

# Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era

Xian-Feng Han\*, Hamid Laga\*, Mohammed Bennamoun *Senior Member, IEEE*

**Abstract**—3D reconstruction is a longstanding ill-posed problem, which has been explored for decades by the computer vision, computer graphics, and machine learning communities. Since 2015, image-based 3D reconstruction using convolutional neural networks (CNN) has attracted increasing interest and demonstrated an impressive performance. Given this new era of rapid evolution, this article provides a comprehensive survey of the recent developments in this field. We focus on the works which use deep learning techniques to estimate the 3D shape of generic objects either from a single or multiple RGB images. We organize the literature based on the shape representations, the network architectures, and the training mechanisms they use. While this survey is intended for methods which reconstruct generic objects, we also review some of the recent works which focus on specific object classes such as human body shapes and faces. We provide an analysis and comparison of the performance of some key papers, summarize some of the open problems in this field, and discuss promising directions for future research.

**Index Terms**—3D Reconstruction, Depth Estimation, SLAM, SfM, CNN, Deep Learning, LSTM, 3D face, 3D Human Body, 3D Video.

## 1 INTRODUCTION

The goal of image-based 3D reconstruction is to infer the 3D geometry and structure of objects and scenes from one or multiple 2D images. This long standing ill-posed problem is fundamental to many applications such as robot navigation, object recognition and scene understanding, 3D modeling and animation, industrial control, and medical diagnosis.

Recovering the lost dimension from just 2D images has been the goal of classic multiview stereo and shape-from-X methods, which have been extensively investigated for many decades. The first generation of methods approached the problem from the geometric perspective; they focused on understanding and formalizing, mathematically, the 3D to 2D projection process, with the aim to devise mathematical or algorithmic solutions to the ill-posed inverse problem. Effective solutions typically require multiple images, captured using accurately calibrated cameras. Stereo-based techniques [1], for example, require matching features across images captured from slightly different viewing angles, and then use the triangulation principle to recover the 3D coordinates of the image pixels. Shape-from-silhouette, or shape-by-space-carving, methods [2] require accurately segmented 2D silhouettes. These methods, which have led to reasonable quality 3D reconstructions, require multiple images of

the same object captured by well-calibrated cameras. This, however, may not be practical or feasible in many situations.

Interestingly, humans are good at solving such ill-posed inverse problems by leveraging prior knowledge. They can infer the approximate size and rough geometry of objects using only one eye. They can even guess what it would look like from another viewpoint. We can do this because all the previously seen objects and scenes have enabled us to build prior knowledge and develop mental models of what objects look like. The second generation of 3D reconstruction methods tried to leverage this prior knowledge by formulating the 3D reconstruction problem as a recognition problem. The avenue of deep learning techniques, and more importantly, the increasing availability of large training data sets, have led to a new generation of methods that are able to recover the 3D geometry and structure of objects from one or multiple RGB images without the complex camera calibration process. Despite being recent, these methods have demonstrated exciting and promising results on various tasks related to computer vision and graphics.

In this article, we provide a comprehensive and structured review of the recent advances in 3D object reconstruction using deep learning techniques. We first focus on generic shapes and then discuss specific cases, such as human body shapes faces reconstruction, and 3D scene parsing. We have gathered 149 papers, which appeared since 2015 in leading computer vision, computer graphics, and machine learning conferences and journals<sup>1</sup>. The goal is to help the reader navigate in this emerging field, which gained a significant momentum in the past few years. Compared to the existing literature, the main contributions of

- (\* Joint first author) Xian-Feng Han is with College of Computer and Information Science, Southwest University, Chongqing 400715, China, with Tianjin University, Tianjin, 300350, China and with the University of Western Australia, Perth, WA 6009, Australia.
- (\* Joint first author) Hamid Laga is with the Information Technology, Mathematics and Statistics Discipline, Murdoch University (Australia), and with the Phenomics and Bioinformatics Research Centre, University of South Australia. E-mail: H.Laga@murdoch.edu.au
- Mohammed Bennamoun is with the University of Western Australia, Perth, WA 6009, Australia. Email: mohammed.bennamoun@uwa.edu.au

Manuscript received April 19, 2005; revised December 27, 2012.

1. This continuously and rapidly increasing number, even at the time we are finalising this article, does not include many of the CVPR2019 and the upcoming ICCV2019 papers.

this article are as follows;

- 1) To the best of our knowledge, this is the first survey paper in the literature which focuses on image-based 3D object reconstruction using deep learning.
- 2) We cover the contemporary literature with respect to this area. We present a comprehensive review of 149 methods, which appeared since 2015.
- 3) We provide a comprehensive review and an insightful analysis on all aspects of 3D reconstruction using deep learning, including the training data, the choice of network architectures and their effect on the 3D reconstruction results, the training strategies, and the application scenarios.
- 4) We provide a comparative summary of the properties and performance of the reviewed methods for generic 3D object reconstruction. We cover 88 algorithms for generic 3D object reconstruction, 11 methods related to 3D face reconstruction, and 6 methods for 3D human body shape reconstruction.
- 5) We provide a comparative summary of the methods in a tabular form.

The rest of this article is organized as follows; Section 2 formulates the problem and lays down the taxonomy. Section 3 reviews the latent spaces and the input encoding mechanisms. Section 4 surveys the volumetric reconstruction techniques, while Section 5 focuses on surface-based techniques. Section 6 shows how some of the state-of-the-art techniques use additional cues to boost the performance of 3D reconstruction. Section 7 discusses the training procedures. Section 8 focuses on specific objects such as human body shapes and faces. Section 9 summarizes the most commonly used datasets to train, test, and evaluate the performance of various deep learning-based 3D reconstruction algorithms. Section 10 compares and discusses the performance of some key methods. Finally, Section 11 discusses potential future research directions while Section 12 concludes the paper with some important remarks.

## 2 PROBLEM STATEMENT AND TAXONOMY

Let  $\mathbf{I} = \{I_k, k = 1, \dots, n\}$  be a set of  $n \geq 1$  RGB images of one or multiple objects  $X$ . 3D reconstruction can be summarized as the process of learning a predictor  $f_\theta$  that can infer a shape  $\hat{X}$  that is as close as possible to the unknown shape  $X$ . In other words, the function  $f_\theta$  is the minimizer of a reconstruction objective  $\mathcal{L}(\mathbf{I}) = d(f_\theta(\mathbf{I}), X)$ . Here,  $\theta$  is the set of parameters of  $f$  and  $d(\cdot, \cdot)$  is a certain measure of distance between the target shape  $X$  and the reconstructed shape  $f(\mathbf{I})$ . The reconstruction objective  $\mathcal{L}$  is also known as the *loss function* in the deep learning literature.

This survey discusses and categorizes the state-of-the-art based on the nature of the input  $\mathbf{I}$ , the representation of the output, the deep neural network architectures used during training and testing to approximate the predictor  $f$ , the training procedures they use, and their degree of supervision, see Table 1 for a visual summary. In particular, the *input I* can be (1) a single image, (2) multiple images captured using RGB cameras whose intrinsic and extrinsic parameters can be *known* or *unknown*, or (3) a video stream, i.e., a sequence of images with temporal correlation. The

TABLE 1: Taxonomy of the state-of-the-art image-based 3D object reconstruction using deep learning.

Input	Training	1 vs. multi RGB, 3D ground truth, Segmentation.	One vs. multiple objects, Uniform vs. cluttered background.
	Testing	1 vs. multi RGB, Segmentation	
Output	Volumetric	High vs. low resolution	
	Surface	Parameterization, template deformation, Point cloud.	
		Direct vs. intermediating	
Network architecture	Architecture at training		Architecture at testing
	Encoder - Decoder TL-Net (Conditional) GAN		Encoder - Decoder 3D-VAE
	3D-VAE-GAN		
Training	Degree of supervision	2D vs. 3D supervision. Weak supervision. Loss functions.	
	Training procedure	Adversarial training. Joint 2D-3D embedding. Joint training with other tasks.	

first case is very challenging because of the ambiguities in the 3D reconstruction. When the input is a video stream, one can exploit the temporal correlation to facilitate the 3D reconstruction while ensuring that the reconstruction is smooth and consistent across all the frames of the video stream. Also, the input can be depicting one or multiple 3D objects belonging to known or unknown shape categories. It can also include additional information such as silhouettes, segmentation masks, and semantic labels as priors to guide the reconstruction.

The representation of the output is crucial to the choice of the network architecture. It also impacts the computational efficiency and quality of the reconstruction. In particular,

- *Volumetric representations*, which have been extensively adopted in early deep learning-based 3D reconstruction techniques, allow the parametrization of 3D shapes using regular voxel grids. As such, 2D convolutions used in image analysis can be easily extended to 3D. They are, however, very expensive in terms of memory requirements, and only a few techniques can achieve sub-voxel accuracy.
- *Surface-based representations*: Other papers explored surface-based representations such as meshes and point clouds. While being memory-efficient, such representations are not regular structures and thus, they do not easily fit into deep learning architectures.
- *Intermediation*: While some 3D reconstruction algorithms predict the 3D geometry of an object from RGB images directly, others decompose the problem into sequential steps, each step predicts an intermediate representation.

A variety of *network architectures* have been utilized to implement the predictor  $f$ . The backbone architecture, which can be different during training and testing, is composed of an encoder  $h$  followed by a decoder  $g$ , i.e.,  $f = g \circ h$ . The encoder maps the input into a latent variable  $\mathbf{x}$ , referred to as a feature vector or a code, using a sequence of convolutions and pooling operations, followed by fully connected layers of neurons. The decoder, also called the generator, decodes the feature vector into the desired output by using either fully connected layers or a deconvolution network (a sequence of convolution and upsampling operations, also referred to as upconvolutions). The former is suitable for unstructured output, e.g., 3D point clouds, while the latter

is used to reconstruct volumetric grids or parametrized surfaces. Since the introduction of this vanilla architecture, several extensions have been proposed by varying the architecture (e.g., ConvNet vs. ResNet, Convolutional Neural Networks (CNN) vs. Generative Adversarial Networks (GAN), CNN vs. Variational Auto-Encoders, and 2D vs. 3D convolutions), and by cascading multiple blocks each one achieving a specific task.

While the architecture of the network and its building blocks are important, the performance depends highly on the way it is trained. In this survey, we will look at:

- *Datasets*: There are various datasets that are currently available for training and evaluating deep learning-based 3D reconstruction. Some of them use real data, others are CG-generated.
- *Loss functions*: The choice of the loss function can significantly impact on the reconstruction quality. It also defines the degree of supervision.
- *Training procedure and degree of supervision*: Some methods require real images annotated with their corresponding 3D models, which are very expensive to obtain. Other methods rely on a combination of real and synthetic data. Others avoid completely 3D supervision by using loss functions that exploit supervisory signals that are easy to obtain.

The following sections review in detail these aspects.

### 3 THE ENCODING STAGE

Deep learning-based 3D reconstruction algorithms encode the input  $\mathbf{I}$  into a feature vector  $\mathbf{x} = h(\mathbf{I}) \in \mathcal{X}$  where  $\mathcal{X}$  is the latent space. A good mapping function  $h$  should satisfy the following properties:

- Two inputs  $\mathbf{I}_1$  and  $\mathbf{I}_2$  that represent similar 3D objects should be mapped into  $\mathbf{x}_1$  and  $\mathbf{x}_2 \in \mathcal{X}$  that are close to each other in the latent space.
- A small perturbation  $\partial\mathbf{x}$  of  $\mathbf{x}$  should correspond to a small perturbation of the shape of the input.
- The latent representation induced by  $h$  should be invariant to extrinsic factors such as the camera pose.
- A 3D model and its corresponding 2D images should be mapped onto the same point in the latent space. This will ensure that the representation is not ambiguous and thus facilitate the reconstruction.

The first two conditions have been addressed by using encoders that map the input onto discrete (Section 3.1) or continuous (Section 3.2) latent spaces. These can be flat or hierarchical (Section 3.3). The third one has been addressed by using disentangled representations (Section 3.4). The latter has been addressed by using TL-architectures during the training phase. This is covered in Section 7.3.1 as one of the many training mechanisms that have been used in the literature. Table 2 summarizes this taxonomy.

#### 3.1 Discrete latent spaces

Wu *et al.* in their seminal work [3] introduced 3D ShapeNet, an encoding network which maps a 3D shape, represented as a discretized volumetric grid of size  $30^3$ , into a latent

TABLE 2: Taxonomy of the encoding stage. FC: fully-connected layers. VAE: Variational Auto-Encoder.

Latent spaces	Architectures
Discrete (3.1) vs. continuous (3.2)	ConvNet, ResNet,
Flat vs. hierarchical (3.3)	FC, 3D-VAE
Disentangled representation (3.4)	

representation of size  $4000 \times 1$ . Its core network is composed of  $n_{conv} = 3$  convolutional layers (each one using 3D convolution filters), followed by  $n_{fc} = 3$  fully connected layers. This standard vanilla architecture has been used for 3D shape classification and retrieval [3], and for 3D reconstruction from depth maps represented as voxel grids [3]. It has also been used in the 3D encoding branch of the TL architectures during the training of 3D reconstruction networks, see Section 7.3.1.

2D encoding networks that map input images into a latent space follow the same architecture as 3D ShapeNet [3] but use 2D convolutions [4], [5], [6], [7], [8], [9], [10], [11]. Early works differ in the type and number of layers they use. For instance, Yan *et al.* [4] use  $n_{conv} = 3$  convolutional layers with 64, 128, and 256 channels, respectively, and  $n_{fc} = 3$  fully-connected layers with 1024, 1024, and 512 neurons, respectively. Wiles and Zisserman [10] use  $n_{conv} = 6$  convolutional layers of 3, 64, 128, 256, 128, and 160 channels, respectively. Other works add pooling layers [7], [12], and leaky Rectified Linear Units (ReLU) [7], [12], [13]. For example, Wiles and Zisserman [10] use max pooling layers between each pair of convolutional layers, except after the first layer and before the last layer. ReLU layers improve learning since the gradient during the back propagation is never zero.

Both 3D shape and 2D image encoding networks can be implemented using deep residual networks (ResNet) [14], which add residual connections between the convolutional layers, see for example [6], [7], [9]. Compared to conventional networks such as VGGNet [15], ResNets improve and speed up the learning process for very deep networks.

#### 3.2 Continuous latent spaces

Using the encoders presented in the previous section, the latent space  $\mathcal{X}$  may not be continuous and thus it does not allow easy interpolation. In other words, if  $\mathbf{x}_1 = h(\mathbf{I}_1)$  and  $\mathbf{x}_2 = h(\mathbf{I}_2)$ , then there is no guarantee that  $\frac{1}{2}(\mathbf{x}_1 + \mathbf{x}_2)$  can be decoded into a valid 3D shape. Also, small perturbations of  $\mathbf{x}_1$  do not necessarily correspond to small perturbations of the input. Variational Autoencoders (VAE) [16] and their 3D extension (3D-VAE) [17] have one fundamentally unique property that makes them suitable for generative modeling: their latent spaces are, by design, continuous, allowing easy sampling and interpolation. The key idea is that instead of mapping the input into a feature vector, it is mapped into a mean vector  $\mu$  and a vector of standard deviations  $\sigma$  of a multivariate Gaussian distribution. A sampling layer then takes these two vectors, and generates, by random sampling from the Gaussian distribution, a feature vector  $\mathbf{x}$ , which will serve as input to the subsequent decoding stages.

This architecture has been used to learn continuous latent spaces for volumetric [17], [18], depth-based [19],

surface-based [20], and point-based [21], [22] 3D reconstruction. In Wu *et al.* [17], for example, the image encoder takes a  $256 \times 256$  RGB image and outputs two 200-dimensional vectors representing, respectively, the mean and the standard deviation of a Gaussian distribution in the 200-dimensional space. Compared to standard encoders, 3D-VAE can be used to randomly sample from the latent space, to generate variations of an input, and to reconstruct multiple plausible 3D shapes from an input image [21], [22]. It generalizes well to images that have not been seen during the training.

### 3.3 Hierarchical latent spaces

Liu *et al.* [18] showed that encoders that map the input into a single latent representation cannot extract rich structures and thus may lead to blurry reconstructions. To improve the quality of the reconstruction, Liu *et al.* [18] introduced a more complex internal variable structure, with the specific goal of encouraging the learning of a hierarchical arrangement of latent feature detectors. The approach starts with a global latent variable layer that is hardwired to a set of local latent variable layers, each tasked with representing one level of feature abstraction. The skip-connections tie together the latent codes in a top-down directed fashion: local codes closer to the input will tend to represent lower-level features while local codes farther away from the input will tend towards representing higher-level features. Finally, the local latent codes are concatenated to a flattened structure when fed into the task-specific models such as 3D reconstruction.

### 3.4 Disentangled representation

The appearance of an object in an image is affected by multiple factors such as the object’s shape, the camera pose, and the lighting conditions. Standard encoders represent all these variabilities in the learned code  $\mathbf{x}$ . This is not desirable in applications such as recognition and classification, which should be invariant to extrinsic factors such as pose and lighting [23]. 3D reconstruction can also benefit from disentangled representations where shape, pose, and lighting are represented with different codes. To this end, Grant *et al.* [5] proposed an encoder, which maps an RGB image into a shape code and a transformation code. The former is decoded into a 3D shape. The latter, which encodes lighting conditions and pose, is decoded into (1) another  $80 \times 80$  RGB image with correct lighting, using upconvolutional layers, and (2) camera pose using fully-connected layers (FC). To enable a disentangled representation, the network is trained in such a way that in the forward pass, the image decoder receives input from the shape code and the transformation code. In the backward pass, the signal from the image decoder to the shape code is suppressed to force it to only represent shape.

Zhu *et al.* [24] followed the same idea by decoupling the 6DOF pose parameters and shape. The network reconstructs from the 2D input the 3D shape but in a canonical pose. At the same time, a pose regressor estimates the 6DOF pose parameters, which are then applied to the reconstructed canonical shape. Decoupling pose and shape reduces the number of free parameters in the network, which results in improved efficiency.

## 4 VOLUMETRIC DECODING

Volumetric representations discretize the space around a 3D object into a 3D voxel grid  $V$ . The finer the discretization is, the more accurate the representation will be. The goal is then to recover a grid  $\hat{V} = f_\theta(\mathbf{I})$  such that the 3D shape  $\hat{X}$  it represents is as close as possible to the unknown real 3D shape  $X$ . The main advantage of using volumetric grids is that many of the existing deep learning architectures that have been designed for 2D image analysis can be easily extended to 3D data by replacing the 2D pixel array with its 3D analogue and then processing the grid using 3D convolution and pooling operations. This section looks at the different volumetric representations (Section 4.1) and reviews the decoder architectures for low-resolution (Section 4.2) and high-resolution (Section 4.3) 3D reconstruction.

### 4.1 Volumetric representations of 3D shapes

There are four main volumetric representations that have been used in the literature:

- *Binary occupancy grid.* In this representation, a voxel is set to one if it belongs to the objects of interest, whereas background voxels are set to zero.
- *Probabilistic occupancy grid.* Each voxel in a probabilistic occupancy grid encodes its probability of belonging to the objects of interest.
- *The Signed Distance Function (SDF).* Each voxel encodes its signed distance to the closest surface point. It is negative if the voxel is located inside the object and positive otherwise.
- *Truncated Signed Distance Function (TSDF).* Introduced by Curless and Levoy [37], TSDF is computed by first estimating distances along the lines of sight of a range sensor, forming a projective signed distance field, and then truncating the field at small negative and positive values.

Probabilistic occupancy grids are particularly suitable for machine learning algorithms which output likelihoods. SDFs provide an unambiguous estimate of surface positions and normal directions. However, they are not trivial to construct from partial data such as depth maps. TSDFs sacrifice the full signed distance field that extends indefinitely away from the surface geometry, but allow for local updates of the field based on partial observations. They are suitable for reconstructing 3D volumes from a set of depth maps [26], [31], [35], [38].

In general, volumetric representations are created by regular sampling of the volume around the objects. Knyaz *et al.* [30] introduced a representation method called Frustum Voxel Model or Fruxel, which combines the depth representation with voxel grids. It uses the slices of the camera’s 3D frustum to build the voxel space, and thus provides precise alignment of voxel slices with the contours in the input image.

Also, common SDF and TSDF representations are discretised into a regular grid. Recently, however, Park *et al.* [39] proposed Deep SDF (deepSDF), a generative deep learning model that produces a continuous SDF field from an input point cloud. Unlike the traditional SDF representation, DeepSDF can handle noisy and incomplete data. It can also represent an entire class of shape

TABLE 3: Taxonomy of the various volumetric decoders used in the literature. Number in parentheses are the corresponding section numbers. MDN: Mixture Density Network. BBX: Bounding Box primitives. Part.: partitioning.

	Representation (4.1)		Resolution					Architecture	
	Sampling	Content	Low res: 32 <sup>3</sup> , 64 <sup>3</sup> (4.2)	High resolution (4.3)				Network	Intermediation (6.1)
				Space part. (4.3.1)	Shape part. (4.3.3)	Subspace param. (4.3.4)	Refinement (4.3.5)		
	Regular, Fruxel, Adaptive	Occupancy, SDF, TSDF	Normal, O-CNN, OctNet	HSP, OGN, Patch-guide	Parts, Patches	PCA, DCT	Upsampling, Volume slicing, Patch synthesis, Patch refinement	FC, UpConv.	(1) image → voxels, (2) image → (2.5D, silh.) → voxels
[7]	Regular	Occupancy	✓	—	—	—	—	LSTM + UpConv	image → voxels
[25]	Regular	Occupancy	✓	—	—	—	—	UpConv	image → voxels
[17]	Regular	Occupancy	✓	—	—	—	—	UpConv	image → voxels
[4]	Regular	Occupancy	✓	—	—	—	—	UpConv	image → voxels
[6]	Regular	Occupancy	✓	128 <sup>3</sup>	—	—	—	UpConv	(2)
[26]	Regular	SDF	✓	—	—	—	patch synthesis	UpConv	scans → voxels
[27]	Regular	Occupancy	✓	—	—	—	—	UpConv	image → voxels
[12]	Regular	Occupancy	✓	—	—	—	DCT	IDCT	image → voxels
[18]	Regular	Occupancy	✓	—	—	—	—	UpConv	image → voxels
[28]	Regular	Occupancy	—	128 <sup>3</sup>	—	—	Volume slicing	CNN → LSTM → CNN	image → voxels
[24]	Regular	Occupancy	✓	—	—	—	—	UpConv	image → voxels
[29]	Regular	TSDF	—	—	—	Parts	—	LSTM + MDN	depth → BBX
[30]	Fruxel	Occupancy	—	128 <sup>3</sup>	—	—	—	UpConv	image → voxels
[31]	Regular	TSDF	—	—	—	PCA	—	FC	image → voxels
[32]	Adaptive	—	—	O-CNN	—	—	—	—	—
[33]	Adaptive	—	—	OGN	—	—	—	—	—
[8]	Regular	Occupancy	✓	—	—	—	—	UpConv	image → voxels
[9]	Regular	Occupancy	—	128 <sup>3</sup>	—	—	—	UpConv	(2)
[34]	Adaptive	Occupancy	—	O-CNN	patch-guided	—	—	UpConv	image → voxels
[13]	Regular	Occupancy	✓	—	—	—	—	UpConv	image → voxels
[35]	Regular	TSDF	—	OctNet	—	—	Global to local	UpConv	scans → voxels
[11]	Regular	Occupancy	✓	—	—	—	—	UpConv	image → voxels
[36]	Adaptive	Occupancy	—	—	HSP	—	—	UpConv nets	image → voxels

## 4.2 Low resolution 3D volume reconstruction

Once a compact vector representation of the input is learned using an encoder, the next step is to learn the decoding function  $g$ , known as the *generator* or the *generative model*, which maps the vector representation into a volumetric voxel grid. The standard approach uses a *convolutional decoder*, called also *up-convolutional network*, which mirrors the convolutional encoder. Wu *et al.* [3] were among the first to propose this methodology to reconstruct 3D volumes from depth maps. Wu *et al.* [6] proposed a two-stage reconstruction network called MarrNet. The first stage uses an encoder-decoder architecture to reconstruct, from an input image, the depth map, the normal map, and the silhouette map. These three maps, referred to as 2.5 sketches, are then used as input to another encoder-decoder architecture, which regresses a volumetric 3D shape. The network has been later extended by Sun *et al.* [9] to also regress the pose of the input. The main advantage of this two-stage approach is that, compared to full 3D models, depth maps, normal maps, and silhouette maps are much easier to recover from 2D images. Likewise, 3D models are much easier to recover from these three modalities than from 2D images alone. This method, however, fails to reconstruct complex, thin structures.

Wu *et al.*'s work [3] has led to several extensions [7], [8], [17], [27], [40]. In particular, recent works tried to directly regress the 3D voxel grid [8], [11], [13], [18] without intermediation. Tulsiani *et al.* [8], and later in [11], used a decoder composed of 3D upconvolution layers to predict the voxel occupancy probabilities. Liu *et al.* [18] used a 3D up-convolutional neural network, followed by an element-wise logistic sigmoid, to decode the learned latent features into a 3D occupancy probability grid. These methods have been successful in performing 3D reconstruction from a single or

a collection of images captured with uncalibrated cameras. Their main advantage is that the deep learning architectures proposed for the analysis of 2D images can be easily adapted to 3D models by replacing the 2D up-convolutions in the decoder with 3D up-convolutions, which also can be efficiently implemented on the GPU. However, given the computational complexity and memory requirements, these methods produce low resolution grids, usually of size 32<sup>3</sup> or 64<sup>3</sup>. As such, they fail to recover fine details.

## 4.3 High resolution 3D volume reconstruction

There have been attempts to upscale the deep learning architectures for high resolution volumetric reconstruction. For instance, Wu *et al.* [6] were able to reconstruct voxel grids of size 128<sup>3</sup> by simply expanding the network. Volumetric grids, however, are very expensive in terms of memory requirements, which grow cubically with the grid resolution. This section reviews some of the techniques that have been used to infer high resolution volumetric grids, while keeping the computational and memory requirements tractable. We classify these methods into four categories based on whether they use space partitioning, shape partitioning, subspace parameterization, or coarse-to-fine refinement strategies.

### 4.3.1 Space partitioning

While regular volumetric grids facilitate convolutional operations, they are very sparse since surface elements are contained in few voxels. Several papers have exploited this sparsity to address the resolution problem [32], [33], [41], [42]. They were able to reconstruct 3D volumetric grids of size 256<sup>3</sup> to 512<sup>3</sup> by using space partitioning techniques such as octrees. There are, however, two main challenging issues when using octree structures for deep-learning

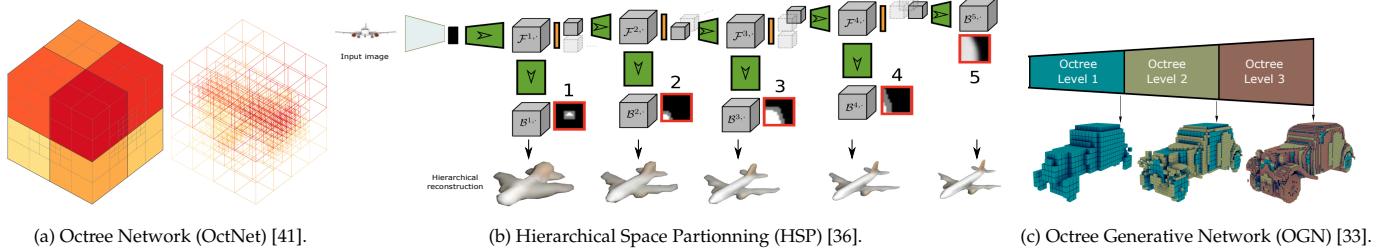


Fig. 1: Space partitioning. OctNet [41] is a hybrid grid-octree, which enables deep and high-resolution 3D CNNs. High-resolution octrees can also be generated, progressively, in a depth-first [36] or breadth-first [33] manner.

based reconstruction. The first one is computational since convolutional operations are easier to implement (especially on GPUs) when operating on regular grids. For this purpose, Wang *et al.* [32] designed O-CNN, a novel octree data structure, to efficiently store the octant information and CNN features into the graphics memory and execute the entire training and evaluation on the GPU. O-CNN supports various CNN structures and works with 3D shapes of different representations. By restraining the computations on the octants occupied by 3D surfaces, the memory and computational costs of the O-CNN grow quadratically as the depth of the octree increases, which makes the 3D CNN feasible for high-resolution 3D models.

The second challenge stems from the fact that the octree structure is object-dependent. Thus, ideally, the deep neural network needs to learn how to infer both the structure of the octree and its content. In this section, we will discuss how these challenges have been addressed in the literature.

**4.3.1.1 Using pre-defined octree structures:** The simplest approach is to assume that, at runtime, the structure of the octree is known. This is fine for applications such as semantic segmentation where the structure of the output octree can be set to be identical to that of the input. However, in many important scenarios, *e.g.*, 3D reconstruction, shape modeling, and RGB-D fusion, the structure of the octree is not known in advance and must be predicted. To this end, Riegler *et al.* [41] proposed a hybrid grid-octree structure called OctNet (Fig. 1-(a)). The key idea is to restrict the maximal depth of an octree to a small number, *e.g.*, three, and place several such shallow octrees on a regular grid. This representation enables 3D convolutional networks that are both deep and of high resolution. However, at test time, Riegler *et al.* [41] assume that the structure of the individual octrees is known. Thus, although the method is able to reconstruct 3D volumes at a resolution of  $256^3$ , it lacks flexibility since different types of objects may require different training.

**4.3.1.2 Learning the octree structure :** Ideally, the octree structure and its content should be simultaneously estimated. This can be done as follows;

- First, the input is encoded into a compact feature vector using a convolutional encoder (Section 3).
- Next, the feature vector is decoded using a standard up-convolutional network. This results in a coarse volumetric reconstruction of the input, usually of resolution  $32^3$  (Section 4.2).

- The reconstructed volume, which forms the root of the octree, is subdivided into 8 octants. Octants with boundary voxels are upsampled and further processed, using an up-convolutional network, to refine the reconstruction of the regions in that octant.
- The octants are processed recursively until the desired resolution is reached.

Häne *et al.* [36] introduced the Hierarchical Surface Prediction (HSP), see Fig. 1-(b), which used the approach described above to reconstruct volumetric grids of resolution up to  $256^3$ . In this approach, the octree is explored in depth-first manner. Tatarchenko *et al.* [33], on the other hand, proposed the Octree Generating Networks (OGN), which follows the same idea but the octree is explored in breadth-first manner, see Fig. 1-(c). As such, OGN produces a hierarchical reconstruction of the 3D shape. The approach was able to reconstruct volumetric grids of size  $512^3$ .

Wang *et al.* [34] introduced a patch-guided partitioning strategy. The core idea is to represent a 3D shape with an octree where each of its leaf nodes approximates a planar surface. To infer such structure from a latent representation, Wang *et al.* [34] used a cascade of decoders, one per octree level. At each octree level, a decoder predicts the planar patch within each cell, and a predictor (composed of fully connected layers) predicts the patch approximation status for each octant, *i.e.*, whether the cell is "empty", "surface well approximated" with a plane, and "surface poorly approximated". Cells of poorly approximated surface patches are further subdivided and processed by the next level. This approach reduces the memory requirements from 6.4GB for volumetric grids of size  $256^3$  [32] to 1.7GB, and the computation time from 1.39s to 0.30s, while maintaining the same level of accuracy. Its main limitation is that adjacent patches are not seamlessly reconstructed. Also, since a plane is fitted to each octree cell, it does not approximate well curved surfaces.

#### 4.3.2 Occupancy networks

While it is possible to reduce the memory footprint by using various space partitioning techniques, these approaches lead to complex implementations and existing data-adaptive algorithms are still limited to relatively small voxel grids ( $256^3$  to  $512^2$ ). Recently, several papers proposed to learn implicit representations of 3D shapes using deep neural networks. For instance, Chen and Zhang [43] proposed a decoder that takes the latent representation of a shape and a 3D point, and returns a value indicating

whether the point is outside or inside the shape. The network can be used to reconstruct high resolution 3D volumetric representations. However, when retrieving generated shapes, volumetric CNNs only need one shot to obtain the voxel model, while this method needs to pass every point in the voxel grid to the network to obtain its value. Thus, the time required to generate a sample depends on the sampling resolution.

Tatarchenko *et al.* [44] introduced occupancy networks that implicitly represent the 3D surface of an object as the continuous decision boundary of a deep neural network classifier. Instead of predicting a voxelized representation at a fixed resolution, the approach predicts the complete occupancy function with a neural network that can be evaluated at any arbitrary resolution. This drastically reduces the memory footprint during training. At inference time, a mesh can be extracted from the learned model using a simple multi-resolution isosurface extraction algorithm.

#### 4.3.3 Shape partitioning

Instead of partitioning the volumetric space in which the 3D shapes are embedded, an alternative approach is to consider the shape as an arrangement of geometric parts, reconstruct the individual parts independently from each other, and then stitch the parts together to form the complete 3D shape. There has been a few works which attempted this approach. For instance, Li *et al.* [42] only generate voxel representations at the part level. They proposed a Generative Recursive Autoencoder for Shape Structure (GRASS). The idea is to split the problem into two steps. The first step uses a Recursive Neural Nets (RvNN) encoder-decoder architecture coupled with a Generative Adversarial Network to learn how to best organize a shape structure into a symmetry hierarchy and how to synthesize the part arrangements. The second step learns, using another generative model, how to synthesize the geometry of each part, represented as a voxel grid of size  $32^3$ . Thus, although the part generator network synthesizes the 3D geometry of parts at only  $32^3$  resolution, the fact that individual parts are treated separately enables the reconstruction of 3D shapes at high resolution.

Zou *et al.* [29] reconstruct a 3D object as a collection of primitives using a generative recurrent neural network called 3D-PRNN. The architecture transforms the input into a feature vector of size 32 via an encoder network. Then, a recurrent generator composed of stacks of Long Short-Term Memory (LSTM) and a Mixture Density Network (MDN) sequentially predicts from the feature vector the different parts of the shape. At each time step, the network predicts a set of primitives conditioned on both the feature vector and the previously estimated single primitive. The predicted parts are then combined together to form the reconstruction result. This approach predicts only an abstracted representation in the form of cuboids. Coupling it with volumetric-based reconstruction techniques, which would focus on individual cuboids, could lead to a refined 3D reconstruction at the part level.

#### 4.3.4 Subspace parameterization

The space of all possible shapes can be parameterized using a set of orthogonal basis  $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$ . Every shape

$X$  can then be represented as a linear combination of the bases, *i.e.*,  $X = \sum_{i=1}^n \alpha_i \mathbf{b}_i$ , with  $\alpha_i \in \mathbb{R}$ . This formulation simplifies the reconstruction problem; instead of trying to learn how to reconstruct the volumetric grid  $V$ , one can design a decoder composed of fully connected layers to estimate the coefficients  $\alpha_i, i = 1, \dots, n$  from the latent representation, and then recover the complete 3D volume. Johnston *et al.* [12] used the Discrete Cosine Transform-II (DCT-II) to define  $\mathbf{B}$ . They then proposed a convolutional encoder to predict the low frequency DCT-II coefficients  $\alpha_i$ . These coefficients are then converted by a simple Inverse DCT (IDCT) linear transform, which replaces the decoding network, to a solid 3D volume. This had a profound impact on the computational cost of training and inference: using  $n = 20^3$  DCT coefficients, the network is able to reconstruct surfaces at volumetric grids of size  $128^3$ .

The main issue when using generic bases such as the DCT bases is that, in general, one requires a large number of basis elements to accurately represent complex 3D objects. In practice, we usually deal with objects of known categories, *e.g.*, human faces and 3D human bodies, and usually, training data is available, see Section 8. As such, one can use Principal Component (PCA) bases, learned from the training data, to parameterize the space of shapes [31]. This would require a significantly smaller number of bases (in the order of 10) compared to the number of generic basis, which is in the order of thousands.

#### 4.3.5 Coarse-to-fine refinement

Another way to improve the resolution of volumetric techniques is by using multi-staged approaches [26], [28], [35], [45], [46]. The first stage recovers a low resolution voxel grid, say  $32^3$ , using an encoder-decoder architecture. The subsequent stages, which function as upsampling networks, refine the reconstruction by focusing on local regions. Yang *et al.* [46] used an up-sampling module which simply consists of two up-convolutional layers. This simple up-sampling module upgrades the output 3D shape to a higher resolution of  $256^3$ .

Wang *et al.* [28] treat the reconstructed coarse voxel grid as a sequence of images (or slices). The 3D object is then reconstructed slice by slice at high resolution. While this approach allows efficient refinement using 2D up-convolutions, the 3D shapes used for training should be consistently aligned so that the volumes can be sliced along the first principal direction. Also, reconstructing individual slices independently from each other may result in discontinuities and incoherences in the final volume. To capture the dependencies between the slices, Wang *et al.* [28] use a Long term Recurrent Convolutional Network (LRCN) [47] composed of a 3D encoder, an LSTM unit, and a 2D decoder. At each time, the 3D encoder processes five consecutive slices to produce a fixed-length vector representation as input to the LSTM. The output of the LSTM is passed to the 2D convolutional decoder to produce a high resolution image. The concatenation of the high-resolution 2D images forms the high-resolution output 3D volume.

Instead of using volume slicing, other papers used additional CNN modules, which focus on regions that require refinement. For example, Dai *et al.* [26] firstly predict a coarse but complete shape volume of size  $32^3$  and then refine it

into a  $128^3$  grid via an iterative volumetric patch synthesis process, which copy-pastes voxels from the k-nearest-neighbors retrieved from a database of 3D models. Han *et al.* [45] extend Dai *et al.*'s approach by introducing a local 3D CNN to perform patch-level surface refinement. Cao *et al.* [35], which recover in the first stage a volumetric grid of size  $128^3$ , take volumetric blocks of size  $16^3$  and predict whether they require further refinement. Blocks that require refinement are resampled into  $512^3$  and fed into another encoder-decoder for refinement, along with the initial coarse prediction to guide the refinement. Both subnetworks adopt the U-net architecture [48] while substituting convolution and pooling layers with the corresponding operations from OctNet [41].

Note that these methods need separate and sometimes time-consuming steps before local inference. For example, Dai *et al.* [26] require nearest neighbor searches from a 3D database. Han *et al.* [45] require 3D boundary detection while Cao *et al.* [35] require assessing whether a block requires further refinement or not.

#### 4.4 Deep marching cubes

While volumetric representations can handle 3D shapes of arbitrary topologies, they require a post processing step, *e.g.*, marching cubes [49], to retrieve the actual 3D surface mesh, which is the quantity of interest in 3D reconstruction. As such, the whole pipeline cannot be trained end-to-end. To overcome this limitation, Liao *et al.* [50] introduced the Deep Marching Cubes, an end-to-end trainable network, which predicts explicit surface representations of arbitrary topology. They use a modified differentiable representation, which separates the mesh topology from the geometry. The network is composed of an encoder and a two-branch decoder. Instead of predicting signed distance values, the first branch predicts the probability of occupancy for each voxel. The mesh topology is then implicitly (and probabilistically) defined by the state of the occupancy variables at its corners. The second branch of the decoder predicts a vertex location for every edge of each cell. The combination of both implicitly-defined topology and vertex location defines a distribution over meshes that is differentiable and can be used for back propagation. While the approach is trainable end-to-end, it is limited to low resolution grids of size  $32^3$ .

Instead of directly estimating high resolution volumetric grids, some methods produce multiview depth maps, which are fused into an output volume. The main advantage is that, in the decoding stage, one can use 2D convolutions, which are more efficient, in terms of computation and memory storage, than 3D convolutions. Their main limitation, however, is that depth maps only encode the external surface. To capture internal structures, Richter *et al.* [51] introduced Matryoshka Networks, which use  $L$  nested depth layers; the shape is recursively reconstructed by first fusing the depth maps in the first layer, then subtracting shapes in even layers, and adding shapes in odd layers. The method is able to reconstruct volumetric grids of size  $256^3$ .

### 5 3D SURFACE DECODING

Volumetric representation-based methods are computationally very wasteful since information is rich only on or

TABLE 4: Taxonomy of mesh decoders. GCNN: graph CNN. MLP: Multilayer Perceptron. Param.: parameterization.

	Param.-based	Deformation-based		Decoder architecture
		Defo. model	Template	
[52]	Geometry Images	–	–	UpConv
[53]	Spherical maps	Vertex defo.	Sphere / ellipse	ResNet blocks + 2 Conv layers
[54]	Patch-based	Morphable FFD	(k)-NN	FC layers
[55]	Mesh	vertex defo.	Learned (PCA)	UpConv
[56]	Mesh	vertex defo.	Learned (CNN)	MLP
[57]	Mesh	vertex	cube	FC
[58]	Mesh	FFD	k-NN	GCNN blocks
[59]	Mesh	FFD	NN	Feed-forward
[60]	Mesh	FFD	k-NN	UpConv

near the surfaces of 3D shapes. The main challenge when working directly with surfaces is that common representations such as meshes or point clouds are not regularly structured and thus, they do not easily fit into deep learning architectures, especially those using CNNs. This section reviews the techniques used to address this problem. We classify the state-of-the-art into three main categories: parameterization-based (Section 5.1), template deformation-based (Section 5.2), and point-based methods (Section 5.3).

#### 5.1 Parameterization-based 3D reconstruction

Instead of working directly with triangular meshes, we can represent the surface of a 3D shape  $X$  as a mapping  $\zeta : \mathcal{D} \rightarrow \mathbb{R}^3$  where  $\mathcal{D}$  is a regular parameterization domain. The goal of the 3D reconstruction process is then to recover the shape function  $\zeta$  from an input  $I$ . When  $\mathcal{D}$  is a 3D domain then the methods in this class fall within the volumetric techniques described in Section 4. Here, we focus on the case where  $\mathcal{D}$  is a regular 2D domain, which can be a subset of the two dimensional plane, *e.g.*,  $\mathcal{D} = [0, 1]^2$ , or the unit sphere, *i.e.*,  $\mathcal{D} = S^2$ . In the first case, one can implement encoder-decoder architectures using standard 2D convolution operations. In the latter case, one has to use spherical convolutions [61] since the domain is spherical.

Spherical parameterizations and geometry images [62], [63], [64] are the most commonly used parameterizations. They are, however, suitable only for genus-0 and disk-like surfaces. Surfaces of arbitrary topology need to be cut into disk-like patches, and then unfolded into a regular 2D domain. Finding the optimal cut for a given surface, and more importantly, finding cuts that are consistent across shapes within the same category is challenging. In fact, naively creating independent geometry images for a shape category and feeding them into deep neural networks would fail to generate coherent 3D shape surfaces [52].

To create, for genus-0 surfaces, robust geometry images that are consistent across a shape category, the 3D objects within the category should be first put in correspondence [65], [66], [67]. Sinha *et al.* [52] proposed a cut-invariant procedure, which solves a large-scale correspondence problem, and an extension of deep residual nets to automatically generate geometry images encoding the  $x, y, z$  surface coordinates. The approach uses three separate encoder-decoder networks, which learn, respectively, the

$x$ ,  $y$  and  $z$  geometry images. The three networks are composed of standard convolutions, up-residual, and down-residual blocks. They take as input a depth image or a RGB image, and learn the 3D reconstruction by minimizing a shape-aware  $L_2$  loss function.

Pumarola *et al.* [53] reconstruct the shape of a deformable surface using a network which has two branches: a detection branch and a depth estimation branch, which operate in parallel, and a third shape branch, which merges the detection mask and the depth map into a parameterized surface. Groueix *et al.* [54] decompose the surface of a 3D object into  $m$  patches, each patch  $i$  is defined as a mapping  $\zeta_i : \mathcal{D} = [0, 1]^2 \mapsto \mathbb{R}^3$ . They have then designed a decoder which is composed of  $m$  branches. Each branch  $i$  reconstructs the  $i$ -th patch by estimating the function  $\zeta_i$ . At the end, the reconstructed patches are merged together to form the entire surface. Although this approach can handle surfaces of high genus, it is still not general enough to handle surfaces of arbitrary genus. In fact, the optimal number of patches depends on the genus of the surface ( $n = 1$  for genus-0,  $n = 2$  for genus-1, etc.). Also, the patches are not guaranteed to be connected, although in practice one can still post-process the result and fill in the gaps between disconnected patches.

In summary, parameterization methods are limited to low-genus surfaces. As such, they are suitable for the reconstruction of objects that belong to a given shape category, e.g., human faces and bodies.

## 5.2 Deformation-based 3D reconstruction

Methods in this class take an input  $I$  and estimate a deformation field  $\Delta$ , which, when applied to a template 3D shape, results in the reconstructed 3D model  $X$ . Existing techniques differ in the type of deformation models they use (Section 5.2.1), the way the template is defined (Section 5.2.2), and in the network architecture used to estimate the deformation field  $\Delta$  (Section 5.2.3). In what follows, we assume that a 3D shape  $X = (\mathcal{V}, \mathcal{F})$  is represented with  $n$  vertices  $\mathcal{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$  and faces  $\mathcal{F}$ . Let  $\tilde{X} = (\tilde{\mathcal{V}}, \mathcal{F})$  denote a template shape.

### 5.2.1 Deformation models

(1) *Vertex deformation*. This model assumes that a 3D shape  $X$  can be written in terms of linear displacements of the individual vertices of the template, *i.e.*,  $\forall \mathbf{v}_i \in \mathcal{V}, \mathbf{v}_i = \tilde{\mathbf{v}}_i + \delta_i$ , where  $\delta_i \in \mathbb{R}^3$ . The deformation field is defined as  $\Delta = (\delta_1, \dots, \delta_n)$ . This deformation model, illustrated in Fig. 2-(top), has been used in) [55], [56], [57]. It assumes that (1) there is a one-to-one correspondence between the vertices of the shape  $X$  and those of the template  $\tilde{X}$ , and (2) the shape  $X$  has the same topology as the template  $\tilde{X}$ .

(2) *Morphable models*. Instead of using a generic template, one can use learned morphable models [68] to parameterize a 3D mesh. Let  $\tilde{\mathcal{V}}$  be the mean shape and  $\Lambda_1, \dots, \Lambda_K$  be a set of orthonormal basis. Any shape  $\mathcal{V}$  can be written in the form:

$$\mathcal{V} = \tilde{\mathcal{V}} + \sum_{i=1}^K \alpha_i \Lambda_i, \quad \alpha_i \in \mathbb{R}. \quad (1)$$

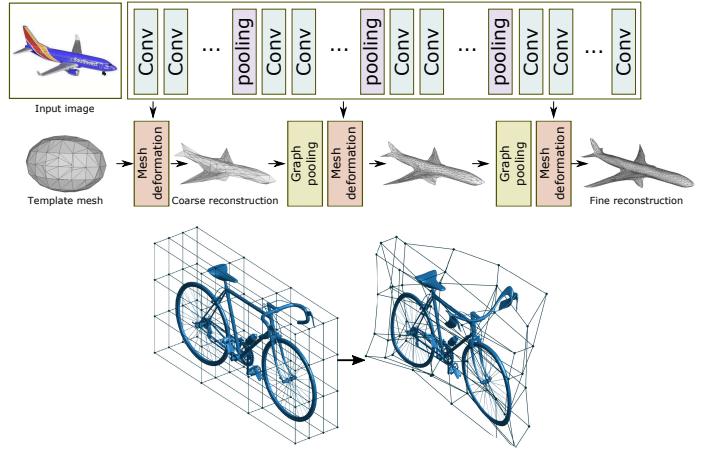


Fig. 2: Template deformation (top) [56] vs. domain deformation (bottom) [60].

The second term of Equation (1) can be seen as a deformation field,  $\Delta = \sum_{i=1}^K \alpha_i \Lambda_i$ , applied to the vertices  $\tilde{\mathcal{V}}$  of the mean shape. By setting  $\Lambda_0 = \tilde{\mathcal{V}}$  and  $\alpha_0 = 1$ , Equation (1) can be written as  $\mathcal{V} = \sum_{i=0}^K \alpha_i \Lambda_i$ . In this case, the mean  $\tilde{\mathcal{V}}$  is treated as a bias term.

One approach to learning a morphable model is by using Principal Component Analysis (PCA) on a collection of clean 3D mesh exemplars [68]. Recent techniques showed that, with only 2D annotations, it is possible to build category-specific 3D morphable models from 2D silhouettes or 2D images [69], [70]. These methods require efficient detection and segmentation of the objects, and camera pose estimation, which can also be done using CNN-based techniques.

(3) *Free-Form Deformation (FFD)*. Instead of directly deforming the vertices of the template  $\tilde{X}$ , one can deform the space around it, see Fig. 2-(bottom). This can be done by defining around  $\tilde{X}$  a set  $P \in \mathbb{R}^{m \times 3}$  of  $m$  control points, called deformation handles. When the deformation field  $\Delta = (\delta_1, \dots, \delta_m)$ ,  $m \ll n$ , is applied to these control points, they deform the entire space around the shape and thus, they also deform the vertices  $\mathcal{V}$  of the shape according to the following equation:

$$\mathcal{V}^\top = B \Phi(P + \Delta)^\top, \quad (2)$$

where the deformation matrix  $B \in \mathbb{R}^{n \times m}$  is a set of polynomial basis, *e.g.*, the Bernstein polynomials [60].  $\Phi$  is a  $m \times m$  matrix used to impose symmetry in the FFD field, see [71], and  $\Delta$  is the displacements.

This approach has been used by Kuryenkov *et al.* [59], Pontes *et al.* [60], and Jack *et al.* [58]. The main advantage of free-form deformation is that it does not require one-to-one correspondence between the shapes and the template. However, the shapes that can be approximated by the FFD of the template are only those that have the same topology as the template.

### 5.2.2 Defining the template

Kato *et al.* [55] used a sphere as a template. Wang *et al.* [56] used an ellipse. Henderson *et al.* [20] defined two types

of templates: a complex shape abstracted into cuboidal primitives, and a cube subdivided into multiple vertices. While the former is suitable for man-made shapes that have multiple components, the latter is suitable for representing genus-0 shapes and does not offer advantage compared to using a sphere or an ellipsoid.

To speed up the convergence, Kuryenkov *et al.* [59] introduced DeformNet, which takes an image as input, searches the nearest shape from a database, and then deforms, using the FFD model of Equation (2), the retrieved model to match the query image. This method allows detail-preserving 3D reconstruction.

Pontes *et al.* [60] used an approach that is similar to DeformNet [59]. However, once the FFD field is estimated and applied to the template, the result is further refined by adding a residual defined as a weighted sum of some 3D models retrieved from a dictionary. The role of the deep neural network is to learn how to estimate the deformation field  $\Delta$  and the weights used in computing the refinement residual. Jack *et al.* [58], on the other hand, deform, using FFD, multiple templates and select the one that provides the best fitting accuracy.

Another approach is to learn the template, either separately using statistical shape analysis techniques, *e.g.*, PCA, on a set of training data, or jointly with the deformation field using deep learning techniques. For instance, Tulsiani *et al.* [70] use the mean shape of each category of 3D models as a class-specific template. The deep neural network estimates both the class of the input shape, which is used to select the class-specific mean shape, and the deformation field that needs to be applied to the class-specific mean shape. Kanazawa *et al.* [57] learn, at the same time, the mean shape and the deformation field. Thus, the approach does not require a separate 3D training set to learn the morphable model. In both cases, the reconstruction results lack details and are limited to popular categories such as cars and birds.

### 5.2.3 Network architectures

Deformation-based methods also use encoder-decoder architectures. The encoder maps the input into a latent variable  $\mathbf{x}$  using successive convolutional operations. The latent space can be discrete or continuous as in [20], which used a variational auto-encoder (see Section 3). The decoder is, in general, composed of fully-connected layers. Kato *et al.* [55], for example, used two fully connected layers to estimate the deformation field to apply to a sphere to match the input's silhouette.

Instead of deforming a sphere or an ellipse, Kuryenkov *et al.* [59] retrieve from a database the 3D model that is most similar to the input  $\mathbf{I}$  and then estimate the FFD needed to deform it to match the input. The retrieved template is first voxelized and encoded, using a 3D CNN, into another latent variable  $\mathbf{x}_t$ . The latent representation of the input image and the latent representation of the retrieved template are then concatenated and decoded, using an up-convolutional network, into an FFD field defined on the vertices of a voxel grid.

Pontes *et al.* [60] used a similar approach, but the latent variable  $\mathbf{x}$  is used as input into a classifier which finds, from a database, the closest model to the input. At the

same time, the latent variable is decoded, using a feed-forward network, into a deformation field  $\Delta$  and weights  $\alpha_i, i = 1, \dots, K$ . The retrieved template is then deformed using  $\Delta$  and a weighted combination of a dictionary of CAD models, using the weights  $\alpha_i$ .

Note that, one can design several variants to these approaches. For instance, instead of using a 3D model retrieved from a database as a template, one can use a class-specific mean shape. In this case, the latent variable  $\mathbf{x}$  can be used to classify the input into one of the shape categories, and then pick the learned mean shape of this category as a template [70]. Also, instead of learning separately the mean shape, *e.g.*, using morphable models, Kanazawa *et al.* [57] treated the mean shape as a bias term, which can then be predicted by the network, along with the deformation field  $\Delta$ . Finally, Wang *et al.* [56] adopted a coarse to fine strategy, which makes the procedure more stable. They proposed a deformation network composed of three deformation blocks, each block is a graph-based CNN (GCNN), intersected by two graph unpooling layers. The deformation blocks update the location of the vertices while the graph unpooling layers increase the number of vertices.

Parameterization and deformation-based techniques can only reconstruct surfaces of fixed topology. The former is limited to surfaces of low genus while the latter is limited to the topology of the template.

## 5.3 Point-based techniques

A 3D shape can be represented using an unordered set  $S = \{(x_i, y_i, z_i)\}_{i=1}^N$  of  $N$  points. Such point-based representation is simple but efficient in terms of memory requirements. It is well suited for objects with intriguing parts and fine details. As such, an increasing number of papers, at least one in 2017 [72], more than 12 in 2018 [21], [21], [22], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], and a few others in 2019 [81], explored their usage for deep learning-based reconstruction. This section discusses the state-of-the-art point-based representations and their corresponding network architectures.

### 5.3.1 Representations

The main challenge with point clouds is that they are not regular structures and do not easily fit into the convolutional architectures that exploit the spatial regularity. Three representations have been proposed to overcome this limitation:

- Point set representation treats a point cloud as a matrix of size  $N \times 3$  [21], [22], [72], [75], [77], [81].
- One or multiple 3-channel grids of size  $H \times W \times 3$  [72], [73], [82]. Each pixel in a grid encodes the  $(x, y, z)$  coordinates of a 3D point.
- Depth maps from multiple viewpoints [78], [83].

The last two representations, hereinafter referred to as grid representations, are well suited for convolutional networks. They are also computationally efficient as they can be inferred using only 2D convolutions. Note that depth map-based methods require an additional fusion step to infer the entire 3D shape of an object. This can be done in a straightforward manner if the camera parameters are known. Otherwise, the fusion can be done using point cloud

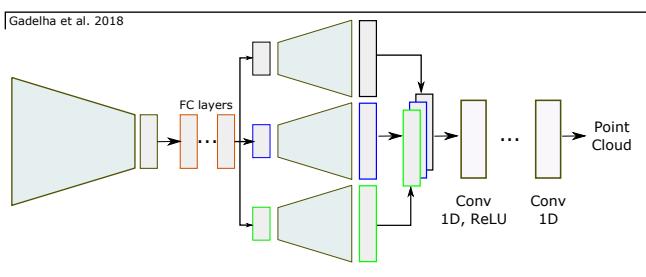
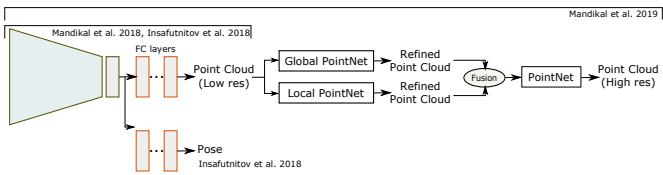
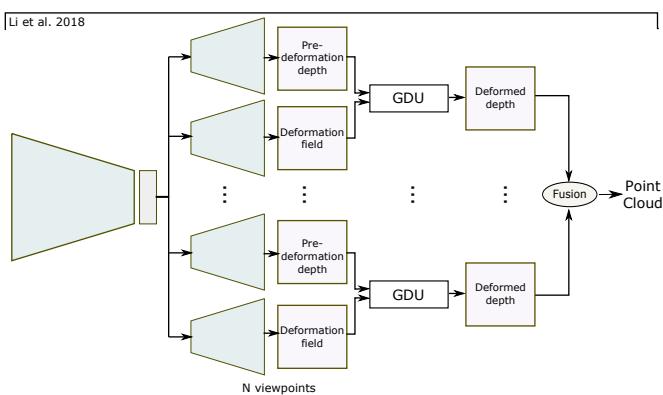
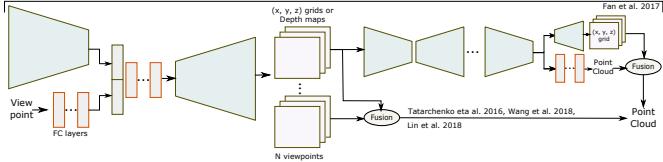


Fig. 3: The different network architectures used in point-based 3D reconstruction.

registration techniques [84], [85] or fusion networks [86]. Also, point set representations require fixing in advance the number of points  $N$  while in methods that use grid representations, the number of points can vary based on the nature of the object but it is always bounded by the grid resolution.

### 5.3.2 Network architectures

Similar to volumetric and surface-based representations, techniques that use point-based representations follow the encoder-decoder model. While they all use the same architecture for the encoder, they differ in the type and architecture of their decoder, see Fig. 3. In general, grid representations use up-convolutional networks to decode the latent variable [72], [73], [78], [82], see Fig. 3-(a) and (b). Point set representations (Fig. 3-(c)) use fully connected layers [21], [72], [74], [77], [81] since point clouds are unordered.

The main advantage of fully-connected layers is that they capture the global information. However, compared to convolutional operations, they are computationally expensive. To benefit from the efficiency of convolutional operations, Gadelha *et al.* [22] order, spatially, the point cloud using a space-partitioning tree such as KD-tree and then process them using 1D convolutional operations, see Fig. 3-(d). With a conventional CNN, each convolutional operation has a restricted receptive field and is not able to leverage both global and local information effectively. Gadelha *et al.* [22] resolve this issue by maintaining three different resolutions. That is, the latent variable is decoded into three different resolutions, which are then concatenated and further processed with 1D convolutional layers to generate a point cloud of size  $4K$ .

Fan *et al.* [72] proposed a generative deep network that combines both the point set representation and the grid representation (Fig. 3-(a)). The network is composed of a cascade of encoder-decoder blocks:

- The first block takes the input image and maps it into a latent representation, which is then decoded into a 3-channel image of size  $H \times W$ . The three values at each pixel are the coordinates of a point.
- Each of the subsequent blocks takes the output of its previous block and further encodes and decodes it into a 3-channel image of size  $H \times W$ .
- The last block is an encoder, of the same type as the previous ones, followed by a predictor composed of two branches. The first branch is a decoder which predicts a 3-channel image of size  $H \times W$  ( $32 \times 24$  in this case), of which the three values at each pixel are the coordinates of a point. The second branch is a fully-connected network, which predicts a matrix of size  $N \times 3$ , each row is a 3D point ( $N = 256$ ).
- The predictions of the two branches are merged using set union to produce a 3D point set of size 1024.

This approach has been also used by Jiang *et al.* [74]. The main difference between the two is in the training procedure, which we will discuss in Section 7.

Tatarchenko *et al.* [83], Wang *et al.* [82], and Lin *et al.* [73] followed the same idea but their decoder regresses  $N$  grids, see Fig. 3-(a). Each grid encodes the depth map [83] or the  $(x, y, z)$  coordinates [73], [82] of the visible surface from that view point. The viewpoint, encoded with a sequence of fully connected layers, is provided as input to the decoder along with the latent representation of the input image. Li *et al.* [78], on the other hand, used a multi-branch decoder, one for each viewpoint, see Fig. 3-(b). Unlike [83], each branch regresses a canonical depth map from a given view point and a deformation field, which deforms the estimated canonical depth map to match the input, using Grid Deformation Units (GDUs). The reconstructed grids are then lifted to 3D and merged together.

Similar to volumetric techniques, the vanilla architecture for point-based 3D reconstruction only recovers low resolution geometry. For high-resolution reconstruction, Mandikal *et al.* [81], see Fig. 3-(c), use a cascade of multiple networks. The first network predicts a low resolution point cloud. Each subsequent block takes the previously predicted

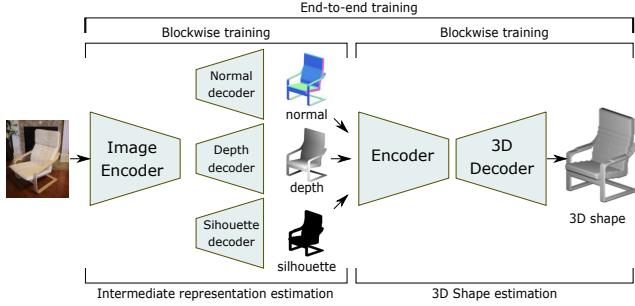


Fig. 4: Intermediating via 2.5D sketches (depth, normals, and silhouettes).

point cloud, computes global features, using a multi-layer perceptron architecture (MLP) similar to PointNet [87] or Pointnet++ [88], and local features by applying MLPs in balls around each point. Local and global features are then aggregated and fed to another MLP, which predicts a dense point cloud. The process can be repeated recursively until the desired resolution is reached.

Mandikal *et al.* [21] combine TL-embedding with a variational auto-encoder (Fig. 3-(c)). The former allows mapping a 3D point cloud and its corresponding views onto the same location in the latent space. The latter enables the reconstruction of multiple plausible point clouds from the input image(s).

Finally, point-based representations can handle 3D shapes of arbitrary topologies. However, they require a post processing step, *e.g.*, Poisson surface reconstruction [89] or SSD [90], to retrieve the 3D surface mesh, which is the quantity of interest. The pipeline, from the input until the final mesh is obtained, cannot be trained end-to-end. Thus, these methods only optimise an auxiliary loss defined on an intermediate representation.

## 6 LEVERAGING OTHER CUES

The previous sections discussed methods that directly reconstruct 3D objects from their 2D observations. This section shows how additional cues such as intermediate representations (Section 6.1) and temporal correlations (Section 6.2) can be used to boost 3D reconstruction.

### 6.1 Intermediating

Many of the deep learning-based 3D reconstruction algorithms directly predict the 3D geometry of an object from RGB images. Some techniques, however, decompose the problem into sequential steps, which estimate 2.5D information such as depth maps, normal maps, and/or segmentation masks, see Fig. 4. The last step, which can be implemented using traditional techniques such as space carving or 3D back-projection followed by filtering and registration, recovers the full 3D geometry and the pose of the input.

While early methods train separately the different modules, recent works proposed end-to-end solutions [6], [9], [38], [53], [80], [91], [92]. For instance, Wu *et al.* [6] and later Sun *et al.* [9] used two blocks. The first block is an encoder followed by a three-branch decoder, which estimates the

depth map, the normal map, and the segmentation mask (called 2.5D sketches). These are then concatenated and fed into another encoder-decoder, which regresses a full 3D volumetric grid [6], [9], [91], and a set of fully-connected layers, which regress the camera pose [9]. The entire network is trained end-to-end.

Other techniques convert the intermediate depth map into (1) a 3D occupancy grid [46] or a truncated signed distance function volume [38], which is then processed using a 3D encoder-decoder network for completion and refinement, or (2) a partial point cloud, which is further processed using a point-cloud completion module [80]. Zhang *et al.* [92] convert the inferred depth map into a spherical map and unpaint it, to fill in holes, using another encoder-decoder. The unpainted spherical depth map is then back-projected to 3D and refined using a voxel refinement network, which estimates a voxel occupancy grid of size  $128^3$ .

Other techniques estimate multiple depth maps from pre-defined or arbitrary viewpoints. Tatarchenko *et al.* [83] proposed a network, which takes as input an RGB image and a target viewpoint  $v$ , and infers the depth map of the object as seen from the viewpoint  $v$ . By varying the viewpoint, the network is able to estimate multiple depths, which can then be merged into a complete 3D model. The approach uses a standard encoder-decoder and an additional network composed of three fully-connected layers to encode the viewpoint. Soltani *et al.* [19] and Lin *et al.* [73] followed the same approach but predict the depth maps, along with their binary masks, from pre-defined view points. In both methods, the merging is performed in a post-processing step. Smith *et al.* [93] first estimate a low resolution voxel grid. They then take the depth maps, of the low resolution voxel grid, computed from the six axis-aligned views and refine them using a silhouette and depth refinement network. The refined depth maps are finally combined into a volumetric grid of size  $256^3$  using space carving techniques.

Tatarchenko *et al.* [83], Lin *et al.* [73], and Sun *et al.* [9] also estimate the binary/silhouette masks, along with the depth maps. The binary masks have been used to filter out points that are not back-projected to the surface in 3D space. The side effect of these depth mask-based approaches is that it is a huge computation waste as a large number of points are discarded, especially for objects with thin structures. Li *et al.* [78] overcome this problem by deforming a regular depth map using a learned deformation field. Instead of directly inferring depth maps that best fit the input, Li *et al.* [78] infer a set of 2D pre-deformation depth maps and their corresponding deformation fields at pre-defined canonical viewpoints. These are each passed to a Grid Deformation Unit (GDU) that transforms the regular grid of the depth map to a *deformed depth map*. Finally, the deformed depth maps are transformed into a common coordinate frame for fusion into a dense point cloud.

The main advantage of multi-staged approaches is that depth, normal, and silhouette maps are much easier to recover from 2D images. Likewise, 3D models are much easier to recover from these three modalities than from 2D images alone.

## 6.2 Exploiting spatio-temporal correlations

There are many situations where multiple spatially distributed images of the same object(s) are acquired over an extended period of time. Single image-based reconstruction techniques can be used to reconstruct the 3D shapes by processing individual frames independently from each other, and then merging the reconstruction using registration techniques. Ideally, we would like to leverage on the spatio-temporal correlations that exist between the frames to resolve ambiguities especially in the presence of occlusions and highly cluttered scenes. In particular, the network at time  $t$  should remember what has been reconstructed up to time  $t - 1$ , and use it, in addition to the new input, to reconstruct the scene or objects at time  $t$ . This problem of processing sequential data has been addressed by using Recurrent Neural Networks (RNN) and Long-Short Term Memory (LSTM) networks, which enable networks to remember their inputs over a period of time.

Choy *et al.* [7] proposed an architecture called 3D Recurrent Reconstruction Network (3D-R2N2), which allows the network to adaptively and consistently learn a suitable 3D representation of an object as (potentially conflicting) information from different viewpoints becomes available. The network can perform incremental refinement every time a new view becomes available. It is composed of two parts; a standard convolution encoder-decoder and a set of 3D Convolutional Long-Short Term Memory (3D-LSTM) units placed at the start of the convolutional decoder. These take the output of the encoder, and then either selectively update their cell states or retain the states by closing the input gate. The decoder then decodes the hidden states of the LSTM units and generates a probabilistic reconstruction in the form of a voxel occupancy map.

The 3D-LSTM allows the network to retain what it has seen and update its memory when it sees a new image. It is able to effectively handle object self-occlusions when multiple views are fed to the network. At each time step, it selectively updates the memory cells that correspond to parts that became visible while retaining the states of the other parts.

LSTM and RNNs are time consuming since the input images are processed sequentially without parallelization. Also, when given the same set of images with different orders, RNNs are unable to estimate the 3D shape of an object consistently due to permutation variance. To overcome these limitations, Xie *et al.* [86] introduced Pix2Vox, which is composed of multiple encoder-decoder blocks, running in parallel, each one predicts a coarse volumetric grid from its input frame. This eliminates the effect of the order of input images and accelerates the computation. Then, a context-aware fusion module selects high-quality reconstructions from the coarse 3D volumes and generates a fused 3D volume, which fully exploits information of all input images without long-term memory loss.

## 7 TRAINING

In addition to their architectures, the performance of deep learning networks depends on the way they are trained. This section discusses the various supervisory modes (Sec-

tion 7.1) and training procedures that have been used in the literature (Section 7.3).

### 7.1 Degree of supervision

Early methods rely on 3D supervision (Section 7.1.1). However, obtaining ground-truth 3D data, either manually or using traditional 3D reconstruction techniques, is extremely difficult and expensive. As such, recent techniques try to minimize the amount of 3D supervision by exploiting other supervisory signals such consistency across views (Section 7.1.2).

#### 7.1.1 Training with 3D supervision

Supervised methods require training using images paired with their corresponding ground-truth 3D shapes. The training process then minimizes a loss function that measures the discrepancy between the reconstructed 3D shape and the corresponding ground-truth 3D model. The discrepancy is measured using loss functions, which are required to be differentiable so that gradients can be computed. Examples of such functions include:

**(1) Volumetric loss.** It is defined as the distance between the reconstructed and the ground-truth volumes;

$$\mathcal{L}_{vol}(\mathbf{I}) = d(f(\mathbf{I}), X). \quad (3)$$

Here,  $d(\cdot, \cdot)$  can be the  $L_2$  distance between the two volumes or the negative Intersection over Union (IoU)  $\mathcal{L}_{IoU}$  (see Equation (16)). Both metrics are suitable for binary occupancy grids and TSDF representations. For probabilistic occupancy grids, the cross-entropy loss is the most commonly used [25]:

$$\mathcal{L}_{CE} = -\frac{1}{N} \sum_{i=1}^N \{p_i \log \hat{p}_i + (1 - p_i) \log(1 - \hat{p}_i)\}. \quad (4)$$

Here,  $p_i$  is the ground-truth probability of voxel  $i$  being occupied,  $\hat{p}_i$  is the estimated probability, and  $N$  is the number of voxels.

**(2) Point set loss.** When using point-based representations, the reconstruction loss can be measured using the Earth Mover's Distance (EMD) [59], [72] or the Chamfer Distance (CD) [59], [72]. The EMD is defined as the minimum of the sum of distances between a point in one set and a point in another set over all possible permutations of the correspondences. More formally, given two sets of points  $S_{gt}$  and  $S_{rec}$ , the EMD is defined as:

$$\mathcal{L}_{EMD} = \min_{S_{gt} \rightarrow S_{rec}} \sum_{p \in S_{gt}} \|p - \phi(p)\|. \quad (5)$$

Here,  $\phi(p) \in S_{rec}$  is the closest point on  $S_{rec}$  to  $p \in S_{gt}$ . The CD loss, on the other hand, is defined as:

$$\mathcal{L}_{CD} = \frac{1}{N_{gt}} \min_{p \in S_{gt}} \|p - q\|^2 + \frac{1}{N_{rec}} \min_{q \in S_{rec}} \|p - q\|^2. \quad (6)$$

$N_{gt}$  and  $N_{rec}$  are, respectively, the size of  $S_{gt}$  and  $S_{rec}$ . The CD is computationally easier than EMD since it uses sub-optimal matching to determine the pairwise relations.

**(3) Learning to generate multiple plausible reconstructions.** 3D reconstruction from a single image is an ill-posed

problem, thus for a given input there might be multiple plausible reconstructions. Fan *et al.* [72] proposed the Min-of-N (MoN) loss to train neural networks to generate distributional output. The idea is to use a random vector  $r$  drawn from a certain distribution to perturb the input. The network learns to generate a plausible 3D shape from each perturbation of the input. It is trained using a loss defined as follows;

$$\mathcal{L}_{MoN} = \sum_i \min_{r \sim \mathbb{N}(0, I)} \{d(f(\mathbf{I}, r), S_{gt})\}. \quad (7)$$

Here,  $f(\mathbf{I}, r)$  is the reconstructed 3D point cloud after perturbing the input with the random vector  $r$  sampled from the multivariate normal distribution  $\mathbb{N}(0, I)$ ,  $S_{gt}$  is the ground-truth point cloud, and  $d(\cdot, \cdot)$  is a reconstruction loss, which can be any of the loss functions defined above. At runtime, various plausible reconstructions can be generated from a given input by sampling different random vectors  $r$  from  $\mathbb{N}(0, I)$ .

### 7.1.2 Training with 2D supervision

Obtaining 3D ground-truth data for supervision is an expensive and tedious process even for a small scale training. However, obtaining multiview 2D or 2.5D images for training is relatively easy. Methods in the category use the fact that if the estimated 3D shape is as close as possible to the ground truth then the discrepancy between views of the 3D model and the projection of the reconstructed 3D model onto any of these views is also minimized. Implementing this idea requires defining a projection operator, which renders the reconstructed 3D model from a given viewpoint (Section 7.1.2.1), and a loss function that measures the reprojection error (Section 7.1.2.2).

**7.1.2.1 Projection operators:** Techniques from projective geometry can be used to render views of a 3D object. However, to enable end-to-end training without gradient approximation [55], the projection operator should be differentiable. Gadelha *et al.* [27] introduced a differentiable projection operator  $P$  defined as  $P((i, j), V) = 1 - e^{-\sum_k V(i, j, k)}$ , where  $V$  is the 3D voxel grid. This operator sums up the voxel occupancy values along each line of sight. However, it assumes an orthographic projection. Loper and Black [94] introduced OpenDR, an approximate differentiable renderer, which is suitable for orthographic and perspective projections.

Petersen *et al.* [95] introduced a novel  $C^\infty$  smooth differentiable renderer for image-to-geometry reconstruction. The idea is that instead of taking a discrete decision of which triangle is the visible from a pixel, the approach softly blends their visibility. Taking the weighted SoftMin of the  $z$ -positions in the camera space constitutes a smooth z-buffer, which leads to a  $C^\infty$  smooth renderer, where the  $z$ -positions of triangles are differentiable with respect to occlusions. In previous renderers, only the  $xy$ -coordinates were locally differentiable with respect to occlusions.

Finally, instead of using fixed renderers, Rezende *et al.* [96] proposed a learned projection operator, or a learnable camera, which is built by first applying an affine transformation to the reconstructed volume, followed by a combination of 3D and 2D convolutional layers, which map the 3D volume onto a 2D image.

**7.1.2.2 Re-projection loss functions:** There are several loss functions that have been proposed for 3D reconstruction using 2D supervision. We classify them into two main categories; (1) silhouette-based and (2) normal and depth-based loss functions.

**(1) Silhouette-based loss functions.** The idea is that a 2D silhouette projected from the reconstructed volume, under certain camera intrinsic and extrinsic parameters, should match the ground truth 2D silhouette of the input image. The discrepancy, which is inspired by space carving, is then:

$$\mathcal{L}_{proj}(\mathbf{I}) = \frac{1}{n} \sum_{j=1}^n d(P(f(\mathbf{I}); \alpha^{(j)}), S^{(j)}), \quad (8)$$

where  $S^{(j)}$  is the  $j$ -th ground truth 2D silhouette of the original 3D object  $X$ ,  $n$  is the number of silhouettes or views used for each 3D model,  $P(\cdot)$  is a 3D to 2D projection function, and  $\alpha^{(j)}$  are the camera parameters of the  $j$ -th silhouette. The distance metric  $d(\cdot, \cdot)$  can be the standard  $L_2$  metric [77], the negative Intersection over Union (IoU) between the true and reconstructed silhouettes [55], or the binary cross-entropy loss [4], [24].

Kundu *et al.* [31] introduced the render-and-compare loss, which is defined in terms of the IoU between the ground-truth silhouette  $G_s$  and the rendered silhouette  $R_s$ , and the  $L_2$  distance between the ground-truth depth  $G_d$  and the rendered depth  $R_d$ , i.e.,

$$\mathcal{L}_r = 1 - IoU(R_s, G_s; I_s) + d_{L_2}(R_d, G_d; I_d). \quad (9)$$

Here,  $I_s$  and  $I_d$  are binary ignore masks that have value of one at pixels which do not contribute to the loss. Since this loss is not differentiable, Kundu *et al.* [31] used finite difference to approximate its gradients.

Silhouette-based loss functions cannot distinguish between some views, *e.g.*, front and back. To alleviate this issue, Insafutdinov and Dosovitskiy [77] use multiple pose regressors during training, each one using silhouette loss. The overall network is trained with the min of the individual losses. The predictor with minimum loss is used at test time.

Gwak *et al.* [97] minimize the reprojection error subject to the reconstructed shape being a valid member of a certain class, *e.g.*, chairs. To constrain the reconstruction to remain in the manifold of the shape class, the approach defines a barrier function  $\phi$ , which is set to be 1 if the shape is in the manifold and 0 otherwise. The loss function is then:

$$\mathcal{L} = \mathcal{L}_{reprojection} - \frac{1}{t} \log \phi(\hat{X}). \quad (10)$$

The barrier function is learned as the discriminative function of a GAN, see Section 7.3.2.

Finally, Tulsiani *et al.* [8] define the re-projection loss using a differentiable ray consistency loss for volumetric reconstruction. First, it assumes that the estimated shape  $\hat{X}$  is defined in terms of the probability occupancy grid. Let  $(O, C)$  be an observation-camera pair. Let also  $\mathcal{R}$  be a set of rays where each ray  $r \in \mathcal{R}$  has the camera center as origin and is casted through the image plane of the camera  $C$ . The ray consistency loss is then defined as:

$$\mathcal{L}_{ray\_cons}(\hat{X}; (O, C)) = \sum_{r \in \mathcal{R}} \mathcal{L}_r(\hat{X}), \quad (11)$$

where  $\mathcal{L}_r(\hat{X})$  captures if the inferred 3D model  $\hat{X}$  correctly explains the observations associated with the specific ray  $r$ . If the observation  $O$  is a ground-truth foreground mask taking values 0 at foreground pixels and 1 elsewhere, then  $\mathcal{L}_r$  is the probability that the ray  $r$  hits a surface voxel weighted by the mask value at the pixel associated with the ray  $r$ . This loss is differentiable with respect to the network predictions. Note that when using foreground masks as observations, this loss, which requires known camera parameters, is similar to the approaches designed to specifically use mask supervision where a learned [25] or a fixed [4] reprojection function is used. Also, the binary cross-entropy loss used in [4], [24] can be thought of as an approximation of the one derived using ray consistency.

**(2) Surface normal and depth-based loss.** Additional cues such as surface normals and depth values can be used to guide the training process. Let  $n_{x,y} = (n_a, n_b, n_c)$  be a normal vector to a surface at a point  $(x, y, z)$ . The vectors  $n_x = (0, -n_c, n_b)$  and  $(-n_c, 0, n_a)$  are orthogonal to  $n_{x,y}$ . By normalizing them, we obtain two vectors  $n'_x = (0, -1, n_b/n_c)$  and  $n'_y = (-1, 0, n_a/n_c)$ . The normal loss tries to guarantee that the voxels at  $(x, y, z) \pm n'_x$  and  $(x, y, z) \pm n'_y$  should be 1 to match the estimated surface normals. This constraint only applies when the target voxels are inside the estimated silhouette. The projected surface normal loss is then:

$$\mathcal{L}_{normal} = \left(1 - v_{x,y-1,z+\frac{n_b}{n_c}}\right)^2 + \left(1 - v_{x,y+1,z-\frac{n_b}{n_c}}\right)^2 + \left(1 - v_{x-1,y,z+\frac{n_a}{n_c}}\right)^2 + \left(1 - v_{x+1,y,z-\frac{n_a}{n_c}}\right)^2 \quad (12)$$

This loss has been used by Wu *et al.* [6], which includes, in addition to the normal loss, the projected depth loss. The idea is that the voxel with depth  $v_{x,y,d_{x,y}}$  should be 1, and all voxels in front of it should be 0. The depth loss is then defined as:

$$\mathcal{L}_{depth}(x, y, z) = \begin{cases} v_{x,y,z}^2 & \text{if } z < d_{x,y}, \\ (1 - v_{x,y,z})^2 & \text{if } z = d_{x,y}, \\ 0 & \text{otherwise.} \end{cases} \quad (13)$$

This ensures the estimated 3D shape matches the estimated depth values.

**(3) Combining multiple losses.** One can also combine 2D and 3D losses. This is particularly useful when some ground-truth 3D data is available. One can for example train first the network using 3D supervision, and then fine-tune it using 2D supervision. Yan *et al.* [4], on the other hand, take the weighted sum of a 2D and a 3D loss.

In addition to the reconstruction loss, one can impose additional constraints to the solution. For instance, Kato *et al.* [55] used a weighted sum of silhouette loss, defined as the negative intersection over union (IoU) between the true and reconstructed silhouettes, and a smoothness loss. For surfaces, the smoothness loss ensures that the angles between adjacent faces is close to  $180^\circ$ , encouraging flatness.

**7.1.2.3 Camera parameters and viewpoint estimation:** Reprojection-based loss functions use the camera parameters to render the estimated 3D shape onto image planes. Some methods assume the availability of one or multiple observation-camera pairs [4], [8], [10]. Here, the

observation can be an RGB image, a silhouette/foreground mask or a depth map of the target 3D shape. Other methods optimize at the same time for the camera parameters and the 3D reconstruction that best describe the input [27], [77].

Gadelha *et al.* [27] encode an input image into a latent representation and a pose code using fully-connected layers. The pose code is then used as input to the 2D projection module, which renders the estimated 3D volume onto the view of the input. Insafutdinov and Dosovitskiy [77], on the other hand, take two views of the same object, and predict the corresponding shape (represented as a point cloud) from the first view, and the camera pose (represented as a quaternion) from the second one. The approach then uses a differentiable projection module to generate the view of the predicted shape from the predicted camera pose. The shape and pose predictor is implemented as a convolutional network with two branches. The network starts with a convolutional encoder with a total of 7 layers followed by 2 shared fully connected layers, after which the network splits into two branches for shape and pose prediction. The pose branch is implemented as a multi-layer perceptron.

There has been a few papers that only estimate the camera pose [70], [98], [99]. Unlike techniques that do simultaneously reconstruction, these approaches are trained with only pose annotations. For instance, Kendall *et al.* [98] introduced PoseNet, a convolutional neural network which estimates the camera pose from a single image. The network, which represents the camera pose using its location vector and orientation quaternion, is trained to minimize the  $L_2$  loss between the ground-truth and the estimate pose. Su *et al.* [99] found that CNNs trained for viewpoint estimation of one class do not perform well on another class, possibly due to the huge geometric variation between the classes. As such, they proposed a network architecture where the lower layers (both convolutional layers and fully connected layers) are shared by all classes, while class-dependent fully-connected layers are stacked over them.

## 7.2 Training with video supervision

Another approach to significantly lower the level of supervision required to learn the 3D geometry of objects is by replacing 3D supervision with motion. To this end, Novotni *et al.* [100] used Structure-from Motion (SfM) to generate a supervisory signal from videos. That is, at training, the approach takes a video sequences, generates a partial point cloud and the relative camera parameters using SfM [101]. Each RGB frame is then processed with a network that estimates a depth map, an uncertainty map, and the camera parameters. The different depth estimates are fused, using the estimated camera parameters, into a partial point cloud, which is further processed for completion using the point cloud completion network PointNet [87]. The network is trained using the estimates of the SfM as supervisory signals. That is, the loss functions measure the discrepancy between the depth maps estimated by the network and the depth maps estimated by SfM, and between the camera parameters estimated by the network and those estimated by SfM. At test time, the network is able to recover a full 3D geometry from a single RGB image.

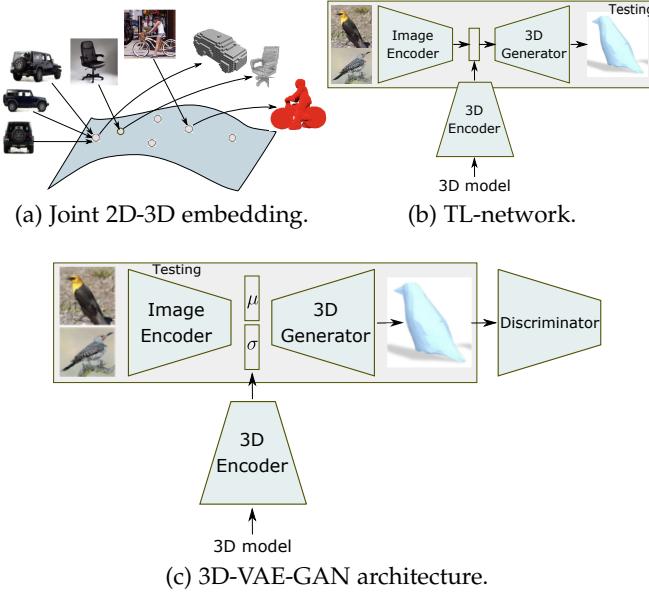


Fig. 5: At test time, the 3D encoder and the discriminator are removed and only the highlighted modules are kept.

### 7.3 Training procedure

In addition to the datasets, loss functions, and degree of supervision, there are a few practical aspects that one needs to consider when training deep learning architectures for 3D reconstruction.

#### 7.3.1 Joint 2D-3D embedding

Most of the state-of-the-art works map the input (*e.g.*, RGB images) into a latent representation, and then decode the latent representation into a 3D model. A good latent representation should be **(1)** generative in 3D, *i.e.*, we should be able to reconstruct objects in 3D from it, and **(2)** it must be predictable from 2D, *i.e.*, we should be able to easily infer this representation from images [25]. Achieving these two goals has been addressed by using TL-embedding networks during the training phase, see Fig. 5-(a) and (b). It is composed of two jointly trained encoding branches: the 2D encoder and the 3D encoder. They map, respectively, a 2D image and its corresponding 3D annotation into the same point in the latent space [24], [25].

Gidhar *et al.* [25], which use the TL-embedding network to reconstruct volumetric shapes from RGB images, train the network using batches of (image, voxel) pairs. The images are generated by rendering the 3D model and the network is then trained in a three stage procedure.

- In the first stage, the 3D encoder part of the network and its decoder are initialized at random. They are then trained, end-to-end with the sigmoid cross-entropy loss, independently of the 2D encoder.
- In the second stage, the 2D encoder is trained to regress the latent representation. The encoder generates the embedding for the voxel, and the image network is trained to regress the embedding.
- The final stage jointly fine-tunes the entire network.

This approach has been extended by Li *et al.* [79] and Mandikal *et al.* [21] for point cloud-based 3D reconstruction

by replacing the volume encoder by a point cloud auto-encoder.

#### 7.3.2 Adversarial training

In general, a good reconstruction model should be able to go beyond what has been seen during training. Networks trained with standard procedures may not generalize well to unseen data. Also, Yang *et al.* [46] noted that the results of standard techniques tend to be grainy and lack fine details. To overcome these issues, several recent papers train their networks with adversarial loss by using Generative Adversarial Networks (GAN). GANs generate a signal from a given random vector [102]. Conditional GANs, on the other hand, conditions the generated signal on the input image(s), see Fig. 5-(c). It consists of a generator  $g$ , which mirrors the encoder  $h$ , and a discriminator  $D$ , which mirrors the generator.

In the case of 3D reconstruction, the encoder can be a ConvNet/ResNet [46], [103] or a variational auto-encoder (VAE) [17]. The generator decodes the latent vector  $\mathbf{x}$  into a 3D shape  $X = g(\mathbf{x})$ . The discriminator, which is only used during training, evaluates the authenticity of the decoded data. It outputs a confidence  $C(X)$  between 0 and 1 of whether the 3D object  $X$  is real or synthetic, *i.e.*, coming from the generator. The goal is to jointly train the generator and the discriminator to make the reconstructed shape as close as possible to the ground truth.

Central to GAN is the adversarial loss function used to jointly train the discriminator and the generator. Following Goodfellow *et al.* [102], Wu *et al.* [17] use the binary cross entropy as the classification loss. The overall adversarial loss function is defined as:

$$\mathcal{L}_{3D-GAN} = \log(D(X)) + \log(1 - D(g(\mathbf{x}))). \quad (14)$$

Here  $\mathbf{x} = h(\mathbf{I})$  where  $\mathbf{I}$  is the 2D images(s) of the training shape  $X$ . Yang *et al.* [46], [103] observed that the original GAN loss function presents an overall loss for both real and fake input. They then proposed to use the WGAN-GP loss [104], [105], which separately represents the loss for generating fake reconstruction pairs and the loss for discriminating fake and real reconstruction pairs, see [104], [105] for the details.

To jointly train the three components of the network, *i.e.*, the encoder, the generator, and the discriminator, the overall loss is defined as the sum of the reconstruction loss, see Section 7.1, and the GAN loss. When the network uses a variational auto-encoder, *e.g.*, the 3D VAE-GAN [17], then an additional term is added to the overall loss in order to push the variational distribution towards the prior distribution. For example, Wu *et al.* [17] used a KL-divergence metric, and a multivariate Gaussian distribution with zero-mean and unit variance as a prior distribution.

The potential of GANs is huge, because they can learn to mimic any distribution of data. They are also very suitable for single-view 3D shape reconstruction. They have been used for volumetric [13], [17], [30], [40], [46], [103] and point cloud [74], [75] reconstruction. They have been used with 3D supervision [17], [30], [40], [46], [103] and with 2D supervision as in [13], [27], [97], see Section 7.1.2. The latter methods train a single discriminator with 2D silhouette images. However, among plausible shapes, there are still

multiple shapes that fit the 2D image equally well. To address this ambiguity, Wu *et al.* [91] used the discriminator of the GAN to penalize the 3D estimator if the predicted 3D shape is unnatural. Li *et al.* [106], on the other hand, use multiple discriminators, one for each view, resulting in a better generation quality.

GANs are hard to train, especially for the complex joint data distribution over 3D objects of many categories and orientations. They also become unstable for high-resolution shapes. In fact, one must carefully balance the learning of the generator and the discriminator, otherwise the gradients can vanish, which will prevent improvement [40]. To address this issue, Smith and Meger [40] and later Wu *et al.* [91] used as a training objective the Wasserstein distance normalized with the gradient penalization.

### 7.3.3 Joint training with other tasks

Jointly training for reconstruction and segmentation leads to improved performance in both tasks, when compared to training for each task individually. Mandikal *et al.* [107] proposed an approach, which generates a part-segmented 3D point cloud from one RGB image. The idea is to enable propagating information between the two tasks so as to generate more faithful part reconstructions while also improving segmentation accuracy. This is done using a weighted sum of a reconstruction loss, defined using the Chamfer distance, and a segmentation loss, defined using the symmetric softmax cross-entropy loss.

## 8 APPLICATIONS AND SPECIAL CASES

Image-based 3D reconstruction is an important problem and a building block to many applications ranging from robotics and autonomous navigation to graphics and entertainment. While some of these applications deal with generic objects, many of them deal with objects that belong to specific classes such as human bodies or body parts (*e.g.*, faces and hands), animals in the wild, and cars. The techniques described above can be applied to these specific classes of shapes. However, the quality of the reconstruction can be significantly improved by designing customised methods that leverage the prior knowledge of the shape class. In this section, we will briefly summarize recent developments in the image-based 3D reconstruction of human body shapes (Section 8.1), and body parts such as faces (Section 8.2). We will also discuss in Section 8.3 methods that deal with the parsing entire 3D scenes.

### 8.1 3D human body reconstruction

3D static and dynamic digital humans are essential for a variety of applications ranging from gaming, visual effects to free-viewpoint videos. However, high-end 3D capture solutions use a large number of cameras and active sensors, and are restricted to professionals as they operate under controlled lighting conditions and studio settings. With the avenue of deep learning techniques, several papers have explored more lightweight solutions that are able to recover the 3D human shape and pose from a few RGB images. We can distinguish two classes of methods; **(1)** volumetric methods (Section 4), and **(2)** template or parameteric-based

methods (Section 5.2). Some methods in both categories reconstruct naked 3D human body shapes [108], [109], while others recover also cloths and garments [110], [111].

#### 8.1.1 Parametric methods

Parametric methods regularize the problem using statistical models such as morphable models [112], SCAPE [113], and SMPL [114]. The problem of 3D human body shape reconstruction then boils down to estimating the parameters of the model.

Dibra *et al.* [108] used an encoder followed by three fully connected layers to regress the SCAPE parameters from one or multiple silhouette images. Later, Dibra *et al.* [109] first learn a common embedding of 2D silhouettes and 3D human body shapes (see Section 7.3.1). The latter are represented using their Heat Kernel Signatures [115]. Both methods can only predict naked body shapes in nearly neutral poses.

SMPL has the advantage of encoding in a disentangled manner both the shape, the pose, and the pose specific details, and thus it has been extensively used in deep learning-based human body shape reconstruction [110], [116], [117], [118]. Bogo *et al.* [116] proposed SMPLify, the first 3D human pose and shape reconstruction from one image. They first used a CNN-based architecture, DeepCut [119], to estimate the 2D joint locations. They then fit an SMPL model to the predicted 2D joints giving the estimation of 3D human body pose and shape. The training procedure minimizes an objective function of five terms: a joint-based data term, three pose priors, and a shape prior. Experimental results show that this method is effective in 3D human body reconstruction from arbitrary poses.

Kanazawa *et al.* [120], on the other hand, argue that such a stepwise approach is not optimal and propose an end-to-end solution to learn a direct mapping from image pixels to model parameters. This approach addresses two important challenges: **(1)** the lack of large scale ground truth 3D annotations for in-the-wild images, and **(2)** the inherent ambiguities in single 2D-view-to-3D mapping of human body shapes. An example is depth ambiguity where multiple 3D body configurations can explain the same 2D projections [116]. To address the first challenge, Kanazawa *et al.* observe that there are large-scale 2D keypoint annotations of in-the-wild images and a separate large-scale dataset of 3D meshes of people with various poses and shapes. They then take advantage of these unpaired 2D keypoint annotations and 3D scans in a conditional generative adversarial manner. They propose a network that infers the SMPL [114] parameters of a 3D mesh and the camera parameters such that the 3D keypoints match the annotated 2D keypoints after projection. To deal with ambiguities, these parameters are sent to a discriminator whose task is to determine if the 3D parameters correspond to bodies of real humans or not. Hence, the network is encouraged to output parameters on the human manifold. The discriminator acts as a weak supervisor.

These approaches can handle complex poses from images with complex backgrounds, but are limited to a single person per image and does not handle clothes. Also, these approaches do not capture details such as hair and clothing with garment wrinkles, as well has details on the body

parts. To capture these details, Alldieck *et al.* [118] train an encoder-decoder to predict normals and displacements, which can then be applied to the reconstructed SMPL model.

### 8.1.2 Volumetric methods

Volumetric techniques for 3D human body reconstruction do not use statistical models. Instead, they directly infer 3D occupancy grids. As such, all the methods reviewed in Section 4 can be used for 3D human body shape reconstruction. An example is the approach of Huang *et al.* [121], which takes multiple RGB views and their corresponding camera calibration parameters as input, and predicts a dense 3D field that encodes for each voxel its probability of being inside or outside the human body shape. The surface geometry can then be faithfully reconstructed from the 3D probability field using marching cubes. The approach uses a multi-branch encoder, one for each image, followed by a multi-layer perceptron which aggregates the features that correspond to the same 3D point into a probability value. The approach is able to recover detailed geometry even on human bodies with cloth but it is limited to simple backgrounds.

To exploit domain-specific knowledge, Varol *et al.* [122] introduce BodyNet, a volumetric approach for inferring, from a single RGB image, the 3D human body shape, along with its 2D and 3D pose, and its partwise segmentation. The approach uses a cascade of four networks; (1) a 2D pose and a 2D segmentation network, which operate in parallel, (2) a 3D pose inference network, which estimates the 3D pose of the human body from the input RGB image and the estimated 2D pose and 2D partwise segmentation, and (4) finally, a 3D shape estimation network, which infers a volumetric representation of the human body shape and its partwise segmentation from the input RGB image and the estimates of the previous networks. By dividing the problem into four tasks, the network can benefit from intermediate supervision, which results in an improved performance.

## 8.2 3D face reconstruction

Detailed and dense image-based 3D reconstruction of the human face, which aims to recover shape, pose, expression, skin reflectance, and finer scale surface details, is a longstanding problem in computer vision and computer graphics. Recently, this problem has been formulated as a regression problem and solved using convolutional neural networks.

In this section, we review some of the representative papers. Most of the recent techniques use parametric representations, which parametrize the manifold of 3D faces. The most commonly used representation is the 3D morphable model (3DMM) of Blanz and Vetter [68], which is an extension of the 2D active appearance model [123] (see also Section 5.2.1). The model captures facial variability in terms of geometry and texture. Gerig *et al.* [124] extended the model by including expressions as a separate space. Below, we discuss the various network architectures (Section 8.2.1) and their training procedures (Section 8.2.2). We will also discuss some of the model-free techniques (Section 8.2.3).

### 8.2.1 Network architectures

The backbone architecture is an encoder, which maps the input image into the parametric model parameters. It is composed of convolutional layers followed by fully connected layers. In general, existing techniques use generic networks such as AlexNet, or networks specifically trained on facial images such as VGG-Face [125] or FaceNet [126]. Tran *et al.* [127] use this architecture to regress the 198 parameters of a 3DMM that encodes facial identity (geometry) and texture. It has been trained with 3D supervision using  $L_2$  asymmetric loss, *i.e.*, a loss function that favours 3D reconstructions that are far from the mean.

Richardson *et al.* [128] used a similar architecture but perform the reconstruction iteratively. At each iteration, the network takes the previously reconstructed face, but projected onto an image using a frontal camera, with the input image, and regresses the parameters of a 3DMM. The reconstruction is initialized with the average face. Results show that, with three iterations, the approach can successfully handle face reconstruction from images with various expressions and illumination conditions.

One of the main issues with 3DMM-based approaches is that they tend to reconstruct smooth facial surfaces, which lack fine details such as wrinkles and dimples. As such, methods in this category use a refinement module to recover the fine details. For instance, Richardson *et al.* [128] refine the reconstructed face using Shape from Shading (SfS) techniques. Richardson *et al.* [129], on the other hand, add a second refinement block, FineNet, which takes as input the depth map of the coarse estimation and recovers using an encoder-decoder network a high resolution facial depth map. To enable end-to-end training, the two blocks are connected with a differentiable rendering layer. Unlike traditional SfS, the introduction of FineNet treats the calculation of albedo and lighting coefficients as part of the loss function without explicitly estimating these information. However, lighting is modeled by first-order spherical harmonics, which lead to an inaccurate reconstruction of the facial details.

### 8.2.2 Training and supervision

One of the main challenges is in how to collect enough training images labelled with their corresponding 3D faces, to feed the network. Richardson *et al.* [128], [129] generate synthetic training data by drawing random samples from the morphable model and rendering the resulting faces. However, a network trained on purely synthetic data may perform poorly when faced with occlusions, unusual lighting, or ethnicities that are not well represented. Genova *et al.* [130] address the lack of training data by including randomly generated synthetic faces in each training batch to provide ground truth 3D coordinates, but train the network on real photographs at the same time. Tran *et al.* [127] use an iterative optimization to fit an expressionless model to a large number of photographs, and treat the results where the optimization converged as ground truth. To generalize to faces with expressions, identity labels and at least one neutral image are required. Thus, the potential size of the training dataset is restricted.

Tewari *et al.* [131] train, without 3D supervision, an encoder-decoder network to simultaneously predict the fa-

cial shape, expression, texture, pose, and lighting. The encoder is a regression network from images to morphable model coordinates, and the decoder is a fixed, differentiable rendering layer that attempts to reproduce the input photograph. The loss measures the discrepancy between the reproduced photograph and the input one. Since the training loss is based on individual image pixels, the network is vulnerable to confounding variation between related variables. For example, it cannot readily distinguish between dark skin tone and a dim lighting environment.

To remove the need for supervised training with 3D data and the reliance on inverse rendering, Genova *et al.* [130] propose a framework that learns to minimize a loss based on the facial identity features produced by a face recognition network such as VGG-Face [125] or Google's FaceNet [126]. In other words, the face recognition network encodes the input photograph as well as the image rendered from the reconstructed face into feature vectors that are robust to pose, expression, lighting, and even non-photorealistic inputs. The method then applies a loss that measures the discrepancy between these two feature vectors instead of using pixel-wise distance between the rendered image and the input photograph. The 3D facial shape and texture regressor network is trained using only a face recognition network, a morphable face model, and a dataset of unlabelled facial images. The approach does not only improve on the accuracy of previous works but also produces 3D reconstructions that are often recognizable as the original subjects.

### 8.2.3 Model-free approaches

Morphable model-based techniques are restricted to the modelled subspace. As such, implausible reconstructions are possible outside the span of the training data. Other representations such as volumetric grids, which do not suffer from this problem, have been also explored in the context of 3D face reconstruction. Jackson *et al.* [132], for example, propose a Volumetric Regression Network (VRN), which takes as input 2D images and predicts their corresponding 3D binary volume instead of a 3DMM. Unlike [127], the approach can deal with a wide range of expressions, poses and occlusions without alignment and correspondences. It, however, fails to recover fine details due to the resolution restriction of volumetric techniques.

Other techniques use intermediate representations. For example, Sela *et al.* [133] use an Image-to-Image Translation Network based on U-Net [48] to estimate a depth image and a facial correspondence map. Then, an iterative deformation-based registration is performed followed by a geometric refinement procedure to reconstruct subtle facial details. Unlike 3DMM, this method can handle large geometric variations.

Feng *et al.* [134] also investigated a model-free method. First, a densely connected CNN framework is designed to regress 3D facial curves from horizontal and vertical epipolar plane images. Then, these curves are transformed into a 3D point cloud and the grid-fit algorithm [135] is used to fit a facial surface. Experimental results suggest that this approach is robust to varying poses, expressions and illumination.

### 8.3 3D scene parsing

Methods discussed so far are primarily dedicated to the 3D reconstruction of objects in isolation. Scenes with multiple objects pose the additional challenges of delineating objects, properly handling occlusions, clutter, and uncertainty in shape and pose, and estimating the scene layout. Solutions to this problem involve 3D object detection and recognition, pose estimation, and 3D reconstruction. Traditionally, many of these tasks have been addressed using hand-crafted features. In the deep learning-based era, several of the blocks of the pipeline have been replaced with CNNs.

For instance, Izadinia *et al.* [136] proposed an approach that is based on recognizing objects in indoor scenes, inferring room geometry, and optimizing 3D object poses and sizes in the room to best match synthetic renderings to the input photo. The approach detects object regions, finds from a CAD database the most similar shapes, and then deforms them to fit the input. The room geometry is estimated using a fully convolutional network. Both the detection and retrieval of objects are performed using Faster R-CNN [137]. The deformation and fitting, however, are performed via render and match. Tulsiani *et al.* [138], on the other hand, proposed an approach that is entirely based on deep learning. The input, which consists of an RGB image and the bounding boxes of the objects, is processed with a four-branch network. The first branch is an encoder-decoder with skip connections, which estimates the disparity of the scene layout. The second branch takes a low resolution image of the entire scene and maps it into a latent space using a CNN followed by three fully-connected layers. The third branch, which has the same architecture as the second one, maps the image at its original resolution to convolutional feature maps, followed by ROI pooling to obtain features for the ROI. The last layer maps the bounding box location through fully connected layers. The three features are then concatenated and further processed with fully-connected layers followed by a decoder, which produces a  $32^3$  voxel grid of the object in the ROI and its pose in the form of position, orientation, and scale. The method has been trained using synthetically-rendered images with their associated ground-truth 3D scene.

## 9 DATASETS

Table 5 lists and summarizes the properties of the most commonly used datasets. Unlike traditional techniques, the success of deep learning-based 3D reconstruction algorithms depends on the availability of large training datasets. Supervised techniques require images and their corresponding 3D annotations in the form of (1) full 3D models represented as volumetric grids, triangular meshes, or point clouds, or (2) depth maps, which can be dense or sparse. Weakly supervised and unsupervised techniques, on the other hand, rely on additional supervisory signals such as the extrinsic and intrinsic camera parameters and segmentation masks.

The main challenge in collecting training datasets for deep learning-based 3D reconstruction is two-fold. First, while one can easily collect 2D images, obtaining their corresponding 3D ground truth is challenging. As such, in many datasets, IKEA, PASCAL 3D+, and ObjectNet3D, only a relatively small subset of the images are annotated with

TABLE 5: Some of the datasets that are used to train and evaluate the performance of deep learning-based 3D reconstruction algorithms. "img": image. "obj": object. "Bkg": background. "cats": categories. "GT": ground truth.

Year	Images					Objects		3D ground truth			Camera params
	No. imgs	Size	objs per img	Type	Bkg	Type	No. cats	No.	Type	Img with 3D GT	
ShapeNet [139]	2015	—	—	Single rendered	Uniform	Generic	55	51,300	3D model	51,300	Intrinsic
ModelNet [3]	2015	—	—	Single Rendered	Uniform	Generic	662	127,915	3D model	127,915	Intrinsic
IKEA [140]	2013	759	Variable	Single Real, indoor	Cluttered	Generic	7	219	3D model	759	Intrinsic+extrinsic
Pix3D [9]	2018	9,531	110 × 110 to 3264 × 2448	Single Real, indoor	Cluttered	Generic	9	1015	3D model	9,531	Focal length, extrinsic
PASCAL 3D+ [141]	2014	30,899	Variable	Multiple Real, indoor, outdoor	Cluttered	Generic	12	36,000	3D model	30,809	Intrinsic+extrinsic
ObjectNet3D [142]	2016	90,127	Variable	Multiple Real, indoor, outdoor	Cluttered	Generic	100	44,147	3D model	90,127	Intrinsic+extrinsic
KITTI12 [143]	2012	41,778	1240 × 376	Multiple Real, outdoor	Cluttered	Generic	2	40,000	Point cloud	12,000	Intrinsic+extrinsic
ScanNet [144]	2017, 2018	2,492, 518	640 × 480, RGB 1296 × 968	Multiple Real, indoor	Cluttered	Generic	296	36,123	Dense depth	2,492, 518	Intrinsic+extrinsic
Stanford Car [145]	2013	16,185	Variable	Single Real, outdoor	Cluttered	Cars	196	—	—	—	—
Caltech-UCSD Birds 200 [146], [147]	2010, 2011	6,033	Variable	Single Real	Cluttered	Birds	200	—	—	—	Extrinsic
SUNCG [148]	2017	130,269	—	Multiple Synthetic	Cluttered	Generic	84	5,697,217	Depth, voxel grid	130,269	Intrinsic
Stanford 2D-3D-S [149], [150]	2016, 2017	70,496	1080 × 1080	Multiple Real, indoor	Cluttered	Generic	13	6,005	Point cloud, mesh	70,496	Intrinsic + extrinsic
ETHZ CVL RueMonge [151]	2014	428	—	Multiple Real, outdoor	Cluttered	Generic	8	—	Point cloud, mesh	428	Intrinsic
NYU V2 [152]	2012	1,449	640 × 480	Multiple Real, indoor	Cluttered	Generic	894	35,064	Dense depth, 3D points	1,449	Intrinsic

their corresponding 3D models. **Second**, datasets such as ShapeNet and ModelNet, which are the largest 3D datasets currently available, contain 3D CAD models without their corresponding *natural* images since they have been originally intended to benchmark 3D shape retrieval algorithms.

This issue has been addressed in the literature by data augmentation, which is the process of augmenting the original sets with synthetically-generated data. For instance, one can generate new images and new 3D models by applying geometric transformations, *e.g.*, translation, rotation, and scaling, to the existing ones. Note that, although some transformations are similarity-preserving, they still enrich the datasets. One can also synthetically render, from existing 3D models, new 2D and 2.5D (*i.e.*, depth) views from various (random) viewpoints, poses, lighting conditions, and backgrounds. They can also be overlaid with natural images or random textures. This, however, results in the domain shift problem, *i.e.*, the space of synthetic images is different from the space of real images, which often results in a decline in performance when methods are tested on images of a completely different type.

Domain shift problem in machine learning has been traditionally addressed using domain adaptation or translation techniques, which are becoming popular in depth estimation [153]. They are, however, not commonly used for 3D reconstruction. An exception is the work of Petersen *et al.* [95], which observed that a differentiable renderer used in unsupervised techniques may produce images that are different in appearance compared to the input images. This has been alleviated through the use of image domain translation.

Finally, weakly supervised and unsupervised techniques (Section 7.1.2) minimize the reliance on 3D annotations. They, however, require (1) segmentation masks, which can be obtained using the recent state-of-the-art object detection and segmentation algorithms [154], and/or (2) camera parameters. Jointly training for 3D reconstruction, segmentation, and camera parameters estimation can be a promising direction for feature research.

## 10 PERFORMANCE COMPARISON

This section discusses the performance of some key methods. We will present the various performance criteria and metrics (Section 10.1), and discuss and compare the performance of some key methods (Section 10.2).

### 10.1 Accuracy metrics and performance criteria

Let  $X$  be the ground truth 3D shape and  $\hat{X}$  the reconstructed one. Below, we discuss some of the accuracy metrics (Section 10.1.1) and performance criteria (Section 10.1.2) used to compare 3D reconstruction algorithms.

#### 10.1.1 Accuracy metrics

The most commonly used quantitative metrics for evaluating the accuracy of 3D reconstruction algorithms include:

(1) **The Mean Squared Error (MSE)** [60]. It is defined as the symmetric surface distance between the reconstructed shape  $\hat{X}$  and the ground-truth shape  $X$ , *i.e.*,

$$d(\hat{X}, X) = \frac{1}{n_X} \sum_{p \in X} d(p, \hat{X}) + \frac{1}{n_{\hat{X}}} \sum_{\hat{p} \in \hat{X}} d(\hat{p}, X). \quad (15)$$

Here,  $n_X$  and  $n_{\hat{X}}$  are, respectively, the number of densely sampled points on  $X$  and  $\hat{X}$ , and  $d(p, X)$  is the distance, *e.g.*, the  $L_1$  or  $L_2$  distance, of  $p$  to  $X$  along the normal direction to  $X$ . The smaller this measure is, the better is the reconstruction.

(2) **Intersection over Union (IoU)**. The IoU measures the ratio of the intersection between the volume of the predicted shape and the volume of the ground-truth, to the union of the two volumes, *i.e.*,

$$IoU_{\epsilon} = \frac{\hat{V} \cap V}{\hat{V} \cup V} = \frac{\sum_i \{I(\hat{V}_i > \epsilon) * I(V_i)\}}{\sum_i \{I(I(\hat{V}_i > \epsilon) + I(V_i))\}}, \quad (16)$$

where  $I(\cdot)$  is the indicator function,  $\hat{V}_i$  is the predicted value at the  $i$ -th voxel,  $V_i$  is the ground truth, and  $\epsilon$  is a threshold. The higher the IoU value, the better is the reconstruction. This metric is suitable for volumetric reconstructions. Thus, when dealing with surface-based representations, the reconstructed and ground-truth 3D models need to be voxelized.

(3) **Mean of Cross Entropy (CE) loss** [103]. It is defined as follows;

$$CE = -\frac{1}{N} \sum_{i=1}^N \{p_i \log \hat{p}_i + (1 - p_i) \log(1 - \hat{p}_i)\}. \quad (17)$$

where  $N$  is the total number of voxels or points, depending whether using a volumetric or a point-based representation.  $p$  and  $\hat{p}$  are, respectively, the ground-truth and the predicted

value at the  $i$ -voxel or point. The lower the CE value is, the better is the reconstruction.

**(4) Earth Mover Distance (EMD) and Chamfer Distance (CD).** These distances are, respectively, defined in Equations (5) and (6).

#### 10.1.2 Performance criteria

In addition to these quantitative metrics, there are several qualitative aspects that are used to evaluate the efficiency of these methods. This includes:

**(1) Degree of 3D supervision.** An important aspect of deep learning-based 3D reconstruction methods is the degree of 3D supervision they require at training. In fact, while obtaining RGB images is easy, obtaining their corresponding ground-truth 3D data is quite challenging. As such, techniques that require minimal or no 3D supervision are usually preferred over those that require ground-truth 3D information during training.

**(2) Computation time.** While training can be slow, in general, it is desirable to achieve real-time performance at runtime.

**(3) Memory footprint.** Deep neural networks have a large number of parameters. Some of them operate on volumes using 3D convolutions. As such, they usually require a large memory storage, which can affect their performance at runtime and limit their usage.

## 10.2 Comparison and discussion

We present the improvement in reconstruction accuracy over the past 4 years in Fig. 6, and the performance of some representative methods in Table 6.

The majority of early works resort to voxelized representations [4], [7], [17], [25], [156], which can represent both the surface and the internal details of complex objects of arbitrary topology. With the introduction of space partitioning techniques such as O-CNN [32], OGN [33], and OctNet [41], volumetric techniques can attain relatively high resolutions, e.g.,  $512^3$ . This is due to the significant gain in memory efficiency. For instance, the OGN of [33] reduces the memory requirement for the reconstruction of volumetric grids of size  $32^3$  from 4.5GB in [7] and 1.7GB in [12] to just 0.29GB (see Table 6). However, only a few papers, e.g., [35], adopted these techniques due to the complexity of their implementation. To achieve high resolution 3D volumetric reconstruction, many recent papers use intermediation, through multiple depth maps, followed by volumetric [38], [46], [51], [92] or point-based [80] fusion. More recently, several papers start to focus on mechanisms for learning continuous Signed Distance Functions [39], [43] or continuous occupancy grids [157], which are less demanding in terms of memory requirement. Their advantage is that since they learn a continuous field, the reconstructed 3D object can be extracted at the desired resolution.

Fig. 6 shows the evolution of the performance over the years, since 2016, using the ShapeNet dataset [3] as a benchmark. On the IoU metric, computed on volumetric grids of size  $32^3$ , we can see that methods that use multiple views at training and/or at testing outperform those

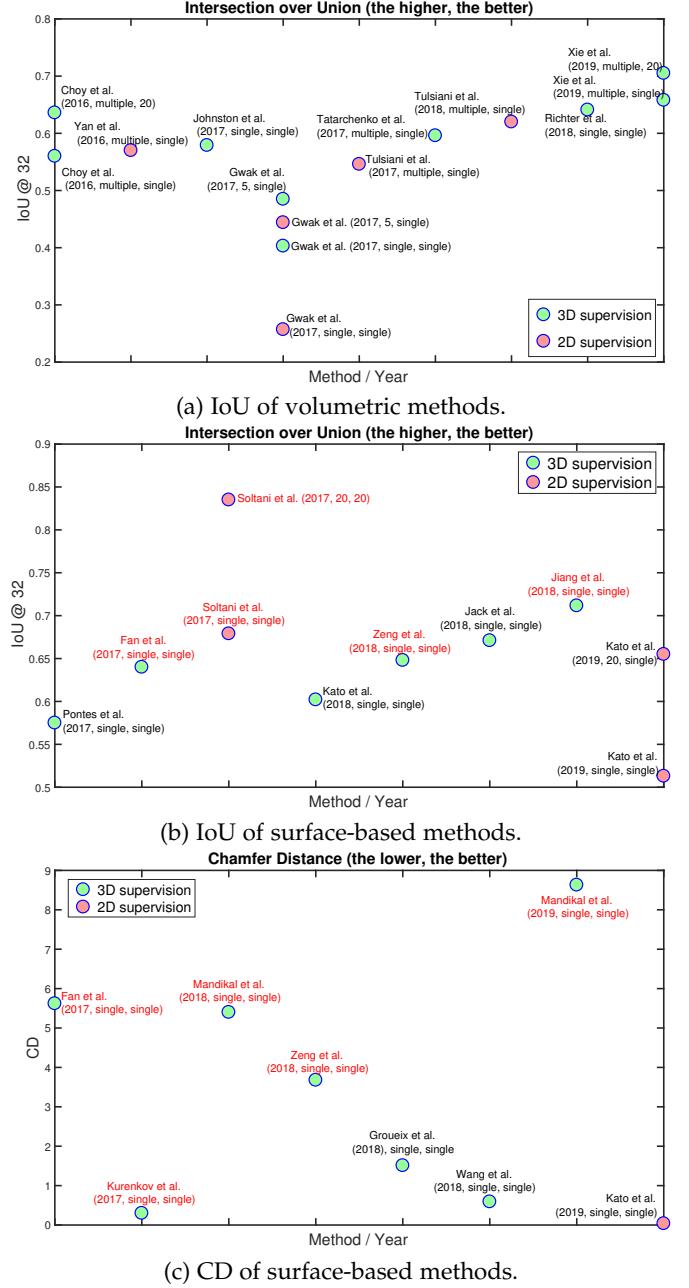


Fig. 6: Performance of some key methods on the ShapeNet dataset. References highlighted in red are point-based. The IoU is computed on grids of size  $32^3$ . The label next to each circle is encoded as follow: First author *et al.* (year,  $n$  at training,  $n$  at test), where  $n$  is the number of input images. Table 6 provides a detailed comparison.

that are based solely on single views. Also, surface-based techniques, which started to emerge in 2017 (both mesh-based [60] and point-based [59], [72]), slightly outperform volumetric methods. Mesh-based techniques, however, are limited to genus-0 surfaces or surfaces with the same topology as the template.

Fig. 6 shows that, since their introduction in 2017 by Yan *et al.* [4], 2D supervision-based methods significantly improved in performance. The IoU curves of Figures 6-(a) and (b), however, show that methods that use 3D su-

TABLE 6: Performance summary of some representative methods. "Obj.": objects. "Time" refers to timing in milliseconds. "U3D": unlabelled 3D. # params: number of the parameters of the network. "mem.": memory requirements. "Resol.": resolution. "Bkg.": background.

Method	Input				Output				Training				Performance@ShapeNet, Pix3D,Pascal3D+)			
	Train	Test	Bkg	# objects	Type	Resol.	Supervision	Network	IoU	CD	Time	Memory	(# params, mem.)			
Xie <i>et al.</i> [86]	$n \geq 1$ RGB, 3D GT	1 RGB 20 RGB	Clutter	1	Volumetric	3 <sup>2</sup>	3D 3D	Encoder (VGG16), Decoder, Refiner	(0.658, −, 0.669)@32 <sup>3</sup> (0.705, −, −)@32 <sup>3</sup>	(−, −, −) (−, −, −)	9.9	(114.4M, −)	−	−	−	−
Richter <i>et al.</i> [51]	$n = 1$ RGB, 3D GT	1 RGB	Clean	1	Volumetric	512 <sup>3</sup>	3D	encoder, 2D decoder	(0.641, −, −)@32 <sup>3</sup>	(−, −, −)	−	−	−	−	−	−
Tulsiani <i>et al.</i> [11]	$n > 1$ RGB, segmentation	1 RGB	Clutter	1	Volumetric + pose	−	2D (multiview)	Encoder, decoder, pose CNN	(0.62, −, −)@32 <sup>3</sup>	(−, −, −)	−	−	−	−	−	−
Tatarchenko <i>et al.</i> [33]	$n \geq 1$ RGB + 3D GT	1 RGB	Clutter	1	Volumetric	512 <sup>3</sup> 3 <sup>2</sup> <sub>3</sub>	3D	Octree Generating Network	(0.596, −, 0.594)@32 <sup>3</sup> (0.546, −, 0.536)@32 <sup>3</sup>	(−, −, −) (−, −, −)	2.06s	(−, 0.88GB)	16	(12.46M, 0.29GB)	−	−
Tulsiani <i>et al.</i> [8]	$n > 1$ RGB, silh., depth	1 RGB	Clutter	1	Volumetric	3 <sup>2</sup> <sub>3</sub>	2D (multiview)	encoder-decoder	(0.546, −, 0.539)@32 <sup>3</sup>	(−, −, −)	−	−	−	−	−	−
Wu <i>et al.</i> [6]	$n = 1$ RGB, 3D GT	1 RGB	Clutter	1	Volumetric	128 <sup>3</sup>	2D and 2.5D	TL_3D-VAE + encoder-decoder	(0.57, −, 0.39)@128 <sup>3</sup>	(−, −, −)	−	−	−	−	−	−
Gwak <i>et al.</i> [97]	1 RGB, silh., pose 1 RGB, silh., pose, U3D 5 RGB, silh., pose, U3D 5 RGB, silh., pose, U3D	1 RGB 1 RGB 1 RGB 1 RGB	Clutter	1	Volumetric	−	2D	GAN	(0.257, −, −)@32 <sup>3</sup> (0.403, −, −)@32 <sup>3</sup> (0.444, −, −)@32 <sup>3</sup> (0.4849, −, −)@32 <sup>3</sup>	(−, −, −) (−, −, −) (−, −, −) (−, −, −)	−	−	−	−	−	−
Johnston <i>et al.</i> [12]	$n = 1$ RGB, 3D GT	1 RGB	Clutter	1	Volumetric	128 <sup>3</sup> 3 <sup>2</sup> <sub>3</sub>	3D 3D	Encoder + IDCT Encoder + IDCT	(0.417, −, 0.4496)@128 <sup>3</sup> (0.579, −, 0.5474)@32 <sup>3</sup>	(−, −, −) (−, −, −)	32	(−, 2.2GB)	15	(−, 1.7GB)	−	−
Yan <i>et al.</i> [4]	$n > 1$ RGB, silh., pose	1 RGB	Clean	1	Volumetric	3 <sup>2</sup> <sub>3</sub>	2D	encoder-decoder	(0.57, −, −)@32 <sup>3</sup>	(−, −, −)	−	−	−	−	−	−
Choy <i>et al.</i> [7]	$n \geq 1$ RGB, 3D GT	1 RGB 20 RGB	Clutter	1	Volumetric	3 <sup>2</sup> <sub>3</sub> 3 <sup>2</sup> <sub>3</sub>	3D	encoder-LSTM- decoder	(0.56, −, 0.537)@32 <sup>3</sup> (0.636, −, −)@32 <sup>3</sup>	(−, −, −) (−, −, −)	73.35	(35.97M, > 4.5GB)	−	−	−	−
Kato <i>et al.</i> [155]	$n = 1$ RGB, silh., pose 20 RGB, silh., poses	1 RGB 1 RGB	Clutter	1	Mesh + texture	−	2D	Encoder-Decoder	(0.513, −, −)@32 <sup>3</sup> (0.655, −, −)@32 <sup>3</sup>	(0.0378, −, −) (−, −, −)	−	−	−	−	−	−
Mandikal <i>et al.</i> [81]	$n = 1$ RGB, 3D GT	1 RGB	Clean	1	Point cloud	16384	3D	Conv + FC layers, Global to local	(−, −, −)	(8.63, −, −)	−	(13.3M, −)	−	−	−	−
Jiang <i>et al.</i> [74]	$n = 1$ RGB + 3D GT	1 RGB	Clutter	1	Point cloud	1024	3D	2 × (encoder-decoder), GAN	(0.7116, −, −)@32 <sup>3</sup>	(−, −, −)	−	−	−	−	−	−
Zeng <i>et al.</i> [80]	1 RGB, silh., pose, 3D	1 RGB	Clutter	1	Point cloud	1024	3D + self	encoder-decoder + point auto-encoder	(0.648, −, −)@32 <sup>3</sup>	(3.678, −, −)	−	−	−	−	−	−
Kato <i>et al.</i> [55]	$n = 1$ RGB, silh.,	1 RGB	Clutter	1	Mesh	64 <sup>2</sup>	2D	encoder + FC layers	(0.602, −, −)@32 <sup>3</sup>	(−, −, −)	−	−	−	−	−	−
Jack <i>et al.</i> [58]	1 RGB, FFD params	1 RGB	Clean	1	Mesh	16384	3D	Encoder + FC layers	(0.671, −, −)@32 <sup>3</sup>	(−, −, −)	−	−	−	−	−	−
Groueix <i>et al.</i> [54]	$n = 1$ RGB, 3D GT	1 RGB	Clean	1	Mesh	1024	3D	Multibranch MLP	(−, −, −)	(1.51, −, −)	−	−	−	−	−	−
Wang <i>et al.</i> [56]	1 RGB, 3D GT	1 RGB	Clean	1	Mesh	2466	3D	see Fig. 2 (left)	(−, −, −)	(0.591, −, −)	−	−	−	−	−	−
Mandikal <i>et al.</i> [21]	$n = 1$ RGB, 3D GT	1 RGB	Clutter	1	Point cloud	2048	3D	3D-VAE, TL-embedding	(−, −, −)	(5.4, −, −)	−	−	−	−	−	−
Soltani <i>et al.</i> [19]	20 silh., poses 1 silh.	1 silh. 1 silh.	Clear	1	20 depth maps 20 depth maps	−	2D	3D-VAE	(0.835, −, −)@32 <sup>3</sup> (0.679, −, −)@32 <sup>3</sup>	(−, −, −) (−, −, −)	−	−	−	−	−	−
Fan <i>et al.</i> [72]	$n = 1$ RGB, 3D GT	1 RGB	Clutter	1	Point set	1024	3D	see Fig. 3(a)	(0.64, −, −)@32 <sup>3</sup>	(5.62, −, −)	−	−	−	−	−	−
Pontes <i>et al.</i> [60]	$n = 1$ RGB, GT FFD	1 RGB	Clean	1	Mesh	−	3D	Encoder + FC layers	(0.375, −, 0.299)@32 <sup>3</sup>	(−, −, −)	−	−	−	−	−	−
Kurenkov <i>et al.</i> [59]	$n = 1$ RGB, 3D GT	1 RGB	Clean	1	Point cloud	1024	3D	2D encoder, 3D encoder, 3D decoder	(−, −, −)	(0.3, −, −)	−	−	−	−	−	−

pervision achieve slightly better performance. This can be attributed to the fact that 2D-based supervision methods use loss functions that are based on 2D binary masks and silhouettes. However, multiple 3D objects can explain the same 2D projections. This 2D to 3D ambiguity has been addressed either by using multiple binary masks captured from multiple viewpoints [19], which can only reconstruct the visual hull and as such, they are limited in accuracy, or by using adversarial training [91], [97], which constrains the reconstructed 3D shapes to be within the manifold of valid classes.

## 11 FUTURE RESEARCH DIRECTIONS

In light of the extensive research undertaken in the past five years, image-based 3D reconstruction using deep learning techniques has achieved promising results. The topic, however, is still in its infancy and further developments are yet to be expected. In this section, we present some of the current issues and highlight directions for future research.

(1) *Training data issue.* The success of deep learning techniques depends heavily on the availability of training data. Unfortunately, the size of the publicly available datasets that include both images and their 3D annotations is small compared to the training datasets used in tasks such as classification and recognition. 2D supervision techniques have been used to address the lack of 3D training data. Many of them, however, rely on silhouette-based supervision and thus they can only reconstruct the visual hull. As such, we expect to see in the future more papers proposing new large-scale datasets, new weakly-supervised and unsupervised methods that leverage various visual cues, and new domain adaptation techniques where networks trained with data from a certain domain, *e.g.*, synthetically rendered images, are adapted to a new domain, *e.g.*, in-the-wild images, with minimum retraining and supervision. Research on realistic rendering techniques that are able to close the gap between real images and synthetically rendered images can potentially contribute towards addressing the training data issue.

(2) *Generalization to unseen objects.* Most of the state-of-the-art papers split a dataset into three subsets for training, validation, and testing, *e.g.*, ShapeNet or Pix3D, then report the performance on the test subsets. However, it is not clear how these methods would perform on a completely unseen object/image categories. In fact, the ultimate goal of 3D reconstruction method is to be able to reconstruct any arbitrary 3D shape from arbitrary images. Learning-based techniques, however, perform well only on images and objects spanned by the training set. Some recent papers, *e.g.*, Cherabier *et al.* [38], started to address this issue. An interesting direction for future research, however, would be to combine traditional and learning based techniques to improve the generalization of the latter methods.

(2) *Fine-scale 3D reconstruction.* Current state-of-the-art techniques are able to recover the coarse 3D structure of shapes. Although recent works have significantly improved the resolution of the reconstruction by using refinement modules, they still fail to recover thin and small parts such as plants, hair, and fur.

(3) *Reconstruction vs. recognition.* 3D reconstruction from images is an ill-posed problem. As such, efficient solutions need to combine low-level image cues, structural knowledge, and high-level object understanding. As outlined in the recent paper of Tatarchenko *et al.* [44], deep learning-based reconstruction methods are biased towards recognition and retrieval. As such, many of them do not generalize well and fail to recover fine-scale details. Thus, we expect in the future to see more research on how to combine top-down approaches (*i.e.*, recognition, classification, and retrieval) with bottom-up approaches (*i.e.*, pixel-level reconstruction based on geometric and photometric cues). This also has the potential to improve the generalization ability of the methods, see item (2) above.

(4) *Specialized instance reconstruction.* We expect in the future to see more synergy between class-specific knowledge modelling and deep learning-based 3D reconstruction in order to leverage domain-specific knowledge. In fact, there is an increasing interest in reconstruction methods that are specialized in specific classes of objects such as human bodies and body parts (which we have briefly covered in this survey), vehicles, animals [57], trees, and buildings. Specialized methods exploit prior and domain-specific knowledge to optimise the network architecture and its training process. As such, they usually perform better than the general framework. However, similar to deep learning-based 3D reconstruction, modelling prior knowledge, *e.g.*, by using advanced statistical shape models [65], [66], [67], [158], [159], requires 3D annotations, which are not easy to obtain for many classes of shapes, *e.g.*, animals in the wild.

(5) *Handling multiple objects in the presence of occlusions and cluttered backgrounds.* Most of the state-of-the-art techniques deal with images that contain a single object. In-the-wild images, however, contain multiple objects of different categories. Previous works employ detection followed by reconstruction within regions of interests, *e.g.*, [138]. The detection and then reconstruction modules operate independently from each other. However, these tasks are inter-related and can benefit from each other if solved jointly. Towards this goal, two important issues should be addressed. The first one is the lack of training data for multiple-object reconstruction. Second, designing appropriate CNN architectures, loss functions, and learning methods are important especially for methods that are trained without 3D supervision. These, in general, use silhouette-based loss functions, which require accurate object-level segmentation.

(6) *3D video.* This paper focused on 3D reconstruction from one or multiple images, but with no temporal correlation. There is, however, a growing interest in 3D video, *i.e.*, 3D reconstruction of entire video streams where successive frames are temporally correlated. On one hand, the availability of a sequence of frames can improve the reconstruction, since one can exploit the additional information available in subsequent frames to disambiguate and refine the reconstruction at the current frame. On the other hand, the reconstruction should be smooth and consistent across frames.

(7) *Towards full 3D scene parsing.* Finally, the ultimate goal is to be able to semantically parse a full 3D scene from one or

multiple of its images. This requires joint detection, recognition, and reconstruction. It would also require capturing and modeling spatial relationships and interactions between objects and between object parts. While there have been a few attempts in the past to address this problem, they are mostly limited to indoor scenes with strong assumptions about the geometry and locations of the objects that compose the scene.

## 12 SUMMARY AND CONCLUDING REMARKS

This paper provides a comprehensive survey of the past five years developments in the field of image-based 3D object reconstruction using deep learning techniques. We classified the state-of-the-art into volumetric, surface-based, and point-based techniques. We then discussed methods in each category based on their input, the network architectures, and the training mechanisms they use. We have also discussed and compared the performance of some key methods.

This survey focused on methods that define 3D reconstruction as the problem of recovering the 3D geometry of objects from one or multiple RGB images. There are, however, many other related problems that share similar solutions. The closest topics include depth reconstruction from RGB images, which has been recently addressed using deep learning techniques, see the recent survey of Laga [153], 3D shape completion [26], [28], [45], [103], [156], [160], [161], 3D reconstruction from depth images [103], which can be seen as a 3D fusion and completion problem, 3D reconstruction and modelling from hand-drawn 2D sketches [162], [163], novel view synthesis [164], [165], and 3D shape structure recovery [10], [29], [83], [96]. These topics have been extensively investigated in the past five years and require separate survey papers.

## ACKNOWLEDGEMENTS

Xian-Feng Han is supported by a China Scholarship Council (CSC) scholarship. This work was supported in part by ARC DP150100294 and DP150104251.

## REFERENCES

- [1] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- [2] A. Laurentini, "The visual hull concept for silhouette-based image understanding," *IEEE TPAMI*, vol. 16, no. 2, pp. 150–162, 1994.
- [3] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D shapenets: A deep representation for volumetric shapes," in *IEEE CVPR*, 2015, pp. 1912–1920.
- [4] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, "Perspective Transformer Nets: Learning single-view 3D object reconstruction without 3D supervision," in *NIPS*, 2016, pp. 1696–1704.
- [5] E. Grant, P. Kohli, and M. van Gerven, "Deep disentangled representations for volumetric reconstruction," in *ECCV*, 2016, pp. 266–279.
- [6] J. Wu, Y. Wang, T. Xue, X. Sun, B. Freeman, and J. Tenenbaum, "MarrNet: 3D shape reconstruction via 2.5D sketches," in *NIPS*, 2017, pp. 540–550.
- [7] 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 *ECCV*, 2016, pp. 628–644.
- [8] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik, "Multi-view supervision for single-view reconstruction via differentiable ray consistency," in *IEEE CVPR*, vol. 1, no. 2, 2017, p. 3.
- [9] X. Z. Xingyuan Sun, Jiajun Wu and Z. Zhang, "Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling," in *IEEE CVPR*, 2018.
- [10] O. Wiles and A. Zisserman, "SilNet: Single-and Multi-View Reconstruction by Learning from Silhouettes," *BMVC*, 2017.
- [11] S. Tulsiani, A. A. Efros, and J. Malik, "Multi-View Consistency as Supervisory Signal for Learning Shape and Pose Prediction," in *IEEE CVPR*, 2018.
- [12] A. Johnston, R. Garg, G. Carneiro, I. Reid, and A. van den Hengel, "Scaling CNNs for High Resolution Volumetric Reconstruction From a Single Image," in *IEEE CVPR*, 2017, pp. 939–948.
- [13] G. Yang, Y. Cui, S. Belongie, and B. Hariharan, "Learning single-view 3d reconstruction with limited pose supervision," in *ECCV*, 2018.
- [14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016, pp. 770–778.
- [15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [16] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *ICLR*, 2014.
- [17] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *NIPS*, 2016, pp. 82–90.
- [18] S. Liu, C. L. Giles, I. Ororbia, and G. Alexander, "Learning a Hierarchical Latent-Variable Model of 3D Shapes," *International Conference on 3D Vision*, 2018.
- [19] A. A. Soltani, H. Huang, J. Wu, T. D. Kulkarni, and J. B. Tenenbaum, "Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks," in *IEEE CVPR*, 2017, pp. 1511–1519.
- [20] P. Henderson and V. Ferrari, "Learning to generate and reconstruct 3D meshes with only 2D supervision," *BMVC*, 2018.
- [21] P. Mandikal, N. Murthy, M. Agarwal, and R. V. Babu, "3D-LMNet: Latent Embedding Matching for Accurate and Diverse 3D Point Cloud Reconstruction from a Single Image," *BMVC*, pp. 662–674, 2018.
- [22] M. Gadelha, R. Wang, and S. Maji, "Multiresolution tree networks for 3D point cloud processing," in *ECCV*, 2018, pp. 103–118.
- [23] H. Laga, Y. Guo, H. Tabia, R. B. Fisher, and M. Bennamoun, *3D Shape Analysis: Fundamentals, Theory, and Applications*. John Wiley & Sons, 2019.
- [24] R. Zhu, H. K. Galoogahi, C. Wang, and S. Lucey, "Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image," in *IEEE ICCV*, 2017, pp. 57–65.
- [25] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," in *ECCV*, 2016, pp. 484–499.
- [26] A. Dai, C. Ruizhongtai Qi, and M. Nießner, "Shape completion using 3D-encoder-predictor CNNs and shape synthesis," in *IEEE CVPR*, 2017, pp. 5868–5877.
- [27] M. Gadelha, S. Maji, and R. Wang, "3D shape induction from 2D views of multiple objects," in *3D Vision*, 2017, pp. 402–411.
- [28] W. Wang, Q. Huang, S. You, C. Yang, and U. Neumann, "Shape inpainting using 3D generative adversarial network and recurrent convolutional networks," *ICCV*, 2017.
- [29] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem, "3D-PRNN: Generating shape primitives with recurrent neural networks," in *IEEE ICCV*, 2017.
- [30] V. A. Knyaz, V. V. Kniaz, and F. Remondino, "Image-to-Voxel Model Translation with Conditional Adversarial Networks," in *ECCV*, 2018, pp. 0–0.
- [31] A. Kundu, Y. Li, and J. M. Rehg, "3D-RCNN: Instance-Level 3D Object Reconstruction via Render-and-Compare," in *IEEE CVPR*, 2018, pp. 3559–3568.
- [32] 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 TOG*, vol. 36, no. 4, p. 72, 2017.
- [33] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs," in *IEEE CVPR*, 2017, pp. 2088–2096.
- [34] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong, "Adaptive O-CNN: a patch-based deep representation of 3D shapes," *ACM ToG*, p. 217, 2018.
- [35] Y.-P. Cao, Z.-N. Liu, Z.-F. Kuang, L. Kobbelt, and S.-M. Hu, "Learning to reconstruct high-quality 3D shapes with cascaded fully convolutional networks," in *ECCV*, 2018.

- [36] C. Hane, S. Tulsiani, and J. Malik, "Hierarchical Surface Prediction," *IEEE PAMI*, no. 1, pp. 1–1, 2019.
- [37] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," *CUMINCAD*, 1996.
- [38] I. Cherabier, J. L. Schonberger, M. R. Oswald, M. Pollefeys, and A. Geiger, "Learning Priors for Semantic 3D Reconstruction," in *ECCV*, 2018.
- [39] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation," in *IEEE CVPR*, 2019, pp. 165–174.
- [40] E. Smith and D. Meger, "Improved Adversarial Systems for 3D Object Generation and Reconstruction," *arXiv:1707.09557*, 2017.
- [41] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *IEEE CVPR*, vol. 3, 2017.
- [42] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, and L. Guibas, "GRASS: Generative Recursive Autoencoders for Shape Structures," *ACM TOG*, vol. 36, no. 4, p. 52, 2017.
- [43] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *IEEE CVPR*, 2019, pp. 5939–5948.
- [44] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox, "What do single-view 3d reconstruction networks learn?" in *IEEE CVPR*, June 2019.
- [45] X. Han, Z. Li, H. Huang, E. Kalogerakis, and Y. Yu, "High-resolution shape completion using deep neural networks for global structure and local geometry inference," in *IEEE CVPR*, 2017, pp. 85–93.
- [46] B. Yang, S. Rosa, A. Markham, N. Trigoni, and H. Wen, "Dense 3D object reconstruction from a single depth view," *IEEE PAMI*, 2018.
- [47] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *IEEE CVPR*, 2015, pp. 2625–2634.
- [48] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biologically-motivated edge segmentation," in *MICCAI*, 2015, pp. 234–241.
- [49] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," vol. 21, no. 4, pp. 163–169, 1987.
- [50] Y. Liao, S. Donné, and A. Geiger, "Deep Marching Cubes: Learning Explicit Surface Representations," in *IEEE CVPR*, 2018, pp. 2916–2925.
- [51] S. R. Richter and S. Roth, "Matryoshka Networks: Predicting 3D Geometry via Nested Shape Layers," in *IEEE CVPR*, 2018.
- [52] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani, "SurfNet: Generating 3D shape surfaces using deep residual networks," in *IEEE CVPR*, vol. 1, no. 2, 2017.
- [53] A. Pumarola, A. Agudo, L. Porzi, A. Sanfeliu, V. Lepetit, and F. Moreno-Noguer, "Geometry-Aware Network for Non-Rigid Shape Prediction From a Single View," in *IEEE CVPR*, June 2018.
- [54] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "AtlasNet: A papier-Mâche Approach to Learning 3D Surface Generation," in *IEEE CVPR*, 2018.
- [55] H. Kato, Y. Ushiku, and T. Harada, "Neural 3D Mesh Renderer," in *IEEE CVPR*, 2018.
- [56] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images," in *ECCV*, 2018.
- [57] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik, "Learning Category-Specific Mesh Reconstruction from Image Collections," *ECCV*, 2018.
- [58] D. Jack, J. K. Pontes, S. Sridharan, C. Fookes, S. Shirazi, F. Maire, and A. Eriksson, "Learning free-form deformations for 3D object reconstruction," *ACCV*, 2018.
- [59] A. Kurenkov, J. Ji, A. Garg, V. Mehta, J. Gwak, C. Choy, and S. Savarese, "DeformNet: Free-Form Deformation Network for 3D Shape Reconstruction from a Single Image," *IEEE WACV*, 2018.
- [60] J. K. Pontes, C. Kong, S. Sridharan, S. Lucey, A. Eriksson, and C. Fookes, "Image2Mesh: A Learning Framework for Single Image 3D Reconstruction," *ACCV*, 2018.
- [61] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *CVPR*, vol. 1, no. 2, 2017, p. 3.
- [62] C. Gotsman, X. Gu, and A. Sheffer, "Fundamentals of spherical parameterization for 3d meshes," *ACM TOG*, vol. 22, no. 3, pp. 358–363, 2003.
- [63] E. Praun and H. Hoppe, "Spherical parametrization and remeshing," *ACM TOG*, vol. 22, no. 3, pp. 340–349, 2003.
- [64] A. Sheffer, E. Praun, K. Rose *et al.*, "Mesh parameterization methods and their applications," *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 2, pp. 105–171, 2007.
- [65] H. Laga, Q. Xie, I. H. Jermyn, A. Srivastava *et al.*, "Numerical inversion of srfn maps for elastic shape analysis of genus-zero surfaces," *IEEE PAMI*, vol. 39, no. 12, pp. 2451–2464, 2017.
- [66] G. Wang, H. Laga, N. Xie, J. Jia, and H. Tabia, "The shape space of 3d botanical tree models," *ACM TOG*, vol. 37, no. 1, p. 7, 2018.
- [67] G. Wang, H. Laga, J. Jia, N. Xie, and H. Tabia, "Statistical modeling of the 3d geometry and topology of botanical trees," *CGF*, vol. 37, no. 5, pp. 185–198, 2018.
- [68] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Siggraph*, 1999, pp. 187–194.
- [69] S. Vicente, J. Carreira, L. Agapito, and J. Batista, "Reconstructing PASCAL VOC," in *IEEE CVPR*, 2014, pp. 41–48.
- [70] S. Tulsiani, A. Kar, J. Carreira, and J. Malik, "Learning category-specific deformable 3D models for object reconstruction," *IEEE PAMI*, vol. 39, no. 4, pp. 719–731, 2017.
- [71] J. K. Pontes, C. Kong, A. Eriksson, C. Fookes, S. Sridharan, and S. Lucey, "Compact model representation for 3D reconstruction," *3DV*, 2017.
- [72] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *IEEE CVPR*, vol. 38, 2017.
- [73] C.-H. Lin, C. Kong, and S. Lucey, "Learning Efficient Point Cloud Generation for Dense 3D Object Reconstruction," *AAAI*, 2018.
- [74] L. Jiang, S. Shi, X. Qi, and J. Jia, "GAL: Geometric Adversarial Loss for Single-View 3D-Object Reconstruction," in *ECCV*, 2018.
- [75] C.-L. Li, M. Zaheer, Y. Zhang, B. Poczos, and R. Salakhutdinov, "Point cloud GAN," *ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2019.
- [76] Y. Sun, Y. Wang, Z. Liu, J. E. Siegel, and S. E. Sarma, "Point-Grow: Autoregressively learned point cloud generation with self-attention," *arXiv:1810.05591*, 2018.
- [77] E. Insafutdinov and A. Dosovitskiy, "Unsupervised learning of shape and pose with differentiable point clouds," in *NIPS*, 2018, pp. 2802–2812.
- [78] K. Li, T. Pham, H. Zhan, and I. Reid, "Efficient dense point cloud object reconstruction using deformation vector fields," in *ECCV*, 2018, pp. 497–513.
- [79] K. Li, R. Garg, M. Cai, and I. Reid, "Single-view object shape reconstruction using deep shape prior and silhouette," *arXiv:1811.11921*, 2019.
- [80] W. Zeng, S. Karaoglu, and T. Gevers, "Inferring Point Clouds from Single Monocular Images by Depth Intermediation," *arXiv:1812.01402*, 2018.
- [81] P. Mandikal and V. B. Radhakrishnan, "Dense 3D Point Cloud Reconstruction Using a Deep Pyramid Network," in *IEEE WACV*, 2019, pp. 1052–1060.
- [82] J. Wang, B. Sun, and Y. Lu, "MVPNet: Multi-View Point Regression Networks for 3D Object Reconstruction from A Single Image," *arXiv:1811.09410*, 2018.
- [83] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Multi-view 3D models from single images with a convolutional network," in *ECCV*, 2016, pp. 322–337.
- [84] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–607.
- [85] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.
- [86] H. Xie, H. Yao, X. Sun, S. Zhou, S. Zhang, and X. Tong, "Pix2Vox: Context-aware 3D Reconstruction from Single and Multi-view Images," *IEEE ICCV*, 2019.
- [87] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3d classification and segmentation," in *IEEE CVPR*, 2017, pp. 652–660.
- [88] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *NIPS*, 2017, pp. 5099–5108.

- [89] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM TOG*, vol. 32, no. 3, p. 29, 2013.
- [90] F. Calakli and G. Taubin, "Ssd: Smooth signed distance surface reconstruction," *CGF*, vol. 30, no. 7, pp. 1993–2002, 2011.
- [91] 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 *ECCV*, 2018.
- [92] X. Zhang, Z. Zhang, C. Zhang, J. Tenenbaum, B. Freeman, and J. Wu, "Learning to reconstruct shapes from unseen classes," in *NIPS*, 2018, pp. 2257–2268.
- [93] E. Smith, S. Fujimoto, and D. Meger, "Multi-view silhouette and depth decomposition for high resolution 3D object representation," in *NIPS*, 2018, pp. 6478–6488.
- [94] M. M. Loper and M. J. Black, "Opendr: An approximate differentiable renderer," in *ECCV*, 2014, pp. 154–169.
- [95] F. Petersen, A. H. Bermano, O. Deussen, and D. Cohen-Or, "Pix2Vex: Image-to-Geometry Reconstruction using a Smooth Differentiable Renderer," *arXiv:1903.11149*, 2019.
- [96] D. t. Rezende, S. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess, "Unsupervised learning of 3D structure from images," in *NIPS*, 2016, pp. 4996–5004.
- [97] J. Gwak, C. B. Choy, A. Garg, M. Chandraker, and S. Savarese, "Weakly Supervised Generative Adversarial Networks for 3D Reconstruction," *3D Vision*, 2017.
- [98] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-DOF camera relocalization," in *IEEE ICCV*, 2015, pp. 2938–2946.
- [99] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views," in *IEEE ICCV*, 2015, pp. 2686–2694.
- [100] D. Novotny, D. Larlus, and A. Vedaldi, "Capturing the geometry of object categories from video supervision," *IEEE PAMI*, 2018.
- [101] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *IEEE CVPR*, 2016, pp. 4104–4113.
- [102] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *NIPS*, 2014, pp. 2672–2680.
- [103] B. Yang, H. Wen, S. Wang, R. Clark, A. Markham, and N. Trigoni, "3D object reconstruction from a single depth view with adversarial learning," in *IEEE ICCV Workshops*, 2017, pp. 679–688.
- [104] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *ICML*, 2017.
- [105] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *NIPS*, 2017, pp. 5767–5777.
- [106] X. Li, Y. Dong, P. Peers, and X. Tong, "Synthesizing 3d shapes from silhouette image collections using multi-projection generative adversarial networks," in *IEEE CVPR*, June 2019.
- [107] P. Mandikal, N. KL, and R. Venkatesh Babu, "3D-PSRNet: Part segmented 3D point cloud reconstruction from a single image," in *ECCV*, 2018, pp. 0–0.
- [108] E. Dibra, H. Jain, C. Öztireli, R. Ziegler, and M. Gross, "Hs-nets: Estimating human body shape from silhouettes with convolutional neural networks," in *3D Vision*, 2016, pp. 108–117.
- [109] E. Dibra, H. Jain, C. Öztireli, R. Ziegler, and M. Gross, "Human shape from silhouettes using generative hks descriptors and cross-modal neural networks," in *IEEE CVPR (CVPR), Honolulu, HI, USA*, vol. 5, 2017.
- [110] T. Alldieck, M. Magnor, B. L. Bhatnagar, C. Theobalt, and G. Pons-Moll, "Learning to Reconstruct People in Clothing from a Single RGB Camera," in *IEEE CVPR*, 2019.
- [111] B. L. Bhatnagar, G. Tiwari, C. Theobalt, and G. Pons-Moll, "Multi-Garment Net: Learning to Dress 3D People from Images," in *IEEE ICCV*, 2019.
- [112] B. Allen, B. Curless, and Z. Popović, "The space of human body shapes: reconstruction and parameterization from range scans," *ACM TOG*, vol. 22, no. 3, pp. 587–594, 2003.
- [113] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: shape completion and animation of people," *ACM TOG*, vol. 24, no. 3, pp. 408–416, 2005.
- [114] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," *ACM TOG*, vol. 34, no. 6, p. 248, 2015.
- [115] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," *CGF*, vol. 28, no. 5, pp. 1383–1392, 2009.
- [116] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, "Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image," in *ECCV*, 2016, pp. 561–578.
- [117] M. Omran, C. Lassner, G. Pons-Moll, P. Gehler, and B. Schiele, "Neural Body Fitting: Unifying Deep Learning and Model Based Human Pose and Shape Estimation," in *3DV*, 2018.
- [118] T. Alldieck, G. Pons-Moll, C. Theobalt, and M. Magnor, "Tex2Shape: Detailed Full Human Body Geometry from a Single Image," in *IEEE ICCV*, 2019.
- [119] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," in *IEEE CVPR*, 2016, pp. 4929–4937.
- [120] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," *CVPR*, 2018.
- [121] Z. Huang, T. Li, W. Chen, Y. Zhao, J. Xing, C. LeGendre, L. Luo, C. Ma, and H. Li, "Deep volumetric video from very sparse multi-view performance capture," in *ECCV*, 2018, pp. 336–354.
- [122] G. Varol, D. Ceylan, B. Russell, J. Yang, E. Yumer, I. Laptev, and C. Schmid, "BodyNet: Volumetric Inference of 3D Human Body Shapes," in *ECCV*, 2018.
- [123] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE PAMI*, no. 6, pp. 681–685, 2001.
- [124] T. Gerig, A. Morel-Forster, C. Blumer, B. Egger, M. Luthi, S. Schönborn, and T. Vetter, "Morphable face models—an open framework," in *IEEE FG*, 2018, pp. 75–82.
- [125] O. M. Parkhi, A. Vedaldi, A. Zisserman *et al.*, "Deep face recognition," in *BMVC*, vol. 1, no. 3, 2015, p. 6.
- [126] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *IEEE CVPR*, 2015, pp. 815–823.
- [127] A. T. Tran, T. Hassner, I. Masi, and G. Medioni, "Regressing robust and discriminative 3D morphable models with a very deep neural network," in *IEEE CVPR*, 2017, pp. 1493–1502.
- [128] E. Richardson, M. Sela, and R. Kimmel, "3D face reconstruction by learning from synthetic data," in *3D Vision*, 2016, pp. 460–469.
- [129] E. Richardson, M. Sela, R. Or-El, and R. Kimmel, "Learning detailed face reconstruction from a single image," *CoRR*, vol. abs/1611.05053, 2016. [Online]. Available: <http://arxiv.org/abs/1611.05053>
- [130] K. Genova, F. Cole, A. Maschinot, A. Sarna, D. Vlasic, and W. T. Freeman, "Unsupervised Training for 3D Morphable Model Regression," in *IEEE CVPR*, 2018.
- [131] A. Tewari, M. Zollhofer, H. Kim, P. Garrido, F. Bernard, P. Perez, and C. Theobalt, "Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction," in *IEEE CVPR*, 2017, pp. 1274–1283.
- [132] A. S. Jackson, A. Bulat, V. Argyriou, and G. Tzimiropoulos, "Large pose 3d face reconstruction from a single image via direct volumetric cnn regression," in *IEEE CVPR*, 2017, pp. 1031–1039.
- [133] M. Sela, E. Richardson, and R. Kimmel, "Unrestricted facial geometry reconstruction using image-to-image translation," in *IEEE CVPR*, 2017, pp. 1576–1585.
- [134] M. Feng, S. Zulqarnain Gilani, Y. Wang, and A. Mian, "3d face reconstruction from light field images: A model-free approach," in *ECCV*, 2018, pp. 501–518.
- [135] J. D'Errico, "Surface fitting using gridfit," *MATLAB central file exchange*, vol. 643, 2005.
- [136] H. Izadinia, Q. Shan, and S. M. Seitz, "Im2cad," in *IEEE CVPR*, 2017, pp. 5134–5143.
- [137] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *NIPS*, 2015, pp. 91–99.
- [138] S. Tulsiani, S. Gupta, D. F. Fouhey, A. A. Efros, and J. Malik, "Factoring shape, pose, and layout from the 2D image of a 3D scene," in *IEEE CVPR*, 2018, pp. 302–310.
- [139] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3D model repository," *arXiv:1512.03012*, 2015.
- [140] J. J. Lim, H. Pirsiavash, and A. Torralba, "Parsing ikea objects: Fine pose estimation," in *IEEE ICCV*, 2013, pp. 2992–2999.
- [141] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond pascal: A benchmark for 3D object detection in the wild," in *IEEE WACV*, 2014, pp. 75–82.
- [142] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese, "ObjectNet3D: A large scale database for 3D object recognition," in *ECCV*, 2016, pp. 160–176.

- [143] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE CVPR*, 2012, pp. 3354–3361.
- [144] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *IEEE CVPR*, 2017.
- [145] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D object representations for fine-grained categorization," in *IEEE CVPR Workshops*, 2013, pp. 554–561.
- [146] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD Birds 200," no. Technical Report: CNS-TR-2010-001.
- [147] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," no. Technical Report: CNS-TR-2011-001, 2011.
- [148] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *IEEE CVPR*, 2017, pp. 1746–1754.
- [149] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3d semantic parsing of large-scale indoor spaces," in *IEEE CVPR*, 2016.
- [150] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2d-3d-semantic data for indoor scene understanding," *arXiv preprint arXiv:1702.01105*, 2017.
- [151] H. Riemenschneider, A. Bódis-Szomorú, J. Weissenberg, and L. Van Gool, "Learning where to classify in multi-view semantic segmentation," in *ECCV*, 2014, pp. 516–532.
- [152] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," *ECCV*, pp. 746–760, 2012.
- [153] H. Laga, "A survey on deep learning architectures for image-based depth reconstruction," *Under Review (paper available on ArXiv)*, 2019.
- [154] I. R. Ward, H. Laga, and M. Bennamoun, "RGB-D image-based Object Detection: from Traditional Methods to Deep Learning Techniques," *arXiv preprint arXiv:1907.09236*, 2019.
- [155] H. Kato and T. Harada, "Learning view priors for single-view 3d reconstruction," *IEEE CVPR*, 2019.
- [156] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *IEEE/RSJ IROS*, 2017, pp. 2442–2447.
- [157] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy Networks: Learning 3D Reconstruction in Function Space," *IEEE CVPR*, 2019.
- [158] S. Kurtek, A. Srivastava, E. Klassen, and H. Laga, "Landmark-guided elastic shape analysis of spherically-parameterized surfaces," vol. 32, no. 2pt4, pp. 429–438, 2013.
- [159] H. Laga, "A survey on nonrigid 3d shape analysis," in *Academic Press Library in Signal Processing, Volume 6*. Elsevier, 2018, pp. 261–304.
- [160] J. Zelek and N. Lunscher, "Point cloud completion of foot shape from a single depth map for fit matching using deep learning view synthesis," in *IEEE ICCV Workshop*, 2017, pp. 2300–2305.
- [161] O. Litany, A. Bronstein, M. Bronstein, and A. Makadia, "Deformable shape completion with graph convolutional autoencoders," *arXiv:1712.00268*, 2017.
- [162] Z. Lun, M. Gadelha, E. Kalogerakis, S. Maji, and R. Wang, "3D Shape Reconstruction from Sketches via Multi-view Convolutional Networks," *3D Vision*, 2017.
- [163] J. Delanoy, M. Aubry, P. Isola, A. A. Efros, and A. Bousseau, "3D Sketching using Multi-View Deep Volumetric Prediction," *ACM ToG*, vol. 1, no. 1, p. 21, 2018.
- [164] C. Li, M. Z. Zia, Q.-H. Tran, X. Yu, G. D. Hager, and M. Chandraker, "Deep supervision with shape concepts for occlusion-aware 3D object parsing," in *IEEE CVPR*, vol. 1, 2017.
- [165] C. Niu, J. Li, and K. Xu, "Im2Struct: Recovering 3D Shape Structure from a Single RGB Image," *IEEE CVPR*, vol. 4096, p. 80, 2018.