance function f on the entire point set and then extract the zero level set $f = 0$ as the boundary of an input object, as illustrated in Fig. 18. They divide a 3D space threefold into an interior part, an exterior part, and an exact overlap with the object's surface. Given a point (x, y, z), the function f calculates the distance of this point to the boundary of the object, specifies its sign (SDF) [184], and decides the location of the point w.r.t the surface. The sign indicates whether a point is inside or outside of the surface. Therefore, in contrast to SDFs stored in voxels that discretize 3D space and store SDF value in each voxel, SDFs in implicit neural representation are calculated for each point individually using a neural network. DeepSDF [50], which will be explained further down in this survey, was the first paper to propose this approach.
- Occupancy functions model an approximate likelihood of whether a point is occupied by part of an object or not. This can be expressed as a binary classification problem to classify a point as occupied or unoccupied. The approach can be interpreted as a special case of SDF that only considers the sign of SDF values [50]. Occupancy Networks [45] and IM-NET [49] fall into this category and will be clarified subsequently.
- Radiance fields refer to a set of techniques that aim to model the radiance or appearance properties of an object or scene. Notable examples of these methods include NeRF [57] and its variants and subsequent sections will provide thorough explanations of them.

1) Implicit Neural Representation based on Variants of SDF or Occupancy Function:

The key idea behind these implicit neural representations is to represent a shape as a neural network that takes a point in space as input and outputs some property of that space, i.e., mapping it to occupancy or signed distance of the shape at that coordinate. However, implicit neural representations cannot directly derive detailed 3D shape features. Thus, an extraction step is needed to infer a corresponding explicit representation, such as a mesh. A possible isosurface extraction approach is the classical marching cubes algorithm [126].

Compared to voxel-based representations, the memory cost of implicit neural representations remains constant with respect to the resolution. However, the capability of these methods to reconstruct fine details is constrained by the capacity of their underlying network architectures [51].

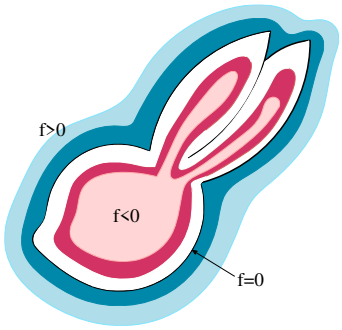


Fig. 18: Level set methods divide a 3D space into three parts: an interior part ($f < 0$), an exterior part ($f > 0$), and an exact overlap with the object's surface ($f = 0$).

As mentioned previously, Occupancy Networks, IM-NET, and DeepSDF [45, 49, 50] represent pioneer works in implicit neural representation concurrently. Mescheder *et al.* introduced a new representation for 3D geometry, called occupancy networks [45], which can predict the continuous occupancy function using a neural network for the extraction of 3D meshes. As illustrated in Fig. 19, the occupancy function is approximated with a deep neural network that determines an occupancy probability value between 0 and 1 for every possible point in 3D point space (similar to a neural network

for binary classification). The mesh is then generated from the occupancy network by utilizing a simple Multiresolution Isosurface Extraction (MISE) algorithm, which employs octree structures and the marching cubes algorithm [126]. This expressive approach does not require discretization of 3D space. The representation can be inferred from different kinds of input, such as single images, noisy point clouds, and coarse discrete voxel grids, and can encode various structures efficiently. In comparison to methods using different 3D representations, such as 3D-R2N2 [20] (a voxel-based method), point set generating networks [30] (a point-based method), and Pixel2Mesh [38] and AtlasNet [34] as mesh-based techniques, occupancy networks shows competitive qualitative and quantitative results for various inputs, e.g., single images, noisy point clouds and coarse discrete voxel grids.

In a similar fashion, Z. Chen *et al.* [49] attempted to solve 3D shape analysis and synthesis problems by proposing an implicit field decoder (IM-NET), that is based on the application of binary classifiers. Based on two inputs, a point coordinate and a feature vector encoding a shape (extracted from a shape encoder), IM-NET specifies whether the point is inside or outside the surface, using only the sign of its signed distance function. They utilized their proposed implicit decoder as the decoder part of some conventional frameworks (such as autoencoders (AEs) and generative adversarial networks (GANs)) and proposed IM-AE and IM-GAN respectively. IM-AE and IM-GAN can be used for both 3D reconstruction and shape generation tasks. Based on visual results, IM-AE generates smoother and high-quality surfaces compared to a classical 3D CNN-based decoder implementation, operating on voxelized shapes. IM-GAN showed better performance compared to AtlasNet [34] (in which output quality is constrained by the number of generated points) and 3D-GAN[23] (low coverage). For single-view 3D reconstruction task, the proposed framework constructs higher quality results than AtlasNet [34] and HSP [28]. However, applying the implicit decoder on each point in the training set increases training time considerably. In addition, the network does not generalize well to other categories, since it is trained individually for each shape category.

With DeepSDF [50], a novel shape representation based on the concept of signed distance functions was introduced. Instead of storing SDF in a discretized regular grid, as done in classical surface reconstruction techniques, the network directly learns continuous 3D models of SDF from point samples. The trained network predicts the corresponding SDF value of the input data, from which the zero level-set surface can be extracted. The zero isosurface can be rendered and visualized through raycasting or polygonization algorithms, e.g., marching cubes [126]. The network takes (x, y, z) coordinates and a shape encoding vector as input to model a dataset of shapes. In order to obtain a meaningful latent space of shapes, an auto-decoder is used for learning a shape embedding without an encoder. One of the advantages of the method is that the network size is considerably smaller compared to the voxel-based methods. DeepSDF outperforms Atlasnet [34] (a mesh-based method) and OGN [29] (an octree-based method) in reconstructing complex topologies with fine details. It further outperforms 3D-EPN [24] (SDFs stored in voxels) for the shape completion task.

Sitzmann *et al.* introduced a novel architecture, called Sinusoidal Representation Networks (SIREN) [51], a fully connected neural network that uses periodic sine as its non-linearity for implicit neural representations. The motivation behind this lies in the fact that many recently published studies on implicit neural representation employing ReLU-based MLPs are incapable of capturing high frequency details of the input signal. There are two possible explanations for this phenomenon: 1) conventional neural network architectures encounter difficulties while learning to apply the same function at two different coordinates and thus, the learned functions are not shift-invariant in general, and 2) ReLu non-linearities cannot parameterize any signal that has information in its second derivative since its second derivative will be zero everywhere. Therefore, the authors suggested to replace conventional non-linearities, such as $tanh$ or $ReLU$, with a periodic sine activation function to improve final results. This replacement results in gaining a certain degree of shift-invariance, and also addresses the problem of the second derivative, since the derivative of sine is a shifted sine itself. The method was applied to a wide variety of areas, including image, audio, and video representations, 3D reconstruction, and solving first-order and second-order differential equations. In the 3D shape reconstruction task, SIREN generates details of complex objects and scenes better than ReLU-based implicit representations, such as NeRF [57].

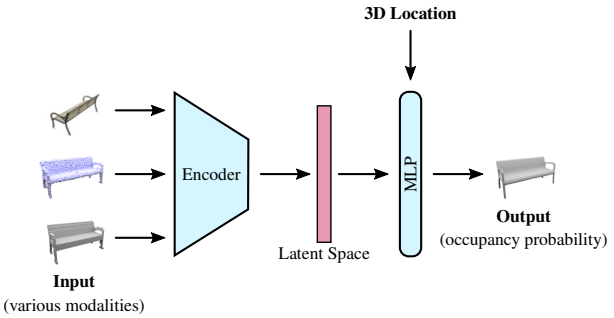


Fig. 19: Occupancy Networks architecture [45] predicts occupancy function for each point in 3D space using a deep neural network. Different encoder architectures are used in Occupancy Networks depending on the task and input. For image input, a ResNet-18 architecture [175], for point cloud input, a PointNet encoder [150], and for voxel input, a 3D convolutional neural network are employed.

Methods based on Unsigned Distances

Some studies exploit unsigned distances instead of occupancy or signed distances for learning representations. With Sign Agnostic Learning (SAL) [52], Atzmon *et al.* proposed a deep learning approach based on raw input data without any oriented normals or signs. Generally, regression-based methods utilize regression loss for training and need inside/outside ground truth information for this process, such as DeepSDF [50] or Occupancy Networks [45]). In contrast to these methods, SAL uses a sign agnostic loss function that

can be directly applied to raw unsigned data. The algorithm generates high quality surfaces in comparison to AtlasNet [34] and a baseline method that approximates SDF based on the work by [9]. D-Faust dataset, which comprises raw scans of humans in various poses, is used for the experiments. Although there is no need to include the signed implicit ground truth representation in the calculation of the loss function during training and also closing surfaces for training data is unnecessary in this work, SAL predicts SDF as the final output, which also results in closing the gaps even in open surfaces and generating only closed outputs (closed surfaces, in this case, is a division of 3D space into three regions: inside, outside, and on the surface of an object, and they do not have separate parts). Neural Distance Field (NDF) [53] is a method to predict the unsigned distance field for 3D surfaces using a neural network. Similar to SAL, NDF does not close shapes during training. However, it can successfully generate open surfaces, shapes with inner structures, and open manifolds compared to IF-Net [56] and SAL [52].

DUDE [54] is another approach, which is able to represent a surface by combining the unsigned distance field with the normal vector field. Evaluation of this method in comparison to DeepSDF and SAL demonstrates its superiority in producing high quality outputs, especially for open surfaces, with visually pleasant renderings. The main difference between NDF and DUDE compared to SAL is that the first two can reconstruct both open and closed shapes with complex and detailed topology, while the latter attempts to close parts that should be open.

Part-based Approaches

Encoding an entire surface into a single latent vector can lead to substantial information loss, since the limited size and capacity of the latent representation causes accuracy and generalization issues [48]. In order to solve the difficulties of generalizing to other shape categories and scaling to large scenes, researchers resort to conditioning an implicit neural representation on local geometric features [46–48, 55, 56, 185, 186]. There are different approaches to the implementation of such conditioning. Some approaches fuse the volumetric representation (voxel grids) with the implicit neural representation and use local features stored in voxels for inferring implicit neural representation [46, 47, 55, 56]. Others use local patches to learn implicit neural functions [48, 185, 186]. All of these methods leveraged the advantages of encapsulating local as well as global information for proposing more generalizable and scalable approaches.

C. Jiang *et al.* suggested the Local Implicit Grid (LIG) [55] representation, which decomposes 3D space into a regular grid of overlapping part-sized local regions, and encodes each region with implicit feature vectors. The key idea behind the algorithm is that objects in different categories share similar geometric features and details at neither micro scale, i.e., a very small patch, nor macro scale, i.e., the entire object, but part scale. Therefore, a part-autoencoder was used to learn embeddings for different parts of an object and extract meaningful abstraction of its shape. The autoencoder consists of a 3D CNN encoder and an implicit network decoder in form of a reduced version of the IM-NET [49] decoder. During inference, a pre-trained implicit function decoder is used in each grid cell, in order to generate the respective scene part. Eventually, the overlapping latent grids were optimized via the proposed mechanism to reconstruct the entire scene. Since this method generalizes shape priors learned from object dataset, it does not need any training on scene level dataset for reconstructing scenes from sparse oriented point samples. Therefore, it generates higher quality outputs from unseen object categories than other methods such as IM-NET [49], since IM-NET learns only a single embedding for an entire object. Compared to traditional surface reconstruction methods such as PSR [8, 9], Local Implicit Grid is capable of recovering thin structures and details very well.

Likewise, Chibane *et al.* introduced Implicit Feature Networks (IF-Nets) [56], which is composed of an encoding and a decoding tandem. The network takes voxels or point clouds as the input and predicts whether point p lies inside or outside of an object, resulting in a continuous surface at arbitrary resolution. To encode local and global structures of a 3D shape, a 3D multi-scale grid of deep features is extracted instead of using a single vector to summarize an entire object. Consequently, rather than classifying (x, y, z) point coordinates directly, the decoder classifies a point based on these extracted features and creates occupancy predictions. IF-NET achieves better quantitative results than occupancy networks [45], point set generation network [30], deep marching cubes [42], and IM-NET [49] in point cloud completion, voxel super-resolution, and single-view human reconstruction tasks. Moreover, Chibane *et al.* [187] proposed an extension of IF-Nets for 3D texture completion.

Peng *et al.* developed convolutional occupancy networks [46], a hybrid voxel grid/implicit neural representation-based approach that combines convolution operations with implicit representations in form of a convolutional encoder with an implicit occupancy decoder. The method is independent of the input representation. Given a point cloud or voxel grid as input, the method uses a 2D plane encoder/3D volume encoder based on PointNet to process the input by converting it into features and projecting these local features onto a plane(s)/volume. A convolutional 2D plane decoder/3D volume decoder further processes the feature plane(s)/volume using 2D/3D U-Nets [188, 189], integrating both local and global information. At the end, a small fully-connected occupancy networks [45] is used to predict the occupancy probability from a given query point p and its feature in 3D space. For rendering and extracting meshes from the input, the Multiresolution Isosurface Extraction (MISE) algorithm is applied during inference. Evaluation of both object-level and scene-level reconstruction was performed using synthetic and real-world data sets. The major difference between the novel method [46] and the original occupancy networks [45] is that convolutional occupancy networks capture the local features of the space as well as global features, leading to higher generalizablility, scalability and faster training. Moreover, it benefits from the translational equivariance property of convolutional networks, while not supporting the rotational

equivariance property.

In similar work [47], Chabra *et al.* introduced Deep Local Shapes (DeepLS), a method for deep shape representation which uses learned local shape priors. As illustrated in Fig. 20, the key idea is the decomposition of a shape into small components, in order to improve reconstruction results. To this end, local information of these components is stored in a grid of independent latent codes. Based on these, SDFs are predicted by applying DeepSDF [50] as local shape neural network to each grid cell. DeepLS outperforms DeepSDF in accuracy and inference time by approximately an order of magnitude.

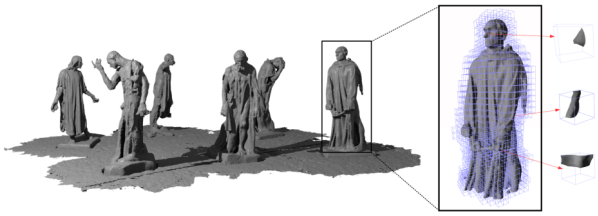


Fig. 20: DeepLS [47] decomposes a scene into local shapes and uses a set of locally learned continuous SDFs defined by a neural network.

Unlike occupancy networks [45] and DeepSDF [50], which extract the global latent code vector from the entire input, local patches are modeled as deep implicit functions in patch-based approaches [48, 185, 186]. Erler *et al.* presented a patch-based learning framework, called Points2Surf [48], that generates accurate implicit surfaces directly from raw point clouds without surface normals. The underlying algorithm is based on the notion of considering a shape as a collection of small shape patches. Instead of representing an entire surface as a single latent vector, Points2Surf creates separate feature vectors for different patches to describe local details in addition to global information. By decomposing the surface reconstruction problem into learning a global function (that learns the sign of SDF), and a local function (that learns the absolute distance field of SDF with respect to local patches), Points2Surf succeeds in being robust to noise and missing parts and also generalizing well to unseen shapes. Additionally, Points2Surf yields a significant drop in the reconstruction error on unseen classes compared to both data-driven and non-data-driven methods such as DeepSDF [50] and AtlasNet [34], or SPR [9]. However, this patch-based approach results in longer computation time, inconsistencies between outputs of neighboring patches, and non-watertight and bumpy surfaces.

There is a growing number of studies based on implicit neural representation for various tasks. Some authors investigated 3D human reconstruction [186, 190, 191], 4D reconstruction [192], and 3D reconstruction of the appearance and texture of surfaces in addition to their geometry with 3D supervision [193, 194] or without 3D supervision [57, 195–197]. These are some recent articles [198–203], which are SDF-based, and some [186, 190, 191, 204] that are based on predicting occupancy probability.

Equivariant Neural Networks

Chatzipantazis *et al.* introduced a SE(3)-equivariant coordinate-based attention network called TF-ONet for 3D surface reconstruction. Local shape modeling and equivariance are the two core design principles of this method. SE(3) stands for Special Euclidean group in three dimensions representing transformations including translations and rotations in 3D. In simple terms, equivariance means that when the pattern in the input changes, i.e., when it is rotated or shifted to a specific direction, the output should also change in an equivalent proportion. TF-ONet works directly on unoriented and irregular point clouds and outputs the occupancy field of a shape. To predict the occupancy score at any given point in space, TF-ONet creates equivariant features for each point that function as keys and values of specialized attention blocks. This enables TF-ONet to output high-quality reconstructions and to generalize to novel scenes composed of multiple objects, despite being trained on single objects in canonical poses. Inspired by SE(3)-transformers [206] and tensor field networks [207], TF-ONet attention modules ensure equivariance by incorporating symmetries into the learning process. It is basically a two-level approach: 1) the first level, i.e., an encoder, applies self-attention in local neighborhoods around each point to infer local features from the point cloud, and 2) the second level, i.e., a cross-attention occupancy network, uses the extracted point features and the coordinates of a query point in space to calculate the value of the occupancy function for the specific query point.

For single object reconstruction tasks, TF-Onet performs comparably better than non-equivariant networks such as occupancy networks [45], convolutional occupancy networks [46], IF-Net [56], and also equivariant networks such as vector neurons [208], and GraphOnet [209] considering evaluation metrics such as Chamfer-L1, F_1-score, and IoU. For scene reconstruction tasks trained only on single objects, global shape modeling-based techniques such as occupancy networks [45] and vector neurons [208] are not able to generalize to scenes containing multiple objects. Moreover, local shape modeling-based methods such as convolutional occupancy networks [46] that are not equivariant under SE(3) transforms, are only able to produce low-quality objects in novel poses. TF-ONet instead excels at the tasks and can generalize to novel scenes with high quality, benefiting both from local shape modeling and equivariant properties.

2) NeRF-based Approaches:

Fundamentals of NeRF

Neural radiance fields [57], commonly referred to as NeRFs, are basically used for view synthesis. The main idea behind NeRFs is to train a model that can produce new views of a scene or an object and can represent them in 3D, given a set of 2D images from different viewing angles as input. Hence, multiple input views of a scene and their corresponding camera poses are used to render new views of that scene by interpolating between the given views. The NeRF method employs a

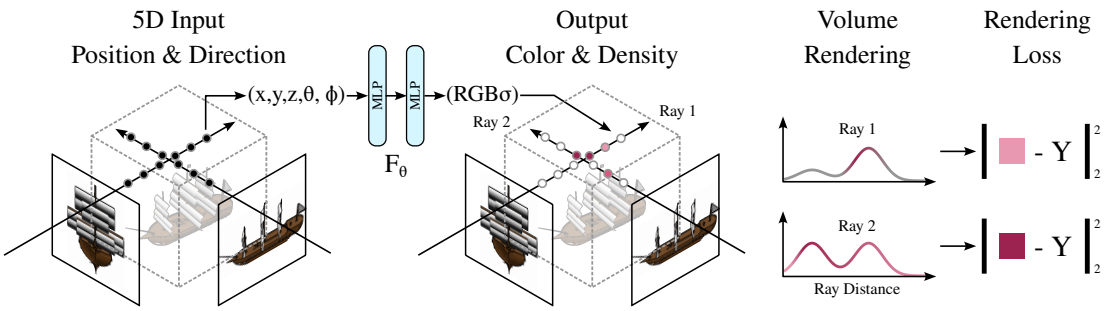


Fig. 21: Overview of neural radiance fields (NeRFs) [57]. The ship in the figure is borrowed from the ShapeNet dataset [97].

fully-connected deep network to represent a scene. Each input (x, y, z, θ, ϕ) is a single continuous five-dimensional coordinate that encompasses spatial position and viewing direction, and each output $(RGB\sigma)$ is density and view-dependent emitted radiance at that particular spatial location. Consequently, the neural network describes an implicit function that exists throughout all locations as a continuous representation without any discretization. As a result, by implicitly encoding density and color through a neural network, NeRF has demonstrated impressive performance on new view synthesis of a particular scene.

Although overfitting is usually an undesirable behavior in machine learning, the key part of this approach is the usage of a neural network that is overfitted to one particular scene and only cares about this specific scene. For rendering a new scene, it is necessary to take a fresh neural network and train it from scratch until it is overfitted to the new scene. Therefore, instead of storing a scene as a mesh or a voxel grid, the scene is stored in the weights of the neural network. For instance, if a scene consists of a tree, the weights represent this tree and are very specific to it, outputting nonsense for another scene if not being trained once again.

To explain the fundamentals of NeRF in more detail according to Fig. 21, the images have to be transformed to 5D coordinates (x, y, z, θ, ϕ)s first. (x, y, z) are coordinates of a pixel point in 3D space and (θ, ϕ) are related to the viewing angle. For each pixel on an image, a ray is sent through. Therefore, every pixel in every input picture defines a ray and then it is sampled along the ray. Consequently, each input image sends out a lot of rays, and for each ray, there are many sampled points. Next, for each location represented as (x, y, z, θ, ϕ), the neural network effectively determines the presence of an object and subsequently identifies its corresponding color. This nine-layer fully connected network provides four numbers $(RGB\sigma)$ as the output: the (RGB) is the color of that particular pixel point, and σ is the density for each of the individual points. The density value serves as an indicator of the presence or absence of an object in the designated region of space, as well as its density. If this process is done for all the points in space from all viewing angles, a complete 3D representation of what it looks like can be inferred. The neural network outputs different results for the same location depending on different viewing angles. Accordingly, it can capture the reflections, lighting effects, and transparency. Eventually, classical techniques for volume rendering are employed to project the network outputs onto a two-dimensional image. Given that volume rendering is intrinsically differentiable, it is possible to define a loss function that measures the difference between the predicted and the ground truth color of the ray. In order to convert NeRF to a mesh, marching cubes can be further applied.

To produce high-resolution complex scenes, two interesting tricks were utilized: 1) positional encoding, and 2) hierarchical sampling. Positional encoding, which is similar to the same one in transformers [210], is used to map the 5D input vector to higher dimensional space using $\sin$ and $\cos$ waves, helping MLP in approximating and representing high-frequency functions. It enhances the ability of a neural network to not only capture coarse-grained structures but also to perform well in representing finer details. Hierarchical sampling is a two-step sampling method with two networks: a coarse network, and a fine network. The points on the ray are sampled in a uniformly distant fashion from each other. These sampled points are run through the network for density prediction. Next, an evaluation step is taken place to decide where should be sampled more in the second round, based on the output of the previous step. So the output of the coarse network discloses where the important stuff is. The second round of sampling starts with points with higher density, i.e., points closer to the particular object which is perceived, and the vicinity of such points will be sampled a lot more. Both coarse-grained and fine-grained networks are optimized at the same time using a loss.

Delving into the advantages associated with NeRFs, it is clear that these methods are not view-dependent, without the need for any 3D input supervision. Additionally, NeRFs are memory-efficient compared to voxel grid representation. One neural network of one scene fits into a few megabytes, which might even be smaller than the input image size for that scene, whereas dozens of gigabytes might be needed for storing the same scene in voxels. Regarding the limitations of NeRFs, what makes them impractical is their requirement for a large number of high-quality posed images as input. The more images are fed, the better the output quality will be. Another downside is related to their high computational cost, originating from optimizing each scene individually without sharing knowledge between different scenes [62]. This implies that for every scene, the network should be trained again and a pre-trained one cannot be utilized. For instance, it takes around 100-300k iterations, i.e., roughly 1-2 days, for the naive NeRF network [57] to be trained on a

single scene using a single NVIDIA V100 GPU.

NeRF and its Variants for View Synthesis

This section provides a summary of some of the papers that aim to enhance NeRF and its abilities. In NeRF++ [61], K. Zhang *et al.* analyzed NeRF and uncovered three major problems and situations in which NeRF might fail: shape-radiance ambiguity, near-field ambiguity, and parameterization of unbounded scenes such as large real-world scenes. The first two issues are related to the fact that NeRF is actually overparameterized, i.e., the degree of freedom for NeRF to hallucinate and move toward a completely wrong answer is high. However, the authors of NeRF [57] use an interesting implementation trick and regularization. They feed viewing angles in the very last layers of the MLP network. Therefore, the MLP actually starts with location-wise coordinates of a point in the beginning, and viewing angles are fed in the last layers, resulting in a limited degree of freedom for NeRF. Accordingly, if all 5D coordinates are fed to the network from the beginning, the shape radiance ambiguity becomes a big issue, affecting the quality of NeRF's outputs drastically.

NeRF++ proposes a couple of solutions to tackle these three problems and enhance output quality. By introducing an auxiliary loss, NeRF can avoid moving towards a poor solution which may lead to completely wrong scene geometry estimation, thus addressing the shape-radiance ambiguity issue. Furthermore, adaptive near-field culling is proposed to solve the near-field ambiguity issue. It culls the front part of each view frustum adaptively based on the geometry of a scene, i.e., it prevents estimating the geometry right in front of the camera contrary to vanilla NeRF. The third issue concerns scenarios in real-world settings where precise reconstruction of objects in front of the camera is essential. However, the camera's ability to capture other items beyond these objects necessitates a certain level of reconstruction for the distant items as well. NeRF++ suggests homogenous parameterization that enables having a detailed reconstruction in the foreground as well as a detailed reconstruction of the background. This is done by training two NeRFs, one for the foreground part of the scene, and the other for the background part, increasing the capacity of the model for reconstructing details at different levels. NeRF++ still needs per-scene training and one scene takes about three days to be trained.

PixelNeRF [62] is built upon the concept of NeRFs for 3D reconstruction and synthesizing photorealistic 3D scenes from a single or a small number of posed images. PixelNeRF attempts to tackle the requirement of NeRFs for a lot of images as the input and make it generalizable. Considering the fact that extracting 3D geometry and the appearance of a scene from limited input is a challenging task and NeRFs do not share knowledge between the scenes, the framework proposes to condition a NeRF on spatial image features. Thus, pixelNeRF employs a fully-convolutional image encoder that infers a pixel-aligned feature grid. Then, a spatial location and its corresponding encoded feature are fed to a NeRF network for color and density prediction. PixelNeRF shows better generalization capabilities and performance compared to NeRF. However, its rendering time is still slow, and more input views cause a linear increase in the runtime.

In another concurrent work to overcome the generalizability issue and long optimization time of NeRFs, MVSNeRF [63] suggests a deep neural network that can reconstruct a neural radiance field, given only three nearby input views. This approach combines plane-swept cost volumes, which are used for geometry-aware scene reasoning in multi-view stereo (MVS), with NeRF models. To create a cost volume, MVSNeRF first warps 2D image features onto a plane sweep. Then, a 3D CNN is leveraged for the reconstruction of a neural encoding volume with per-voxel neural features. Next, features interpolated from the encoding volume are employed to predict density and RGB radiance for an arbitrary point using an MLP. Achieving comparable or better rendering results, MVSNeRF can significantly surpass NeRFs [57] in terms of optimization time efficiency, i.e., roughly 30 times faster, if more images are provided as input. Moreover, it generalizes better than PixelNeRF [62] and IBRNet [211].

MipNeRF [212] attempts to address one of the problems of NeRF, which is the production of blurred or aliased renderings when dealing with training or testing images at different scales. In NeRF, all of the cameras have the same distance from an object. Thus, it is able to do view synthesis without the need to think about scaling or aliasing. However, when new cameras are to be added at different scales, NeRF begins to collapse since it is a single-scale model trying to tackle a multi-scale problem. To fix this issue, MipNeRF proposes some modifications to the vanilla NeRF including: 1) casting a cone instead of sending a ray through each pixel, 2) slicing up the cone into conical frustums instead of sampling single points along each ray, 3) computing integrated positional encoding instead of positional encoding of a single coordinate along the ray, and in general 4) training a single neural network that describes the scene at multiple scales instead of training separate neural networks at various scales. These new properties help MipNeRF reason about the scale of its inputs. MipNeRF is capable of producing high-resolution renderings across multiple scales rather than just at a single scale in vanilla NeRF. NeRF's performance decreases when being trained on multi-scale data, while MipNeRF's does not. The number of parameters in MiPNeRF is half of that in NeRF while also being 7% faster for their multiscale dataset. Mip-NeRF360 [213] and ZipNeRF [214] are some other recent methods used for anti-aliasing Neural Radiance Fields.

In a work proposed by NVIDIA, instant NGP [215], Müller *et al.* try to facilitate and speed up neural graphics primitive tasks. A neural graphics primitive is an object represented by a neural network that takes a query as input, such as position and some extra parameters, and outputs appearance and shape attributes. Examples of NGP can be computing signed distance function, NeRFs, radiance cashing, etc. To bring about simplicity, instant training, real-time rendering, and high-quality results for instant NGP, solutions such as multiresolution hash encoding by storing the trainable feature vectors in a compact spatial hash table, using a small neural network called a fully fused neural network, and improvement of training and rendering algorithm are proposed as main

ideas.

The amount of research efforts based on NeRF is increasing. From relighting [64, 65, 216, 217], and view synthesis without pose supervision [218], to learning non-rigid objects and dynamic scenes [66–68, 219–221], and tackling computational challenges of NeRF and heading towards the real-time rendering [58–60, 222], numerous studies have been conducted to broaden the horizons of NeRF as well as its various applications.

NeRF for 3D Surface Reconstruction

In a NeRF model, the scene geometry is hidden inside the neural networks, i.e., it is implicit. In order to achieve 3D surface reconstruction and transform the NeRF representation into an explicit representation such as a mesh, a surface extraction step is essential. By analyzing and thresholding the learned density, i.e., extracting an arbitrary level set of the density function that is learned by NeRF, and using methods such as marching cubes, the baseline NeRF can extract and reconstruct an approximate explicit 3D geometry [223]. Although NeRF and its variants generate impressive results for the novel view synthesis task, they cannot output high-quality 3D surface reconstruction. The quality of the extracted 3D geometry is not satisfactory because the initial objective of NeRF is novel view synthesis, not 3D surface reconstruction. Since the density-based representation used in NeRFs is flexible and does not have enough constraints on 3D geometry [224], it imposes some limitations on inferring accurate surface geometry, especially when ambiguities exist. Therefore, the extracted surfaces usually contain artifacts. To alleviate this issue, some papers have been presented for the 3D surface reconstruction task that tried to incorporate implicit neural surface representation approaches based on a signed distance function or an occupancy function into NeRF-based methods, benefiting from the advantages of both categories. In these methods, instead of choosing the density-based scene representation used in NeRF, the scene space is usually represented as a signed distance function or an occupancy function.

Oechsle *et al.* proposed UNIfied Neural Implicit SUrface and Radiance Fields (UNISURF) [225] which is a framework for 3D surface reconstruction and capturing high-quality implicit surface geometry from multi-view images without the need for object masks. It unifies the implicit surface models with radiance fields for solid and nontransparent object reconstruction given a set of RGB images. UNISURF represents surfaces and defines object or scene geometries using occupancy values. It learns and optimizes this implicit surface via a volume rendering method like NeRF. The output mesh is extracted using Multiresolution IsoSurface Extraction (MISE) algorithm [45]. Considering reconstruction quality, UNISURF outperforms NeRF [57]. There are some limiting factors for this method, including reconstructing only solid objects and constraints to model transparencies, performance drop for overexposed or rarely visible regions in the ground truth images, and inability to resolve the shape-appearance ambiguities such as shadows and holes in objects.

In another concurrent attempt, P. Wang *et al.* presented NeuS [224], which learns neural implicit surface representation based on SDF using volume rendering, with the goal of reconstructing the 3D surface of an object or scene given multiple images from different viewing points without leveraging mask supervision. Instead of just doing standard volume rendering or standard surface rendering, this framework suggests using volume rendering (inspired by NeRF) in addition to surface representation with neural SDF. The key idea behind this method is to represent a 3D surface as the zero-level set of a signed distance function, i.e., representing a surface with neural implicit SDFs, and to introduce a new volume rendering method by taking inspiration from NeRF, for training a neural SDF representation with robustness. This novel volume rendering technique attempts to learn the weights of the neural network by rendering images from the implicit SDF first, and then minimizing the difference between the rendered images and the input images. NeuS performs quantitatively and qualitatively better than NeRF [57] and UNISURF [225] in high-quality surface reconstruction. However, one failure case of NeuS is its inability to accurately reconstruct textureless regions. This limitation is caused by the ambiguity of these textureless regions for reconstruction in neural rendering.

Variants of NeuS [226, 227] have been proposed with the goal of improving the reconstruction quality. HF-NeuS [226], a method for multi-view surface reconstruction with high-frequency details, breaks down the SDF into fundamental components, namely base and displacement functions, and adopts a gradual increase in high-frequency details through a coarse-to-fine strategy. In Geo-Neus [227], by utilizing sparse 3D points in structure from motion constraint in conjunction with the photometric consistency in multi-view stereo constraint, the learning of neural SDF can be enhanced.

In a similar fashion to NeuS, another concurrent work called VolSDF [228], suggested a volume rendering framework for implicit neural surfaces. Replacing general-purpose MLP densities with densities from a certain family, i.e., in this case representing the density as a function of the signed distance to the scene's surface, is the core contribution of VolSDF. Two fully connected neural networks, one for the approximation of the SDF of the learned geometry, and the other for representing the scene's radiance field, form the structure of this framework. Compared to NeRF [57] and NeRF++ [61], VolSDF generates more accurate results. One of the limitations of VolSDF is that it assumes the object is homogeneous with a constant density. Moreover, its reconstruction time is still high due to the independent training of the network for each scene.

Recently, SDFStudio [229], which is a framework for neural implicit surface reconstruction, has been released. It is built on top of nerfstudio [230] and includes a unified implementation of VolSDF, NeuS, and UNISURF, three popular neural implicit surface reconstruction techniques. Because of the unified and modular implementation of this framework, transferring ideas between methods is simple. The idea from Geo-NeuS can be integrated with VolSDF, bringing about Geo-VolSDF method.

TABLE II: Comparison of various 3D reconstruction methods. Convolutional Neural Network (CNN), Graph Convolution Network (GCN), Intersection over Union (IoU), Chamfer Distance (CD), Earth Mover's Distance (EMD), Hausdorff Distance (HD), Binary Cross-Entropy (BCE), Average Precision (AP), Cross-Entropy (CE), Squared Distance Error (SDE), Normal Consistency (NC), Peak signal-to-noise ratio (PSNR), Structural Similarity Index (SSIM).

Name	Input	Output	Method	Loss	Dataset	Metric
3D-R2N2 [20]	A single-view image or multi-view images	Voxels	Encoder-decoder, 3D conv LSTM	Voxel-wise CE	ShapeNet, Pascal 3D [231], Online products [232]	IoU, CE
VRN [21]	Voxels	Voxels	Encoder-decoder	BCE	ModelNet	Accuracy
TL-embedding [22]	RGB images	Voxels	Encoder-decoder	CE, Euclidean loss	ShapeNet, IKEA dataset [233]	AP
3D-GAN [23]	An image	Voxels	CNN, 3D GAN, Encoder-decoder	BCE loss, KL-divergence loss, reconstruction loss	ModelNet, IKEA dataset, ShapeNet	AP
3D-EPN [24]	Depth maps	Voxels	Encoder-predictor network	L1 loss	ShapeNet	Accuracy, L1 error
3D shape completion [25]	Point cloud and a 3D bounding box	Occupancy grid or SDFs	Encoder-decoder	Reconstruction loss, maximum likelihood loss	ShapeNet, KITTI, ModelNet	Hamming distance, accuracy, completeness
SG-NN [141]	RGB-D scan	A sparse TSDF	Encoder-decoder	L1 loss, BCE loss	Matterport3D	L1 error
Octnetfusion [26]	One/multiple 2.5D depth image(s)	Voxel or Octree	Encoder-decoder	L1 loss, BCE	ModelNet40	IoU, precision, recall
HSP [28]	RGB images or depth images or partial grids	Voxels	Encoder-decoder	CE	ShapeNet	IoU, CD
OGN [29]	Voxels	Structure of an octree and binary occupancy map	Encoder-decoder	CE	ShapeNet	IoU
Adaptive O-CNN [27]	A single image or point cloud	A patch-guided adaptive octree	Encoder-decoder	CE, SDE	ModelNet, ShapeNet	CD, accuracy
Point set generation net [30]	A single RGB or RGB-D image	Point cloud	Encoder-predictor	CD, EMD	ShapeNet	IoU, CD, EMD
Latent 3D points [31]	Point cloud	Point cloud	Encoder-decoder, GAN	CD, EMD	ShapeNet, ModelNet	JSD, coverage, MMD
FoldingNet [32]	Point cloud	Point cloud	Encoder(graph-based)-decoder	CD	ShapeNet, ModelNet	Accuracy
PointFlow [33]	Point cloud	Point cloud	Encoder-decoder	Prior loss, reconstruction loss, posterior loss	ShapeNet	JSD, MMD, coverage, CD, EMD, accuracy
AtlasNet [34]	2D images or point cloud	Mesh	Encoder-decoder	CD loss	ShapeNet	CD
Meshlet [37]	Point cloud	Mesh	Encoder-decoder	CD loss	ShapeNet	CD, HD
Pixel2Mesh [38]	An RGB image	Mesh	Graph convolution network	CD loss, normal loss	Dataset of 3D-R2N2 [20]	F1-score, CD, EMD
Pixel2Mesh++ [39]	A few RGB images or multi-view images	Mesh	GCN	CD loss, normal loss	Dataset of 3D-R2N2 [20]	F1-score, CD
CMR [40]	An image	Mesh	Convolutional encoder	Reprojection loss, regression loss	CUB-200-2011 dataset, PASCAL 3D+ dataset	IoU
Point2Mesh [41]	Point cloud	Mesh	CNN	CD loss, beam-gap loss	A large dataset of object scans [234]	F1-score

DMC [42]	Point cloud	Mesh	Encoder-decoder network with skip connections	Point to mesh loss, occupancy loss, smoothness loss, curvature loss	ShapeNet	CD, accuracy, completeness
Scan2Mesh [43]	One/multiple depth image(s)	Mesh	CNN, graph neural network	CE, CD loss	ShapeNet	CD, normal deviation
Mesh R-CNN [44]	An RGB image	A category label, segmentation mask, boundary box, a 3D triangular mesh	GCN	BCE, CD	ShapeNet, Pix3D dataset	F1-score, CD, NC
Meshing point clouds with IER [35]	Point cloud	Mesh	CNN	Euclidean distance, geodesic distance	ShapeNet	F1-score, CD, NC
REIN [36]	Point cloud	Mesh	Encoder-decoder, RNN	CD, BCE	ShapeNet, ModelNet10	CD, point normal similarity
Occupancy Nets [45]	An image or point cloud or discrete voxel grids	Implicit surface	Encoder, fully connected network	CE	ShapeNet, KITTI, Pix3D	IoU, CD, NC
IM-Net [49]	Images or voxels	Implicit surface	Encoder-decoder, GAN	Weighted mean squared error, Wasserstein GAN loss	ShapeNet	MSE, IoU, CD, LFD, MMD, COV
DeepSDF [50]	Point cloud	Implicit surface	Auto-decoder	L1 loss	ShapeNet	CD, EMD, accuracy
SIREN [51]	Point cloud	Implicit surface	Fully connected neural network	SDF loss (Eikonal equation)	Stanford 3D scanning repository	N/A
SAL [52]	Point cloud or triangle soups	Implicit surface	Variational encoder-decoder	Sign-agnostic loss with L2 distance	D-Faust dataset	CD
NDF [53]	Point cloud	Implicit surface	Encoder-decoder	Unsigned distance field loss	ShapeNet	CD
DUDE [54]	Triangle soups	Implicit surface	Feed-forward networks	L2 loss	ShapeNet	IoU, mean absolute error, normal map error
LIG [55]	Point cloud	Implicit surface	Encoder-decoder	BCE loss	ShapeNet, Matterport 3D, SceneNet	F1-Score, CD
IF-NET [56]	Point cloud or occupancy grid	Implicit surface	Encoder-decoder	CE	ShapeNet	IoU, CD, NC
Conv occupancy nets [46]	Point cloud or voxels/coarse occupancy grid	Implicit surface	Encoder-decoder	BCE	ShapeNet, ScanNet, Matterport 3D	F1-score, IoU, CD, NC
DeepLS [47]	Depth data or mesh	Implicit surface	Autodecoder network	Negative log likelihood loss	Stanford 3D scanning repository, 3D warehouse [235]	CD
Point2surf [48]	Point cloud	Implicit surface	Encoder-decoder	L2 loss, BCE loss	ABC dataset	CD
UNISURF [225]	RGB images	Implicit surface	MLP	Reconstruction loss, Surface regularization	DTU [236], BlendedMVS [237], SceneNet [238]	CD
NeuS [224]	RGB images	Implicit surface	MLP	Color loss, regularization loss, mask loss	DTU [236], BlendedMVS [237]	CD, PSNR, SSIM

TABLE III: Quantitative report about some of the methods' performance on ShapeNet. Chamfer distance (CD), intersection over union (IoU), average precision (AP), and F_score are calculated as the average. For IoU, F_score, and AP, the higher the better. For CD, the lower the better. The number of ShapeNet categories used in an experiment (#Cats), not measured or not mentioned (-), single view reconstruction (SVR), multi-view reconstruction (MVR), reconstruction (R), completion (C), auto-encoding (AE), training time (T), inference time (I), generating a mesh (mg), memory (Mem.), model size (MS). * is calculated for (32^3). + is related to chamfer-L1. For detailed information regarding data preparation methods, train/test splits, metrics, and other specific details, please refer to the context of each individual paper.

Papers	ShapeNet					Task	Time	Size
	CD	IoU	F_score	AP	#Cats			
3D-R2N2 [20]	-	>0.60	-	-	13	MVR (3 views)	-	-
TL-embedding [22]	-	-	-	65.40	5	SVR	-	-
OGN [29]	-	0.59 *	-	-	13	SVR	5 days T for 256^3	0.54G(256^3)
Adaptive O-CNN [27]	0.00460	-	-	-	13	SVR	-	-
Point set generation [30]	0.25000	0.64	-	-	13	SVR	-	-
PointFlow [33]	0.00070	-	-	-	13	AE	-	-
AtlasNet [34]	0.00150	-	-	-	13	AE (25 patches)	-	-
AtlasNet [34]	0.00510	-	-	-	13	SVR (25 patches)	-	-
Meshlet [37]	0.00900	-	-	-	-	R	-	-
Pixel2Mesh [38]	0.59100	-	59.72	-	13	SVR	72h T - 15.5ms I (mg)	-
Pixel2Mesh++ [39]	0.48000	-	66.48	-	13	MVR (3 views)	96h T - 15.5ms I (mg)	-
Scan2Mesh [43]	0.00160	-	-	-	8	C	-	-
Meshing with IER [35]	0.00071	-	87.20	-	8	R	<10s I/a pc with 12,800 pts	-
IM-Net [49]	0.00140	-	-	-	5	SVR	-	-
IM-Net [49]	0.00060	0.75	-	-	5	AE	-	-
DeepSDF [50]	0.00030	-	-	-	5	AE	9.72s I	0.0074(MS)
DeepSDF [50]	0.00160	-	-	-	3	C	9.72s I	0.0074(MS)
IF-NET [56]	0.00002	0.88	-	-	13	R	-	-
Occupancy Nets [45]	0.21500+	0.57	-	-	13	SVR	3s I/per mesh	-
Occupancy Nets [45]	0.07900+	0.77	-	-	13	C	3s I/per mesh	-
Conv onets [46]	0.04800+*	0.87	93.30	-	13	R	-	5.9G Mem.

VII. Discussion and Future Trends

In the previous sections, latest attempts towards 3D reconstruction using deep learning techniques were reviewed. A summary and comparison of presented learning-based surface reconstruction approaches can be found in Table II. Furthermore, Table III contains a quantitative report about the performance of some of the approaches on the ShapeNet dataset. There is a qualitative gap between 3D models created by learning-based approaches, and artist-created CAD models [43] and there are still open problems in this scope. Some of these challenges are listed below.

In the existing approaches, serious bottlenecks are caused by computation time and generalization power. The requirement of long training time is a drawback to the adoption of some of the deep learning-based approaches. On the other hand, there are concerns raised about the environmental impact of prolonged training periods. To this end, designing models with a reduced number of parameters, less complexity, and yet high performance can be a target to hit. Additionally, the utilization of transfer learning may serve as a partial solution. Regarding the generalizability issue, methods with the capability of multi-category generalization, i.e., generalizing well to other topology categories, should be further investigated. One solution might be to learn latent shape spaces which are not class-specific. Consequently, as a future direction, moving toward models with comparable shorter training time and stronger generalizability can be an interesting yet reasonable strategy.

Current methods are highly dependent on an external supervisor for annotating input data. Reducing the need for supervision is a desirable trait for a learning-based approach [40]. Furthermore, there are various large-scale datasets appropriate for geometric deep learning tasks. However, there is still need for creating datasets with richer 3D annotations that are suitable for shape and surface reconstruction.

On the other hand, some of the current evaluation metrics fall short in capturing surface properties accurately. Therefore, it is necessary not to be limited to quantitative results but to explore qualitative results to gain a deeper understanding of surface details as well. Moreover, presenting better and more robust evaluation metrics, which are at the same time computationally efficient and less complex (in point cloud comparison, chamfer distance has quadratic complexity for instance), is another area that is essential to focus on.

In the context of volumetric methods, various challenges exist that should be tackled. Because of the discretization of data, some input information and details may partially be lost. Cubic growth in memory and computational costs with respect to resolution, and poor scalability of these methods with resolution increase lead to difficulty in inferring high-resolution outputs. Considering the influence of 3D resolution on the performance of volumetric CNNs for instance, better performance can be achieved by designing efficient volumetric CNN architectures for instance, that are able to scale to higher resolutions [129].

For point-based approaches, current methods extract a fixed and limited number of points from the point cloud dataset and feed them to their network architecture, thus affecting the output quality. Overcoming this limitation as well as implementing models with the ability to handle variable-length

input can be ambitious yet interesting future directions.

In mesh-based approaches, it is challenging to define a loss on meshes which is easy to optimize [34]. One of the limitations of patch-based approaches in the mesh-based representation category that affects the reconstruction of fine details is the usage of a fixed scale mesh patch [37]. A coarse-to-fine approach and extracting mesh patches at different scales might result in more precise outputs. On the other hand, generating a closed shape using patch-based methods, as well as recognizing and segmenting shapes using these methods are issues that still require solutions [34].

Implicit neural representations have recently gained popularity due to their performance and favorable properties. Existing isosurface extraction approaches used for extracting representations from implicit neural representations are computationally intensive and thus comprise a bottleneck. Furthermore, it may be worthwhile to combine sign-agnostic implicit neural approaches with generative methods such as GANs [52]. Moreover, NeRF-based approaches mostly suffer from high computational cost, long training time, and inability to share knowledge between various scenes thus being scene-specific networks. The necessity for more input images in order to have high-quality outputs should be alleviated. Improving NeRF-based methods' time and computation efficiency, their generalizability to unseen scenes, and their surface reconstruction ability can be important research questions.

In general, reducing the performance gap between synthetic and real-world data, proposing better and more representative evaluation metrics for quantifying shape reconstruction analysis results [49], conducting research in the challenging task of scene-level reconstruction, empowering proposed methods with multi-scale reconstruction (coarse-to-fine manner) [48], implementing and employing methods for capturing high-frequency details with the purpose of reconstructing thin parts of a scene or object in high-quality, considering the equivariance concept for designing a neural network, and fusing different approaches mentioned in Fig. 6 in order to enjoy the benefits of them simultaneously, are aspects that should not be ignored in future studies. Additionally, the application of transformer architectures [210], i.e., a deep learning model that is based on the self-attention mechanism, seems to be promising in 3D vision [239–241]. On the other hand, self-supervised learning [242], which is a technique for predicting unobserved or hidden part of the input from observed or not hidden part of the input, can be one of the interesting approaches for solving reconstruction and in general computer vision problems with low quality and a limited amount of data. Furthermore, considering the current interest, diffusion models [243–246] which learn to infer and generate a meaningful output from pure noise, seem to be another exciting approach to be used in 3D generation, completion, and reconstruction [247, 248].

It is equally expected that surface reconstruction applications play an increasingly important role. One of the major uses will be in observational remote sensing-related disciplines where surface reconstruction will aid in archaeological discoveries, agriculture, disaster prevention and response, and cartography. Equally, design- or projection-based applications have great utilization potential for learned surface reconstruction, including but not limited to, 3D modeling in games and movies, architecture, or CAD. Yet, all of the aforementioned scenarios are considering only (close to) static surfaces. The anticipation is that accurate reconstruction of dynamically changing objects and environments, non-rigid objects or scenes, textureless regions as well as transparent objects, and overcoming the challenges of rarely visible regions, occlusions, shadows, and holes in an object or scene will be crucial and consequential next steps in this field of study. Overall, more applications of neural learning approaches will emerge for surface reconstruction, especially in SFX and VFX animation, human reconstruction, robotics, autonomous driving, and medicine.

VIII. Summary and Conclusion

In this paper, we provided a review of the state-of-the-art approaches for learning-based 3D surface reconstruction. We have taken no special perspective, making the manuscript accessible not only to method researchers, but also to applied users seeking to contextualize these approaches for their domains.

For this, we have reiterated commonly used open and accessible benchmarking datasets, different input and output data modalities, and some acquisition techniques. To make processing results comparable, we have highlighted widely used metrics for evaluating learned models, and detailed their particularities.

The main part of the paper has introduced deep learning-based 3D surface reconstruction approaches. In summary, these can be classified into four major categories based on their output representations: 1) voxel-based, 2) point-based representation, 3) mesh-based, and 4) implicit neural. For each of the categories, we listed some well-known methods, explaining their contributions, challenges, strengths and weaknesses.

Although, 3D deep surface reconstruction has made impressive progress over the last few years, there are several remaining challenges. The following non-exhaustive list will highlight the major open issues:

- Computation time
- Generalizability
- Energy consumption and environmental impact
- Representation compression
- Resolution
- Water tightness
- Non-rigid, dynamic, or transparent object reconstruction
- Reconstruction of rarely visible or occluded regions, shadows, and holes in an object or a scene

Towards the end of the paper, we discussed current challenges and possible future trends in deep 3D surface reconstruction. We assume that coming research will put a strong emphasis on self-attention-based models due to their excelling performance in deep learning in general and 2D computer vision problems, i.e., Vision Transformer and its derivatives, in particular. Moreover, self-supervision will be a strong community focus due to its ability to not only improve reconstructive performance overall, but also to leverage small and potentially

domain-specific datasets. The application of diffusion models seems to be a promising direction as well. Finally, albeit a niche setting, the quantification of reconstruction uncertainties will be of utmost importance for safety-critical applications as well as certain scientific application settings.

ACKNOWLEDGMENTS

We thank our funding agencies.

Anis Farshian received her B.Sc. degree in Computer Engineering from the Technical Faculty of Dr. Shariaty, Iran in 2012 and her M.Sc. degree in Information Technology Engineering in 2017 from the Iran University of Science and Technology, Iran. Presently, she is with the Steinbuch Centre for Computing at the Karlsruhe Institute of Technology. As part of her research as well as her membership within the Helmholtz Analytics Framework project and Helmholtz AI project, she is working on methods for surface reconstruction for applied scientific problems. Her research interests include 3D analysis, machine learning as well as computer-assisted health.

Markus Götz received his B.Sc. and M.Sc. degrees in IT-System Engineering from the University of Potsdam, Germany, in 2010 and 2014 respectively, with intermediate stays the Blekinge Tekniska Högskola, Sweden and CERN, Switzerland. Since 2017 he holds a Ph.D. degree in Computational Engineering from the University of Iceland in conjuction with the Juelich Supercomputing Centre, Germany. Currently, he is a postdoctoral researcher at the Steinbuch Centre for Computing, Karlsruhe Institute of Technology, Germany as the project manager for the Helmholtz Analytics Framework and the head of the HelmholtzAI consultants team. In line with his work, he focuses on applied artificial intelligence and data analysis on high-performance cluster systems to work on the grand challenges in the natural sciences. Markus Götz's research interests include machine learning, global optimization as well as parallel algorithm engineering. He is a member of the IEEE.

Gabriele Cavallaro (Senior Member, IEEE) received his B.Sc. and M.Sc. degrees in Telecommunications Engineering from the University of Trento, Italy, in 2011 and 2013, respectively, and a Ph.D. degree in Electrical and Computer Engineering from the University of Iceland, Iceland, in 2016. From 2016 to 2021 he has been the deputy head of the "High Productivity Data Processing" (HPDP) research group at the Jülich Supercomputing Centre (JSC), Forschungszentrum Jülich, Germany. Since 2022, he is the Head of the "AI and ML for Remote Sensing" Simulation and Data Lab at the JSC and an Adjunct Associate Professor with the School of Natural Sciences and Engineering, University of Iceland, Iceland. From 2020 to 2023, he held the position of Chair for the High-Performance and Disruptive Computing in Remote Sensing (HDCRS) Working Group under the IEEE GRSS Earth Science Informatics Technical Committee (ESI TC). In 2023, he took on the role of Co-chair for the ESI TC. Concurrently, he serves as Visiting Professor at the Φ-lab within the European Space Agency (ESA), where he contributes to the Quantum Computing for Earth Observation (QC4EO) initiative. Additionally, he has been serving as an Associate Editor for the IEEE Transactions on Image Processing (TIP) since October 2022. He was the recipient of the IEEE GRSS Third Prize in the Student Paper Competition of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS) 2015 (Milan - Italy). His research interests cover remote sensing data processing with parallel machine learning algorithms that scale on distributed computing systems and innovative computing technologies.

Charlotte Debus studied Physics at the University of Heidelberg, Germany. After her M.Sc. degree in 2012, she did here PhD in the area of Medical Physics at the German Cancer Research Center (DKFZ) form 2013 – 2016. After a 2-year Post-Doc period at the DKFZ, she was a research associate at the German Aerospace Institute, from 2019 – 2020, where she worked on machine learning methods and high performance computing. Since October 2020, she is a PostDoc at the Steinbuch Center for Computing at the Karslruhe Institue of Technology, in the Helmholtz AI local energy consultants team.

Matthias Niessner is the head of the Visual Computing Lab at Technical University of Munich (TUM). He obtained his PhD from the University of Erlangen-Nuremberg in 2013, and was a Visiting Assistant Professor at Stanford University from 2013 to 2017. Since 2017 he is Professor at TUM focusing on cutting-edge research at the intersection of computer vision, graphics, and machine learning. He is particularly interested in novel techniques for 3D reconstruction, semantic 3D scene understanding, and video editing. In addition to his academic career, Prof. Nießner is a co-founder and director of Synthesia Inc., a startup empowering storytellers with AI. Prof. Nießner is a TUM-IAS Rudolph Moessbauer Fellow, and he has received the Google Faculty Award for Machine Perception (2017), the Nvidia Professor Partnership Award (2018), as well as the ERC Starting Grant 2018.

Jón Atli Benediktsson (Fellow, IEEE) received the Cand.Sci. degree in Electrical Engineering from the University of Iceland, Reykjavik, Iceland, in 1984, and the M.S.E.E. and Ph.D. degrees in Electrical Engineering from Purdue University, West Lafayette, IN, USA, in 1987 and 1990, respectively. Since July 1, 2015, he has been the President and Rector of the University of Iceland. From 2009 to 2015, he was the Pro-Rector of Science and Academic Affairs and a Professor of Electrical and Computer Engineering with the University of Iceland. His research interests are in remote sensing, biomedical analysis of signals, pattern recognition, image processing, and signal processing, and he has published extensively in those fields. Dr. Benediktsson is a fellow of SPIE. He was a member of the 2014 IEEE Fellow Committee. He is a member of Academia Europea, the Association of Chartered Engineers in Iceland (VFI), Societas Scinetiarum Islandica, and Tau Beta Pi. He received the Stevan J. Kristof Award from Purdue University in 1991 as an Outstanding Graduate Student in Remote Sensing. He was the recipient of the Icelandic Research Council's Outstanding Young Researcher Award in 1997, in 2000 he was granted the IEEE Third Millennium Medal, in 2004, he was a co-recipient of the University of Iceland's Technology Innovation Award, in 2006, he received the yearly research award from the Engineering Research Institute of the University of Iceland, in 2007, he received the Outstanding Service Award from the IEEE GRSS, in 2020, the IEEE GRSS Education Award, in 2018, the IEEE GRSS David Landgrebe Award, and in 2016, the OECE Award from the School of ECE, Purdue University. He was co-recipient of the 2012 IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING Paper Award, and in 2013, he was co-recipient of the IEEE GRSS Highest Impact Paper Award. In 2013, he received the IEEE/VFI Electrical Engineer of the Year Award. In 2016 and 2018, he was a co-recipient of the International Journal of Image and Data Fusion Best Paper Award. In 2021, he was honored as a recipient of the Order of the Falcon from the President of Iceland. He is a Highly Cited Researcher (Clarivate Analysis, 2018–2020). He was the 2011–2012 President of the IEEE Geoscience and Remote Sensing Society (GRSS) and has been on the GRSS AdCom since 2000. He was an Editor in Chief of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING (TGRS) from 2003 to 2008 and has served as an Associate Editor of TGRS since 1999, the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS since 2003 and IEEE ACCESS since 2013. He was on the Editorial Board of the PROCEEDINGS OF THE IEEE from 2015 to 2020. He is on the International Editorial Board of the International Journal of Image and Data Fusion, the Editorial Board of Remote Sensing, and was the Chairman of the Steering Committee of the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING (J-STARS) from 2007 to 2010. He is a co-founder of the Biomedical Startup Company Oxymap (www.oxymap.com), Reykjavik.

Achim Streit is one of the directors of the Steinbuch Centre for Computing at the Karlsruhe Institute of Technology (KIT). He is also a Professor for Distributed and Parallel High-performance Computing Systems at KIT's department of Informatics. In 1999, he received a Diploma in Computer Science from the University of Dortmund, Germany and in 2003 a Ph.D. degree in the same subject from the University of Paderborn, Germany. Afterwards Achim Streit led the Federated Systems and Data Division at the Juelich Supercomputing Centre, Germany. He initiated and chaired several national and international research initiatives within the Helmholtz association (e.g. Helmholtz Data Federation and Helmholtz Information & Data Science Academy (HIDA)) on the national level (e.g. NFDI4Ing and NFDI-MatWerk) and the European level (e.g. EUDAT and EOSC). His research interests include high-performance and data-intensive computing, Big Data and federated data management, data analytics as well as job scheduling and resource management for parallel and distributed systems.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. DOI: 10.1038/nature14539.

[2] I. Goodfellow, Y. Bengio, and A. Courville. MIT press, 2016, ISBN: 9780262035613.

[3] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Networks : the Official Journal of the International Neural Network Society*, vol. 61, pp. 85–117, 2015. DOI: 10.1016/j.neunet.2014.09.003.

[4] L. Chen, S. Peng, and X. Zhou, "Towards efficient and photorealistic 3d human reconstruction: A brief survey," *Visual Informatics*, vol. 5, no. 4, pp. 11–19, 2021. DOI: 10.1016/j.visinf.2021.10.003.

[5] Y. Tian, H. Zhang, Y. Liu, and L. Wang, "Recovering 3d human mesh from monocular images: A survey," 2022.

[6] M. Götz, C. Bodenstein, and M. Riedel, "HPDBSCAN: Highly Parallel DBSCAN," in *Proceedings of the workshop on machine learning in high-performance computing environments*, 2015, pp. 1–10. DOI: 10.1145/2834892.2834894.

[7] M. Götz, G. Cavallaro, T. Géraud, M. Book, and M. Riedel, "Parallel Computation of Component Trees on Distributed Memory Machines," *IEEE transactions on parallel and distributed systems*, vol. 29, no. 11, pp. 2582–2598, 2018. DOI: 10.1109/TPDS.2018.2829724.

[8] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson Surface Reconstruction," in *Proceedings of the Fourth Eurographics Symposium on Geometry Processing (SGP'06)*, vol. 7, ACM, 2006, pp. 61–70. DOI: 10.2312/SGP/SGP06/061-070.

[9] M. Kazhdan and H. Hoppe, "Screened Poisson Surface Reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, pp. 1–13, 2013. DOI: 10.1145/2487228.2487237.

[10] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The Ball-pivoting Algorithm for Surface Reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, 1999. DOI: 10.1109/2945.817351.

[11] J.-D. Boissonnat and B. Geiger, "Three-dimensional Reconstruction of Complex Shapes Based on the Delaunay Triangulation," in *Proceedings of the Biomedical Image Processing and Biomedical Visualization*, International Society for Optics and Photonics, vol. 1905, 1993, pp. 964–975. DOI: 10.1117/12.148710.

[12] S. Fortune, "Voronoi Diagrams and Delaunay Triangulations," *Computing in Euclidean Geometry*, pp. 225–265, 1995. DOI: 10.1142/9789812831699_0007.

[13] E. Che, J. Jung, and M. J. Olsen, "Object Recognition, Segmentation, and Classification of Mobile Laser Scanning Point Clouds: A State of the Art Review," *Sensors*, vol. 19, no. 4, p. 810, 2019. DOI: 10.3390/s19040810.

[14] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep Learning for 3D Point Clouds: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. DOI: 10.1109/TPAMI.2020.3005434.

[15] L. Jiao *et al.*, "A Survey of Deep Learning-based Object Detection," *IEEE Access*, vol. 7, pp. 128 837–128 868, 2019. DOI: 10.1109/ACCESS.2019.2939201.

[16] Y. Xie, J. Tian, and X. X. Zhu, "Linking Points with Labels in 3D: A Review of Point Cloud Semantic Segmentation," *IEEE Geoscience and Remote Sensing Magazine*, vol. 8, no. 4, pp. 38–59, 2020. DOI: 10.1109/MGRS.2019.2937630.

[17] M. Berger *et al.*, "A Survey of Surface Reconstruction from Point Clouds," in *Computer Graphics Forum*, Wiley Online Library, vol. 36, 2017, pp. 301–329. DOI: 10.1111/cgf.12802.

[18] M. Zollhöfer *et al.*, "State of the Art on 3D Reconstruction with RGB-D Cameras," in *Computer graphics forum*, Wiley Online Library, vol. 37, 2018, pp. 625–652. DOI: 10.1111/cgf.13386.

[19] X.-F. Han, H. Laga, and M. Bennamoun, "Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 5, pp. 1578–1604, 2019. DOI: 10.1109/TPAMI.2019.2954885.

[20] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A Unified Approach for Single and Multi-View 3D Object Reconstruction," in *European Conference on Computer Vision*, Springer, 2016, pp. 628–644. DOI: 10.1007/978-3-319-46484-8_38.

[21] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, *Generative and Discriminative Voxel Modeling with Convolutional Neural Networks*, 2016. arXiv: 1608.04236 [cs.CV].

[22] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a Predictable and Generative Vector Representation for Objects," in *European Conference on Computer Vision*, Springer, 2016, pp. 484–499. DOI: 10.1007/978-3-319-46466-4_29.

[23] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling," 2016. arXiv: 1610.07584 [cs.CV].

[24] A. Dai, C. Ruizhongtai Qi, and M. Nießner, "Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5868–5877. DOI: 10.1109/CVPR.2017.693.

[25] D. Stutz and A. Geiger, "Learning 3D Shape Completion from Laser Scan Data with Weak Supervision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1955–1964. DOI: 10.1109/CVPR.2018.00209.

[26] G. Riegler, A. O. Ulusoy, H. Bischof, and A. Geiger, "OctNetFusion: Learning Depth Fusion from Data," in *2017 International Conference on 3D Vision (3DV)*, IEEE, 2017, pp. 57–66. DOI: 10.1109/3DV.2017.00017.

[27] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong, "Adaptive O-CNN: a Patch-based Deep Representation of 3D Shapes," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–11, 2018. DOI: 10.1145/3272127.3275050.

[28] C. Häne, S. Tulsiani, and J. Malik, "Hierarchical Surface Prediction for 3D Object Reconstruction," in *Proceedings of the 2017 International Conference on 3D Vision (3DV)*, IEEE, 2017, pp. 412–420. DOI: 10.1109/3DV.2017.00054.

[29] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree Generating Networks: Efficient Convolutional Architectures for High-Resolution 3D Outputs," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2017. DOI: 10.1109/ICCV.2017.230.

[30] H. Fan, H. Su, and L. J. Guibas, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 605–613. DOI: 10.1109/CVPR.2017.264.

[31] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning Representations and Generative Models for 3D Point Clouds," in *Proceedings of the 35th International Conference on Machine Learning (ICML*, PMLR, vol. 80, 2018, pp. 40–49. [Online]. Available: http://proceedings.mlr.press/v80/achlioptas18a.html.

[32] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation," in *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2018, pp. 206–215. DOI: 10.1109/CVPR.2018.00029.

[33] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "PointFlow: 3D Point Cloud Generation with Continuous Normalizing Flows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4541–4550. DOI: 10.1109/ICCV.2019.00464.

[34] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché Approach to Learning 3D Surface Generation," in *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2018, pp. 216–224. DOI: 10.1109/CVPR.2018.00030.

[35] M. Liu, X. Zhang, and H. Su, "Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance," in *2020 European Conference on Computer Vision (ECCV)*, Springer, 2020, pp. 68–84. DOI: 10.1007/978-3-030-58598-3_5.

[36] R. Daroya, R. Atienza, and R. Cajote, "REIN: Flexible Mesh Generation from Point Clouds," in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, IEEE, 2020,

pp. 352–353. DOI: 10.1109/CVPRW50498.2020.00184.

[37] A. Badki, O. Gallo, J. Kautz, and P. Sen, "Meshlet Priors for 3D Mesh Reconstruction," in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2849–2858. DOI: 10.1109/CVPR42600.2020.00292.

[38] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Springer, 2018, pp. 52–67. DOI: 10.1007/978-3-030-01252-6_4.

[39] C. Wen, Y. Zhang, Z. Li, and Y. Fu, "Pixel2Mesh++: Multi-View 3D Mesh Generation via Deformation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1042–1051. DOI: 10.1109/ICCV.2019.00113.

[40] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik, "Learning Category-Specific Mesh Reconstruction from Image Collections," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 371–386. DOI: 10.1007/978-3-030-01267-0_23.

[41] R. Hanocka, G. Metzer, R. Giryes, and D. Cohen-Or, "Point2mesh: A self-prior for deformable meshes," *ACM Transactions on Graphics*, vol. 39, no. 4, pp. 1–12, 2020. DOI: 10.1145/3386569.3392415.

[42] Y. Liao, S. Donne, and A. Geiger, "Deep Marching Cubes: Learning Explicit Surface Representations," in *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2018, pp. 2916–2925. DOI: 10.1109/CVPR.2018.00308.

[43] A. Dai and M. Nießner, "Scan2Mesh: From Unstructured Range Scans to 3D Meshes," in *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2019, pp. 5574–5583. DOI: 10.1109/CVPR.2019.00572.

[44] G. Gkioxari, J. Malik, and J. Johnson, "Mesh R-CNN," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9785–9795. DOI: 10.1109/ICCV.2019.00988.

[45] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy Networks: Learning 3D Reconstruction in Function Space," in *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2019, pp. 4460–4470. DOI: 10.1109/CVPR.2019.00459.

[46] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional Occupancy Networks," in *2020 European Conference on Computer Vision (ECCV)*, Springer International Publishing, 2020. DOI: 10.1007/978-3-030-58580-8_31.

[47] R. Chabra *et al.*, "Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction," in *European Conference on Computer Vision*, Springer, 2020, pp. 608–625. DOI: 10.1007/978-3-030-58526-6_36).

[48] P. Erler, P. Guerrero, S. Ohrhallinger, N. J. Mitra, and M. Wimmer, "Points2Surf Learning Implicit Surfaces from Point Clouds," *Lecture Notes in Computer Science*, pp. 108–124, 2020, ISSN: 1611-3349. DOI: 10.1007/978-3-030-58558-7_7.

[49] Z. Chen and H. Zhang, "Learning Implicit Fields for Generative Shape Modeling," in *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2019, pp. 5939–5948. DOI: 10.1109/CVPR.2019.00609.

[50] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation," in *Proceedings of the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2019, pp. 165–174. DOI: 10.1109/CVPR.2019.00025.

[51] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit Neural Representations with Periodic Activation Functions," *Advances in Neural Information Processing Systems*, vol. 33, 2020. [Online]. Available: https://proceedings.neurips.cc//paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf.

[52] M. Atzmon and Y. Lipman, "SAL: Sign Agnostic Learning of Shapes from Raw Data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2565–2574. DOI: 10.1109/CVPR42600.2020.00264.

[53] J. Chibane, A. Mir, and G. Pons-Moll, "Neural Unsigned Distance Fields for Implicit Function Learning," 2020. arXiv: 2010.13938 [cs.CV].

[54] R. Venkatesh, S. Sharma, A. Ghosh, L. Jeni, and M. Singh, "DUDE: Deep Unsigned Distance Embeddings for Hi-Fidelity Representation of Complex 3D Surfaces," 2020. arXiv: 2011.02570 [cs.CV].

[55] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, T. Funkhouser, *et al.*, "Local Implicit Grid Representations for 3D Scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6001–6010. DOI: 10.1109/cvpr42600.2020.00604.

[56] J. Chibane, T. Alldieck, and G. Pons-Moll, "Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion," in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2020, pp. 6970–6981. DOI: 10.1109/CVPR42600.2020.00700.

[57] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *European Conference on Computer Vision*, Springer, 2020, pp. 405–421. DOI: 10.1007/978-3-030-58452-8_24.

[58] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. Valentin, "FastNeRF: High-Fidelity Neural Rendering at 200FPS," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 346–14 355. DOI: 10.1109/ICCV48922.2021.01408.

[59] T. Neff *et al.*, "DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks," in *Computer Graphics Forum*, Wiley Online Library, vol. 40, 2021, pp. 45–59. DOI: 10.1111/cgf.14340.

[60] C. Reiser, S. Peng, Y. Liao, and A. Geiger, "KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 335–14 345. DOI: 10.1109/ICCV48922.2021.01407.

[61] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "NeRF++: Analyzing and Improving Neural Radiance Fields," *arXiv preprint arXiv:2010.07492*, 2020.

[62] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelNeRF: Neural Radiance Fields from One or Few Images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4578–4587. DOI: 10.1109/CVPR46437.2021.00455.

[63] A. Chen *et al.*, "MVSNeRF: Fast Generalizable Radiance Field Reconstruction from Multi-View Stereo," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 124–14 133. DOI: 10.1109/ICCV48922.2021.01386.

[64] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron, "NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7495–7504. DOI: 10.1109/CVPR46437.2021.00741.

[65] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. Lensch, "NeRD: Neural Reflectance Decomposition from Image Collections," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 684–12 694. DOI: 10.1109/ICCV48922.2021.01245.

[66] G. Gafni, J. Thies, M. Zollhofer, and M. Nießner, "Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8649–8658. DOI: 10.1109/CVPR46437.2021.00854.

[67] K. Park *et al.*, "Nerfies: Deformable Neural Radiance Fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5865–5874. DOI: 10.1109/ICCV48922.2021.00581.

[68] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt, "Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene from Monocular Video," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 959–12 970. DOI: 10.1109/ICCV48922.2021.01272.

[69] H. Kato, Y. Ushiku, and T. Harada, "Neural 3D Mesh Renderer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3907–3916. DOI: 10.1109/CVPR.2018.00411.

[70] J. Pan, X. Han, W. Chen, J. Tang, and K. Jia, "Deep Mesh Reconstruction from Single RGB Images via Topology Modification Networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9964–9973. DOI: 10.1109/ICCV.2019.01006.

[71] J. Tang, X. Han, J. Pan, K. Jia, and X. Tong, "A Skeleton-Bridged Deep Learning Approach for Generating Meshes of Complex Topologies from Single RGB Images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4541–4550. DOI: 10.1109/CVPR.2019.00467.

[72] Q. Wang, Y. Tan, and Z. Mei, "Computational Methods of Acquisition and Processing of 3D Point Cloud Data for Construction Applications," *Archives of Computational Methods in Engineering*, vol. 27, pp. 479–499, Apr. 2020. DOI: 10.1007/s11831-019-09320-4.

[73] Y. Xie, J. Tian, and X. X. Zhu, "Linking Points With Labels in 3D: A Review of Point Cloud Semantic Segmentation," *IEEE Geoscience and Remote Sensing Magazine*, vol. 8, no. 4, pp. 38–59, 2020. DOI: 10.1109/MGRS.2019.2937630.

[74] S. J. and T. (C. K., *Topographic Laser Ranging and Scanning: Principles and Processing*. CRC Press, 2017. DOI: 10.1201/9781315154381.

[75] Y. Gu, Q. Wang, X. Jia, and J. A. Benediktsson, "A Novel MKL Model of Integrating LiDAR Data and MSI for Urban Area Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 10, pp. 5312–5326, 2015. DOI: 10.1109/TGRS.2015.2421051.

[76] J. C. Fernandez-Diaz *et al.*, "Capability Assessment and Performance Metrics for the Titan Multispectral Mapping Lidar," *Remote Sensing*, vol. 8, no. 11, 2016. DOI: 10.3390/rs8110936. [Online]. Available: https://www.mdpi.com/2072-4292/8/11/936.

[77] M. Pedergnana, P. R. Marpu, M. Dalla Mura, J. A. Benediktsson, and L. Bruzzone, "Classification of Remote Sensing Optical and LiDAR Data Using Extended Attribute Profiles," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 7, pp. 856–865, 2012. DOI: 10.1109/JSTSP.2012.2208177.

[78] M. Kukkonen, M. Maltamo, L. Korhonen, and P. Packalen, "Comparison of Multispectral Airborne Laser Scanning and Stereo Matching of Aerial Images as a Single Sensor Solution to Forest Inventories by Tree Species," *Remote Sensing of Environment*, vol. 231, p. 111 208, 2019, ISSN: 0034-4257. DOI: 10.1016/j.rse.2019.05.027. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425719302214.

[79] M. A. Isa and I. Lazoglu, "Design and Analysis of a 3D Laser Scanner," *Measurement*, vol. 111, pp. 122–133, 2017, ISSN: 0263-2241. DOI: 10.1016/j.measurement.2017.07.028. [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S0263224117304633.

[80] F. Rottensteiner *et al.*, "The ISPRS Benchmark on Urban Object Classification and 3D Building Reconstruction," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences I-3 (2012), Nr. 1*, vol. 1, no. 1, pp. 293–298, 2012. DOI: 10.5194/isprsannals-i-3-293-2012.

[81] Y. Wang, Q. Chen, Q. Zhu, L. Liu, C. Li, and D. Zheng, "A survey of mobile laser scanning applications and key techniques over urban areas," *Remote Sensing*, vol. 11, no. 13, 2019. DOI: 10.3390/rs11131540. [Online]. Available: https://www.mdpi.com/2072-4292/11/13/1540.

[82] M. Elhousni and X. Huang, "A Survey on 3D LiDAR Localization for Autonomous Vehicles," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1879–1884. DOI: 10.1109/IV47402.2020.9304812.

[83] J. Li, B. Yang, Y. Cong, L. Cao, X. Fu, and Z. Dong, "3D Forest Mapping Using A Low-Cost UAV Laser Scanning System: Investigation and Comparison," *Remote Sensing*, vol. 11, no. 6, 2019. DOI: 10.3390/rs11060717.

[84] T. Zwęgliński, "The use of drones in disaster aerial needs reconnaissance and damage assessment – three-dimensional modeling and orthophoto map study," *Sustainability*, vol. 12, no. 15, 2020. DOI: 10.3390/su12156080. [Online]. Available: https://www.mdpi.com/2071-1050/12/15/6080.

[85] F. Leberl *et al.*, "Point Clouds: Lidar versus 3D Vision," *Photogrammetric Engineering & Remote Sensing*, vol. 76, no. 10, pp. 1123–1134, 2010. DOI: 10.14358/PERS.76.10.1123.

[86] Q. Hu, J. Luo, G. Hu, W. Duan, and H. Zhou, "3D Point Cloud Generation Using Incremental Structure-from-Motion," *Journal of Physics: Conference Series*, vol. 1087, p. 062 031, Sep. 2018. DOI: 10.1088/1742-6596/1087/6/062031.

[87] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, 2006, pp. 519–528. DOI: 10.1109/CVPR.2006.19.

[88] K. Kamal *et al.*, "Performance Assessment of Kinect as a Sensor for Pothole Imaging and Metrology," *International Journal of Pavement Engineering*, vol. 19, no. 7, pp. 565–576, 2018. DOI: 10.1080/10298436.2016.1187730.

[89] C. Jia, T. Yang, C. Wang, B. Fan, and F. He, "A New Fast Filtering Algorithm for a 3D Point Cloud based on RGB-D Information," *PLOS ONE*, vol. 14, e0220253, Aug. 2019. DOI: 10.1371/journal.pone.0220253.

[90] C. Chen, B. Yang, and S. Song, "Low Cost and Efficient 3D Indoor Mapping Using Multiple Consumer RGB-D Cameras," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 41B1, pp. 169–174, 2016. DOI: 10.5194/isprs-archives-XLI-B1-169-2016.

[91] H. Sarbolandi, D. Lefloch, and A. Kolb, "Kinect Range Sensing: Structured-Light versus Time-of-Flight Kinect," *Computer Vision and Image Understanding*, vol. 139, pp. 1–20, 2015. DOI: 10.1016/j.cviu.2015.05.006.

[92] R. Buergmann, P. A. Rosen, and E. J. Fielding, "Synthetic Aperture Radar Interferometry to Measure Earth's Surface Topography and Its Deformation," *Annual Review of Earth and Planetary Sciences*, vol. 28, no. 1, pp. 169–209, 2000. DOI: 10.1146/annurev.earth.28.1.169.

[93] A. Ferretti, C. Prati, and F. Rocca, "Permanent Scatterers in SAR Interferometry," in *IEEE 1999 International Geoscience and Remote Sensing Symposium. IGARSS'99 (Cat. No.99CH36293)*, vol. 3, 1999, pp. 1528–1530. DOI: 10.1109/IGARSS.1999.772008.

[94] "Editorial for the special issue "urban deformation monitoring using persistent scatterer interferometry and sar tomography"," *Remote Sensing*, vol. 11, no. 11, 2019. DOI: 10.3390/rs11111306. [Online]. Available: https://www.mdpi.com/2072-4292/11/11/1306.

[95] A. Gruen, "Fundamentals of Videogrammetry — A Review," *Human Movement Science*, vol. 16, no. 2, pp. 155–187, 1997, 3-D Analysis of Human Movement - II, ISSN: 0167-9457. DOI: 10.1016/S0167-9457(96)00048-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167945796000486.

[96] A. Torresani and F. Remondino, "Videogrammetry vs Photogrammetry for Heritage 3D Reconstruction," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2/W15, pp. 1157–1162, 2019. DOI: 10.5194/isprs-archives-XLII-2-W15-1157-2019.

[97] A. X. Chang *et al.*, *ShapeNet: An Information-Rich 3D Model Repository*, 2015. arXiv: 1512.03012 [cs.GR]. [Online]. Available: https://shapenet.org/.

[98] K. Mo *et al.*, "Partnet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding," in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2019, pp. 909–918. DOI: 10.1109/CVPR.2019.00100. [Online]. Available: https://partnet.cs.stanford.edu/.

[99] Z. Wu *et al.*, "3D ShapeNets: A Deep Representation for Volumetric Shapes," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2015, pp. 1912–1920. DOI: 10.1109/CVPR.2015.7298801. [Online]. Available: http://modelnet.cs.princeton.edu/.

[100] A. Geiger, P. Lenz, and R. Urtasun, "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2012, pp. 3354–3361. DOI: 10.1109/CVPR.

2012.6248074. [Online]. Available: http://www.cvlibs.net/datasets/kitti/index.php.

[101] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision Meets Robotics: The KITTI Dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013. DOI: 10.1177/0278364913491297. [Online]. Available: http://www.cvlibs.net/datasets/kitti/index.php.

[102] J. Behley *et al.*, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, 2019, pp. 9297–9307. DOI: 10.1109/ICCV.2019.00939. [Online]. Available: http://www.semantic-kitti.org/index.html.

[103] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Niessner, "ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017. DOI: 10.1109/CVPR.2017.261. [Online]. Available: http://www.scan-net.org/.

[104] A. Chang *et al.*, "Matterport3D: Learning from RGB-D Data in Indoor Environments," in *2017 International Conference on 3D Vision (3DV)*, IEEE, 2017, pp. 667–676. DOI: 10.1109/3DV.2017.00081. [Online]. Available: https://niessner.github.io/Matterport/.

[105] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *European Conference on Computer Vision*, Springer, 2012, pp. 746–760. DOI: 10.1007/978-3-642-33715-4_54. [Online]. Available: https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html.

[106] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A Database of Big Spaces Reconstructed using SFM and Object Labels," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1625–1632. DOI: 10.1109/ICCV.2013.458. [Online]. Available: http://sun3d.cs.princeton.edu/.

[107] S. Song, S. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2015, pp. 567–576. DOI: 10.1109/CVPR.2015.7298655. [Online]. Available: http://rgbd.cs.princeton.edu/.

[108] A. Janoch *et al.*, "A Category-Level 3D Object Dataset: Putting the Kinect to Work," in *Consumer Depth Cameras for Computer Vision*, Springer, 2013, pp. 141–165. DOI: 10.1109/ICCVW.2011.6130382.

[109] M. De Deuge, A. Quadros, C. Hung, and B. Douillard, "Unsupervised Feature Learning for Classification of Outdoor 3D Scans," in *Proceedings of the Australasian Conference on Robotics and Automation*, vol. 2, 2013, p. 1. [Online]. Available: http://www.acfr.usyd.edu.au/papers/SydneyUrbanObjectsDataset.shtml.

[110] S. Koch *et al.*, "ABC: A Big CAD Model Dataset For Geometric Deep Learning," in *The 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. DOI: 10.1109/CVPR.2019.00983. [Online]. Available: https://cs.nyu.edu/~zhongshi/publication/abc-dataset/.

[111] T. Hackel, N. Savinov, L. Ladicky, J. Wegner, K. Schindler, and M. Pollefeys, "Semantic3D.net: A New Large-scale Point Cloud Classification Benchmark," in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-1-W1, 2017, pp. 91–98. DOI: 10.5194/ISPRS-ANNALS-IV-1-W1-91-2017. [Online]. Available: https://www.semantic3d.net/view_dbase.php?chl=1&orderBy=name&orderStyle=ASC#download.

[112] M. Kölle *et al.*, *H3D: Benchmark on Semantic Segmentation of High-Resolution 3D Point Clouds and Textured Meshes from UAV LiDAR and Multi-View-Stereo*, 2021. arXiv: 2102.05346 [cs.CV]. [Online]. Available: https://ifpwww.ifp.uni-stuttgart.de/benchmark/hessigheim/default.aspx.

[113] H. Fu *et al.*, "3D-FRONT: 3D Furnished Rooms with layOuts and semaNTics," 2020. arXiv: 2011.09127 [cs.CV]. [Online]. Available: https://tianchi.aliyun.com/specials/promotion/alibaba-3d-scene-dataset.

[114] H. Fu *et al.*, "3D-FUTURE: 3D Furniture Shape with TextURE," 2020. arXiv: 2009.09633 [cs.CV]. [Online]. Available: https://tianchi.aliyun.com/specials/promotion/alibaba-3d-future.

[115] Q. Hu, B. Yang, S. Khalid, W. Xiao, N. Trigoni, and A. Markham, "Towards Semantic Segmentation of Urban-Scale 3D Point Clouds: A Dataset, Benchmarks and Challenges," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4977–4987. [Online]. Available: http://point-cloud-analysis.cs.ox.ac.uk/.

[116] The Stanford Computer Graphics Laboratory, 2014. [Online]. Available: https://graphics.stanford.edu/data/3Dscanrep/.

[117] G. Turk and B. Mullins, 2021. [Online]. Available: https://www.cc.gatech.edu/projects/large_models/.

[118] Y. Rubner, C. Tomasi, and L. J. Guibas, "A Metric for Distributions with Applications to Image Databases," in *Proceedings of the Sixth International Conference on Computer Vision (ICCV)*, IEEE, 1998, pp. 59–66. DOI: 10.1109/ICCV.1998.710701.

[119] Y. Rubner, C. Tomasi, and L. J. Guibas, "The Earth Mover's Distance as a Metric for Image Retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000. DOI: 10.1023/A:1026543900054.

[120] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox, "What Do Single-View 3D Reconstruction Networks Learn?" In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3405–3414. DOI: 10.1109/CVPR.2019.00352.

[121] O. Taubert, M. Götz, A. Schug, and A. Streit, "Loss Scheduling for Class-Imbalanced Image Segmentation Problems," in *2020 19th IEEE International Conference on Machine Learning and Applications*

(ICMLA), IEEE, 2020, pp. 426–431. DOI: 10.1109/ICMLA51294.2020.00073.

[122] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951. DOI: 10.1214/AOMS/1177729694.

[123] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, "On Visual Similarity Based 3D Model Retrieval," in *Computer Graphics Forum*, vol. 22, Wiley, 2003, pp. 223–232. DOI: 10.1111/1467-8659.00669.

[124] H. Kato *et al.*, *Differentiable Rendering: A Survey*, 2020. arXiv: 2006.12057 [cs.CV].

[125] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, ACM, 1996, pp. 303–312, ISBN: 0897917464. DOI: 10.1145/237170.237269.

[126] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*, ACM, 1987, pp. 163–169. DOI: 10.1145/37401.37422.

[127] D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for Real-time Object Recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 922–928. DOI: 10.1109/IROS.2015.7353481.

[128] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox, *Orientation-boosted Voxel Nets for 3D Object Recognition*, 2017. arXiv: 1604.03351 [cs.CV].

[129] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and Multi-View CNNs for Object Classification on 3D Data," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2016, pp. 5648–5656. DOI: 10.1109/CVPR.2016.609.

[130] V. Hegde and R. Zadeh, "FusionNet: 3D Object Classification using Multiple Data Representations," 2016. arXiv: 1607.05695 [cs.CV].

[131] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors," in *Symposium on Geometry Processing*, vol. 6, 2003, pp. 156–164. DOI: 10.2312/SGP/SGP03/156-165.

[132] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-View Convolutional Neural Networks for 3D Shape Recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 945–953. DOI: 10.1109/ICCV.2015.114.

[133] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[134] A. Kar, S. Tulsiani, J. Carreira, and J. Malik, "Category-specific Object Reconstruction from a Single Image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1966–1974. DOI: 10.1109/CVPR.2015.7298807.

[135] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, IEEE, vol. 1, 2006, pp. 519–528. DOI: 10.1109/CVPR.2006.19.

[136] D. P. Kingma and M. Welling, *Auto-Encoding Variational Bayes*, 2014. arXiv: 1312.6114 [stat.ML].

[137] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," *Foundations and Trends in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019, ISSN: 1935-8245. DOI: 10.1561/2200000056.

[138] I. J. Goodfellow *et al.*, *Generative Adversarial Networks*, 2014. arXiv: 1406.2661 [stat.ML].

[139] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," 2015. arXiv: 1511.06434 [cs.LG].

[140] J. Wu, C. Zhang, X. Zhang, Z. Zhang, W. T. Freeman, and J. B. Tenenbaum, "Learning Shape Priors for Single-View 3D Completion and Reconstruction," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 646–662. DOI: 10.1007/978-3-030-01252-6_40.

[141] A. Dai, C. Diller, and M. Nießner, "SG-NN: Sparse Generative Neural Networks for Self-Supervised Scene Completion of RGB-D Scans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 849–858. DOI: 10.1109/cvpr42600.2020.00093.

[142] C. Choy, J. Gwak, and S. Savarese, "4D Spatio-Temporal Convnets: Minkowski Convolutional Neural Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084. DOI: 10.1109/CVPR.2019.00319.

[143] S. R. Richter and S. Roth, "Matryoshka Networks: Predicting 3D Geometry via Nested Shape Layers," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1936–1944. DOI: 10.1109/CVPR.2018.00207.

[144] D. Meagher, "Geometric Modeling using Octree Encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, 1982. DOI: 10.1016/0146-664X(82)90104-6.

[145] C. L. Jackins and S. L. Tanimoto, "Oct-trees and Their Use in Representing Three-dimensional Objects," *Computer Graphics and Image Processing*, vol. 14, no. 3, pp. 249–270, 1980. DOI: 10.1016/0146-664X(80)90055-6.

[146] G. Riegler, A. Osman Ulusoy, and A. Geiger, "OctNet: Learning Deep 3D Representations at High Resolutions," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 3577–3586. DOI: 10.1109/CVPR.2017.701.

[147] C. Zach, T. Pock, and H. Bischof, "A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration," in *2007 IEEE 11th International Conference on Computer Vision*, IEEE, 2007, pp. 1–8. DOI: 10.1109/ICCV.2007.4408983.

[148] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow, "Structured Prediction of Unobserved Voxels from a Single Depth Image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5431–5440. DOI: 10.1109/CVPR.2016.586.

[149] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017. DOI: 10.1145/3072959.3073608.

[150] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 652–660. DOI: 10.1109/CVPR.2017.16.

[151] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates Inc., 2017, pp. 5099–5108. [Online]. Available: https://papers.nips.cc/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf.

[152] R. Klokov and V. Lempitsky, "Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 863–872. DOI: 10.1109/ICCV.2017.99.

[153] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining Point Cloud Local Structures by Kernel Correlation and Graph Pooling," in *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2018, pp. 4548–4557. DOI: 10.1109/CVPR.2018.00478.

[154] D. Rezende and S. Mohamed, "Variational Inference with Normalizing Flows," in *International Conference on Machine Learning*, PMLR, 2015, pp. 1530–1538. [Online]. Available: http://proceedings.mlr.press/v37/rezende15.html.

[155] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural Ordinary Differential Equations," 2018. arXiv: 1806.07366 [cs.LG].

[156] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models," 2018. arXiv: 1810.01367 [cs.LG].

[157] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point Cloud Upsampling Network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2790–2799. DOI: 10.1109/CVPR.2018.00295.

[158] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, "Patch-based Progressive 3D Point Set Upsampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5958–5967. DOI: 10.1109/CVPR.2019.00611.

[159] R. Cai *et al.*, "Learning Gradient Fields for Shape Generation," vol. 12348, pp. 364–381, 2020. DOI: 10.1007/978-3-030-58580-8_22.

[160] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra, "PCPNet Learning Local Shape Properties from Raw Point Clouds," in *Computer Graphics Forum*, Wiley Online Library, vol. 37, 2018, pp. 75–85. DOI: 10.1111/cgf.13343.

[161] Y. Ben-Shabat and S. Gould, "DeepFit: 3D Surface Fitting via Neural Network Weighted Least Squares," in *European Conference on Computer Vision*, Springer, 2020, pp. 20–34. DOI: 10.1007/978-3-030-58452-8_2.

[162] M. Atzmon, H. Maron, and Y. Lipman, "Point Convolutional Neural Networks by Extension Operators," vol. 37, no. 4, pp. 1–12, 2018. DOI: 10.1145/3197517.3201301.

[163] J. Mao, X. Wang, and H. Li, "Interpolated Convolutional Networks for 3D Point Cloud Understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1578–1587. DOI: 10.1109/ICCV.2019.00166.

[164] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "PointWeb: Enhancing Local Neighborhood Features for Point Cloud Processing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5565–5573. DOI: 10.1109/CVPR.2019.00571.

[165] S. Lan, R. Yu, G. Yu, and L. S. Davis, "Modeling Local Geometric Structure of 3D Point Clouds using Geo-CNN," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 998–1008. DOI: 10.1109/CVPR.2019.00109.

[166] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep Convolutional Networks on 3D Point Clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9621–9630. DOI: 10.1109/CVPR.2019.00985.

[167] Y. Zhao, T. Birdal, J. E. Lenssen, E. Menegatti, L. Guibas, and F. Tombari, "Quaternion Equivariant Capsule Networks for 3D Point Clouds," in *European Conference on Computer Vision*, Springer, 2020, pp. 1–19. DOI: 10.1007/978-3-030-58452-8_1.

[168] F. Engelmann, T. Kontogianni, and B. Leibe, "Dilated Point Convolutions: On the Receptive Field Size of Point Convolutions on 3D Point Clouds," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 9463–9469. DOI: 10.1109/ICRA40945.2020.9197503.

[169] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, "Tangent Convolutions for Dense Prediction in 3D," in *Proceedings of the IEEE Conference on Computer*

Vision and Pattern Recognition, 2018, pp. 3887–3896. DOI: 10.1109/CVPR.2018.00409.

[170] H. Su *et al.*, "SPLATNet: Sparse Lattice Networks for Point Cloud Processing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2530–2539. DOI: 10.1109/CVPR.2018.00268.

[171] S. Shi *et al.*, "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 529–10 538. DOI: 10.1109/cvpr42600.2020.01054.

[172] M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. J. Mitra, and M. Ovsjanikov, "PointCleanNet: Learning to Denoise and Remove Outliers from Dense Point Clouds," in *Computer Graphics Forum*, Wiley Online Library, vol. 39, 2020, pp. 185–203. DOI: 10.1111/cgf.13753.

[173] E. J. Smith, S. Fujimoto, A. Romero, and D. Meger, "Geometrics: Exploiting Geometric Structure for Graph-Encoded Objects," 2019. arXiv: 1901.11461 [cs.CV].

[174] J. K. Pontes, C. Kong, S. Sridharan, S. Lucey, A. Eriksson, and C. Fookes, "Image2Mesh: A Learning Framework for Single Image 3D Reconstruction," in *Asian Conference on Computer Vision*, Springer, 2018, pp. 365–381. DOI: 10.1007/978-3-030-20887-5_23.

[175] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. DOI: 10.1109/cvpr.2016.90.

[176] F. Williams, T. Schneider, C. Silva, D. Zorin, J. Bruna, and D. Panozzo, "Deep Geometric Prior for Surface Reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 130–10 139. DOI: 10.1109/CVPR.2019.01037.

[177] A. Kar, C. Häne, and J. Malik, "Learning a Multi-View Stereo Machine," 2017. arXiv: 1708.05375 [cs.CV].

[178] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "MeshCNN: a Network with an Edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019. DOI: 10.1145/3306346.3322959.

[179] D. Coquelin *et al.*, "Accelerating Neural Network Training with Distributed Asynchronous and Selective Optimization (DASO)," *Journal of Big Data*, vol. 9, no. 1, pp. 1–18, 2022. DOI: 10.1186/s40537-021-00556-1.

[180] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969. DOI: 10.1109/TPAMI.2018.2844175.

[181] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "GraphRNN: Generating Realistic Graphs with Deep Auto-Regressive Models," in *International Conference on Machine Learning*, PMLR, vol. 80, 2018, pp. 5708–5717. [Online]. Available: http://proceedings.mlr.press/v80/you18a/you18a.pdf.

[182] A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel Recurrent Neural Networks," in *International Conference on Machine Learning*, PMLR, 2016, pp. 1747–1756. [Online]. Available: http://proceedings.mlr.press/v48/oord16.pdf.

[183] K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989. DOI: 10.1016/0893-6080(89)90020-8.

[184] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface Reconstruction from Unorganized Points," in *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '92)*, ACM, 1992, pp. 71–78. DOI: 10.1145/142920.134011.

[185] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Stoll, and C. Theobalt, "PatchNets: Patch-Based Generalizable Deep Implicit 3D Shape Representations," in *European Conference on Computer Vision*, Springer, 2020, pp. 293–309. DOI: 10.1007/978-3-030-58517-4_18.

[186] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser, "Local Deep Implicit Functions for 3D Shape," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4857–4866. DOI: 10.1109/cvpr42600.2020.00491.

[187] J. Chibane and G. Pons-Moll, "Implicit Feature Networks for Texture Completion from Partial 3D Data," in *European Conference on Computer Vision*, Springer, 2020, pp. 717–725. DOI: 10.1007/978-3-030-66096-3_48.

[188] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *International Conference on Medical Image Computing and Computer-assisted Intervention*, Springer, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[189] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation," in *International Conference on Medical Image Computing and Computer-assisted Intervention*, Springer, 2016, pp. 424–432. DOI: 10.1007/978-3-319-46723-8_49.

[190] S. Saito, T. Simon, J. Saragih, and H. Joo, "PIFuHD: Multi-level Pixel-aligned Implicit Function for High-Resolution 3D Human Digitization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 84–93. DOI: 10.1109/cvpr42600.2020.00016.

[191] B. L. Bhatnagar, C. Sminchisescu, C. Theobalt, and G. Pons-Moll, "Combining Implicit Function Learning and Parametric Models for 3D Human Reconstruction," 2020. arXiv: 2007.11432 [cs.CV].

[192] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Occupancy Flow: 4D Reconstruction by

Learning Particle Dynamics," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5379–5389. DOI: 10.1109/ICCV.2019.00548.

[193] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, "PIFU: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2304–2314. DOI: 10.1109/ICCV.2019.00239.

[194] M. Oechsle, L. Mescheder, M. Niemeyer, T. Strauss, and A. Geiger, "Texture Fields: Learning Texture Representations in Function Space," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4531–4540. DOI: 10.1109/ICCV.2019.00463.

[195] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations," 2019. arXiv: 1906.01618 [cs.CV].

[196] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable Volumetric Rendering: Learning Implicit 3D Representations without 3D Supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3504–3515. DOI: 10.1109/CVPR42600.2020.00356.

[197] L. Yariv *et al.*, "Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance," *Advances in Neural Information Processing Systems*, vol. 33, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/1a77befc3b608d6ed363567685f70e1e-Paper.pdf.

[198] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann, "DISN: Deep Implicit Surface Network for High-quality Single-View 3D Reconstruction," 2019. arXiv: 1905.10711 [cs.CV].

[199] S. Liu, Y. Zhang, S. Peng, B. Shi, M. Pollefeys, and Z. Cui, "DIST: Rendering Deep Implicit Signed Distance Function with Differentiable Sphere Tracing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2019–2028. DOI: 10.1109/cvpr42600.2020.00209.

[200] Y. Jiang, D. Ji, Z. Han, and M. Zwicker, "SDFDiff: Differentiable Rendering of Signed Distance Fields for 3D Shape Optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1251–1261. DOI: 10.1109/cvpr42600.2020.00133.

[201] Y. Duan, H. Zhu, H. Wang, L. Yi, R. Nevatia, and L. J. Guibas, "Curriculum DeepSDF," in *European Conference on Computer Vision*, Springer, 2020, pp. 51–67. DOI: 10.1007/978-3-030-58598-3_4.

[202] T. Takikawa *et al.*, "Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes," 2021. arXiv: 2101.10994 [cs.CV].

[203] S. Duggal *et al.*, "Secrets of 3D Implicit Object Shape Reconstruction in the Wild," 2021. arXiv: 2101.06860 [cs.CV].

[204] S. Liu, S. Saito, W. Chen, and H. Li, "Learning to Infer Implicit Surfaces without 3D Supervision," 2019. arXiv: 1911.00767 [cs.CV].

[205] E. Chatzipantazis, S. Pertigkiozoglou, E. Dobriban, and K. Daniilidis, "Se(3)-equivariant attention networks for shape reconstruction in function space," *arXiv preprint arXiv:2204.02394*, 2022. DOI: 10.48550/arXiv.2204.02394.

[206] F. Fuchs, D. Worrall, V. Fischer, and M. Welling, "Se(3)-transformers: 3d roto-translation equivariant attention networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1970–1981, 2020.

[207] N. Thomas *et al.*, "Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds," *arXiv preprint arXiv:1802.08219*, 2018.

[208] C. Deng, O. Litany, Y. Duan, A. Poulenard, A. Tagliasacchi, and L. J. Guibas, "Vector neurons: A general framework for so(3)-equivariant networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 200–12 209. DOI: 10.1109/ICCV48922.2021.01198.

[209] Y. Chen, B. Fernando, H. Bilen, M. Nießner, and E. Gavves, "3d equivariant graph implicit functions," in *European Conference on Computer Vision*, Springer, 2022, pp. 485–502. DOI: 10.48550/arXiv.2203.17178.

[210] A. Vaswani *et al.*, "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008. [Online]. Available: https://papers.nips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[211] Q. Wang *et al.*, "IBRNet: Learning Multi-View Image-based Rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4690–4699. DOI: 10.1109/CVPR46437.2021.00466.

[212] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5855–5864. DOI: 10.1109/ICCV48922.2021.00580.

[213] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5470–5479. DOI: 10.1109/CVPR52688.2022.00539.

[214] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Zip-nerf: Anti-aliased grid-based neural radiance fields," *arXiv preprint arXiv:2304.06706*, 2023. DOI: 10.48550/arXiv.2304.06706.

[215] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash

encoding," *ACM Transactions on Graphics (ToG)*, vol. 41, no. 4, pp. 1–15, 2022. DOI: 10.1145/3528223.3530127.

[216] S. Bi *et al.*, "Neural Reflectance Fields for Appearance Acquisition," *arXiv preprint arXiv:2008.03824*, 2020.

[217] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron, "NeRFactor: Neural Factorization of Shape and Reflectance Under an Unknown Illumination," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 6, pp. 1–18, 2021. DOI: 10.1145/3478513.3480496.

[218] Z. Wang, S. Wu, W. Xie, M. Chen, and V. A. Prisacariu, "NeRF–: Neural Radiance Fields without Known Camera Parameters," *arXiv preprint arXiv:2102.07064*, 2021.

[219] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, "D-NeRF: Neural Radiance Fields for Dynamic Scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 318–10 327. DOI: 10.1109/CVPR46437.2021.01018.

[220] Z. Li, S. Niklaus, N. Snavely, and O. Wang, "Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6498–6508. DOI: 10.1109/CVPR46437.2021.00643.

[221] K. Park *et al.*, "HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields," *arXiv preprint arXiv:2106.13228*, 2021.

[222] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, "Baking Neural Radiance Fields for Real-Time View Synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5875–5884. DOI: 10.1109/ICCV48922.2021.00582.

[223] K. Gao, Y. Gao, H. He, D. Lu, L. Xu, and J. Li, "NeRF: Neural Radiance Field in 3D Vision, A Comprehensive Review," *arXiv preprint arXiv:2210.00379*, 2022.

[224] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 171–27 183, 2021. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/e41e164f7485ec4a28741a2d0ea41c74-Paper.pdf.

[225] M. Oechsle, S. Peng, and A. Geiger, "UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5589–5599.

[226] Y. Wang, I. Skorokhodov, and P. Wonka, "HF-NeuS: Improved Surface Reconstruction using High-Frequency Details," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1966–1978, 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/0ce8e3434c7b486bbddff9745b2a1722-Paper-Conference.pdf.

[227] Q. Fu, Q. Xu, Y. S. Ong, and W. Tao, "Geo-Neus: Geometry-Consistent Neural Implicit Surfaces Learning for Multi-View Reconstruction," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3403–3416, 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/16415eed5a0a121bfce79924db05d3fe-Paper-Conference.pdf.

[228] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume Rendering of Neural Implicit Surfaces," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4805–4815, 2021. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/25e2a30f44898b9f3e978b1786dcd85c-Paper.pdf.

[229] Z. Yu *et al.*, *Sdfstudio: A unified framework for surface reconstruction*, 2022. [Online]. Available: https://github.com/autonomousvision/sdfstudio.

[230] M. Tancik *et al.*, "Nerfstudio: A modular framework for neural radiance field development," in *ACM SIGGRAPH 2023 Conference Proceedings*, ser. SIGGRAPH '23, 2023. DOI: 10.1145/3588432.3591516.

[231] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond Pascal: A Benchmark for 3D Object Detection in the Wild," in *IEEE Winter Conference on Applications of Computer Vision*, IEEE, 2014, pp. 75–82. DOI: 10.1109/WACV.2014.6836101.

[232] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep Metric Learning via Lifted Structured Feature Embedding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4004–4012. DOI: 10.1109/CVPR.2016.434.

[233] J. J. Lim, H. Pirsiavash, and A. Torralba, "Parsing IKEA Objects: Fine Pose Estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2992–2999. DOI: 10.1109/ICCV.2013.372.

[234] S. Choi, Q.-Y. Zhou, S. Miller, and V. Koltun, "A Large Dataset of Object Scans," 2016. arXiv: 1602.02481 [cs.CV].

[235] Trimble Inc., *3D Warehouse*, 2023. [Online]. Available: https://3dwarehouse.sketchup.com/.

[236] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs, "Large scale multi-view stereopsis evaluation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 406–413.

[237] Y. Yao *et al.*, "Blendedmvs: A large-scale dataset for generalized multi-view stereo networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1790–1799.

[238] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison, "Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation?" In *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2678–2687. DOI: 10.1109/ICCV.2017.292.

[239] A. Bozic, P. Palafox, J. Thies, A. Dai, and M. Niessner, "TransformerFusion: Monocular RGB Scene Reconstruction using Transformers," in *Advances in Neural Information Processing Systems*, vol. 33, 2021. [Online]. Available: https : / / proceedings . neurips . cc / paper _ files / paper / 2021 / file / 0a87257e5308197df43230edf4ad1dae-Paper.pdf.

[240] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point Transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 259–16 268. [Online]. Available: https : / / openaccess . thecvf . com / content / ICCV2021 / papers / Zhao_Point_Transformer_ICCV_2021_paper.pdf.

[241] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "PCT: Point Cloud Transformer," *Computational Visual Media*, vol. 7, no. 2, pp. 187–199, 2021. DOI: 10.1007/s41095-021-0229-5.

[242] Y. LeCun and I. Misra, *Self-supervised learning: The dark matter of intelligence*, 2021. [Online]. Available: https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/.

[243] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," in *International Conference on Machine Learning*, PMLR, vol. 37, 2015, pp. 2256–2265. [Online]. Available: http : / / proceedings.mlr.press/v37/sohl-dickstein15.pdf.

[244] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020. [Online]. Available: https://proceedings. neurips . cc / paper _ files / paper / 2020 / file / 4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[245] A. Q. Nichol and P. Dhariwal, "Improved Denoising Diffusion Probabilistic Models," in *International Conference on Machine Learning*, PMLR, 2021, pp. 8162–8171. [Online]. Available: http : / / proceedings . mlr . press/v139/nichol21a/nichol21a.pdf.

[246] P. Dhariwal and A. Nichol, "Diffusion Models Beat GANs on Image Synthesis," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8780–8794, 2021. [Online]. Available: https : / / proceedings . neurips . cc / paper / 2021 / file / 49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf.

[247] S. Luo and W. Hu, "Diffusion Probabilistic Models for 3D Point Cloud Generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2837–2845. DOI: 0 . 1109/CVPR46437.2021.00286.

[248] L. Zhou, Y. Du, and J. Wu, "3D Shape Generation and Completion Through Point-Voxel Diffusion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5826–5835. DOI: 10.1109/ICCV48922.2021.00577.