doi:10.20944/preprints202208.0067.v1

Review

3D Reconstruction from a Single RGB Image using Deep Learning: A Review

Muhammad Saif Ullah Khan [1,3], Alain Pagani [3], Marcus Liwicki [4], Didier Stricker [1,3] and Muhammad Zeshan Afzal [1,2,3]

[1] Department of Computer Science, Technical University of Kaiserslautern, 67663 Kaiserslautern, Germany; khanm@rhrk.uni-kl.de (M.S.U.K);
[2] Mindgarage, Technical University of Kaiserslautern, 67663 Kaiserslautern, Germany; muhammad_zeshan.afzal@dfki.de (M.Z.A);
[3] German Research Institute for Artificial Intelligence (DFKI), 67663 Kaiserslautern, Germany; alain.pagani@dfki.de (A.P.); didier.stricker@dfki.de (D.S.)
[4] Department of Computer Science, Luleå University of Technology, 971 87 Luleå, Sweden; marcus.liwicki@ltu.se (M.L.)
* Correspondence: khanm@rhrk.uni-kl.de (M.S.U.K), muhammad_zeshan.afzal@dfki.de (M.Z.A)

Abstract: 3D reconstruction from a single 2D input is a challenging problem that is trending in literature. Until recently, it was an ill-posed optimization problem, but with the advent of learning-based methods, the performance of 3D reconstruction has also significantly improved. Infinitely many different 3D objects can be projected onto the same 2D plane, which makes the reconstructing task very difficult. It is even more difficult for objects with complex deformations or no textures. This paper serves as a review of recent literature on 3D reconstruction from a single view, with a focus on deep learning methods from 2018 to 2021. Due to the lack of standard datasets or 3D shape representation methods, it is hard to compare all reviewed methods directly. However, this paper reviews different approaches for reconstructing 3d shapes as depth maps, surface normals, point clouds, and meshes; along with various loss functions and metrics used to train and evaluate these methods.

Keywords: deep learning; 3D reconstruction; convolutional neural networks; texture-less surfaces

1. Introduction

3D reconstruction is the task of inferring the geometric structure of a scene from a set of 2D images. Given one or more 2D views of a scene, we want to know the 3D shape and position in space of all the objects in the scene. This can be seen as mapping the 2D points in the image space to 3D points in real-world space, where for each 2D point (x, y) in the image, we want to recover the corresponding 3D point $(\bar{x}, \bar{y}, \bar{z})$ in world coordinates, where $\bar{z}$ is the distance of the point from the camera.

There are two main ways to reconstruct a 3D scene: monocular reconstruction from a single image, or multi-view reconstruction from multiple images taken from different perspectives. The goal of both tasks is to infer the 3D geometry of the scene, but monocular reconstruction can be more difficult because you only have one view of the scene. Multi-view reconstruction is easier because you get more information about the hidden faces of objects from different angles, making it easier to infer their shape and position.

The literature also distinguishes between the reconstruction of a whole scene as opposed to the reconstruction of a single object. A scene is usually made up of a set of objects, and scene reconstruction involves reconstructing not only the geometry of all the objects in the scene but also their relative positions. Reconstruction of a whole scene is generally harder than reconstruction of a single object because it involves dealing with more complex shadows and occlusions. An image can be a projection of infinitely many different shapes, which makes correctly reconstructing the 3D shape from a single image very hard. Reconstruction of the non-visible faces of the object is challenging in particular as the input image often provides no information about their shape. Bautista *et al.* [1] showed that many existing monocular 3D reconstruction networks learn shape priors over

doi:10.20944/preprints202208.0067.v1

training object categories to solve this problem. This makes it difficult for these networks to generalize to unseen object categories or scenes. Tatarchenko *et al.* [2] demonstrated that single-view 3D reconstruction networks do not reason about the 3D geometry of the object from visual shape cues, but rather rely on object recognition to perform the reconstruction.

The reconstruction task is also made more difficult when there are no textures in the scene. Without any distinctive features, it becomes harder to infer the 3D shape of an object from a 2D image. This makes it difficult to create a complete and accurate 3D model of the scene.

3D reconstruction from visual data is a long-standing computer vision problem with real-world applications in fields like robotic surgery, autonomous vehicles, and virtual and augmented reality. In recent years, deep learning has replaced the traditional computer vision algorithms for the problem of 3D reconstruction with promising results. Deep learning networks have been shown to be more robust to noise and variations in input data than traditional methods. Additionally, they are able to learn complex features from data without any human intervention. This makes them well suited for tasks like 3D reconstruction where there is a lot of variability in input images.

In this paper, we review different deep learning-based methods proposed for the task of 3D reconstruction from a single view. We only focus on methods reconstructing individual objects. Our main contributions include:

- Overview of the neural networks proposed for monocular 3D object reconstruction in the last five years, including:
  - Bednarik *et al.* [3] and Patch-Net [4] for reconstruction of depth and normal maps using a real dataset of texture-less surfaces,
  - HDM-Net [5] and IsMo-GAN [6], which reconstruct 3D point clouds from a synthetic dataset of textured surfaces, and
  - Pixel2Mesh [7], Salvi *et al.* [8] and Yuan *et al.*[9] for reconstruction of mesh-based models using a subset of the ShapeNet [10].
- Summary of the major 3D datasets that are used by the discussed neural networks.
- Description of common metrics used to evaluate the 3D reconstruction algorithms.
- Comparison of the performance of these methods using different evaluation metrics.

The remainder of the paper is organized as follows. In Section 2, we present an overview of related literature. Section 3 defines various methods of representing 3D data, such as depth maps, normal maps, point clouds, 3D meshes, and voxels. In Section 4, we introduce the major 3D datasets used by the networks reviewed in this paper. Then, we discuss different deep learning methods proposed recently for 3D reconstruction in Section 5. Section 6 defines the evaluation metrics used by these networks and lists the results of various experiments conducted by these methods on different datasets. We further discuss the experiment results in Section 7 before concluding our findings and providing direction for future work in Section 8.

2. Related Work

The problem of 3D reconstruction from visual data has been well-studied in computer vision literature, but reconstructing 3D geometry from images remained an ill-posed problem before 2015 when researchers started using convolutional neural networks for this task. Figure 1 shows the trend of related publications since then. In this section, we shed some light on existing surveys that have previously reviewed the 3D reconstruction methods in the literature.

Zollhöfer *et al.* [11] published a report in 2018 on the state-of-the-art in monocular 3D reconstruction of faces. The authors mainly focus on optimization-based algorithms for facial reconstruction, but also briefly mention the emerging trend of using learning-based techniques for this task. They conclude that they "expect to see heavy use of techniques based on deep learning in the future". In 2019, Yuniarti and Suciati [12] formally defined 3D reconstruction as a learning problem and showed an exponentially growing interest

doi:10.20944/preprints202208.0067.v1

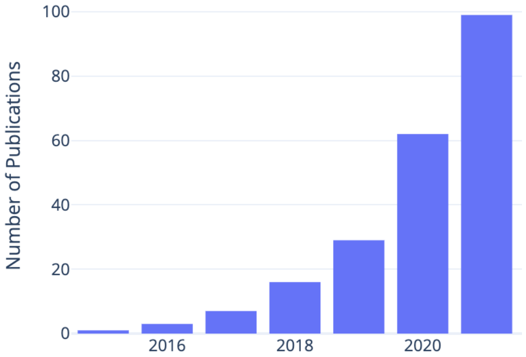


Figure 1. Interest in using deep learning based methods for 3D reconstruction is reflected in the number of publications on ScienceDirect matching the keywords "3d reconstruction" AND "deep learning", which are exponentially growing since 2015.

in 3D reconstruction among the deep learning community. This paper talks about the different ways of representing shapes in 3D, such as parametric models, meshes, and point clouds, lists the 3D datasets available at that time and summarizes various deep learning methods for 3D reconstruction. Han *et al.* [13] published a more extensive review of single and multi-view 3D reconstruction later that year. This work also distinguishes between the reconstruction of scenes and reconstruction of objects in isolation, and reviews techniques for both.

A large body of work in this area focuses solely on producing depth maps from images, which represent a partial representation of 3D geometry. In this context, Laga [14] extensively surveyed more than 100 key contributions using learning-based approaches for recovering depth maps from RGB images. More reviews published in the following years show the shift in trend from using plain CNNs to recurrent neural networks (RNNs), residual networks and generative adversarial networks (GANs) for 3D reconstruction with encouraging results [15,16]. Fu *et al.* [17] also published a review of single-view 3D reconstruction methods, focusing only on objects in isolation. They cover networks proposed between 2016 and 2019 in their review.

3. Representing Shape in 3D

There are many different ways to represent the 3D shape of a scene. The depth and normal maps can be used to represent the partial geometry of the scene which is limited to the surfaces of the objects directly facing the camera. For a more complete representation, point clouds, meshes, and voxels can be used. 3D reconstruction networks can be trained to reconstruct 3D scenes from 2D images in different ways. In this section, we briefly introduce different ways of representing 3D data.

3.1. Depth Map

For each pixel in an image, a depth map provides the distance from the camera of the corresponding point in space. This gives a single-channel image of the same size as the input image, with the corresponding depth value, z at each (x, y) position. The absolute depth values are sometimes mapped to the range [0, 255] and, together with a normal RGB image, the depth map is given as the fourth channel of the so-called RGB-D images with points closer to the camera appearing darker and the points further away appearing brighter.

As depth maps are created from a single viewpoint, they represent a very sparse 3D geometry of the scene containing only points directly in the line of sight of the camera. They say nothing about the occluded planes, nor do they say anything about the 3D orientation of different faces of an object in the scene. For this reason, RGB-D images are sometimes called 2.5D because they cannot represent a complete 3D topology on their own. They are a partial surface model, with only the shape of the front face of the surface represented.

3.2. Normal Map

A normal vector or a "normal" to a surface is a three-dimensional vector perpendicular to the surface at a given point. Analogous to depth maps, normal maps provide normals for each pixel in an image. This means that we can tell the 3D orientation of the surface at any given point in space that is visible in this image.

When RGB-D datasets are combined with normal maps, we can extract both the distance and orientation of every point in a scene from a single viewpoint. However, since we are only seeing the scene from one perspective, information about the hidden surfaces of the objects in the scene cannot be completely represented. Like depth maps, normal maps also represent a partial surface model.

3.3. Point Cloud

A point cloud is a set of 3D points in space. Theoretically, they are able to exactly represent a complete 3D scene by storing the position of every point in space. Depending on how many and which points are present in the point cloud, it can be both a solid and a surface model of the scene. However, due to limited computational memory, it is often necessary to downsample them to reduce the size of the dataset. This can be done by removing the points which are very close to each other or which are not needed to understand the visible shape of the 3D surfaces.

Point clouds can be extremely useful for representing 3D shapes, but they can also be difficult to work with. Sometimes it is necessary to convert them to a mesh in order to get a more accurate representation of the object.

3.4. 3D Mesh

A mesh is a collection of 3D points (or vertices) that have been connected together with edges to form the surfaces (or faces) of 3D objects. Vertices are connected in a way that the faces are made up of many polygons adjacent to each other. Usually, these polygons are triangles, and the meshes are called "triangulated meshes". Meshes can be used to represent the surface models of a 3D scene as "wireframes".

3.5. Voxel

A voxel is a 3D equivalent of a pixel. Voxel-based models represent objects as a collection of cubes (or voxels) stacked in space like a 3D grid. They represent a discretized, solid model of the scene. Accuracy of the 3D model depends on the size of the voxels making up the objects. The bigger the voxels, the more "pixelated" the surfaces of the objects appear.

Like meshes, voxel grids can also be generated directly from point clouds where several adjoining points are all approximated to a single voxel (or a cube) in space. This process is called voxelization and is one way of downsampling the point clouds.

4. Datasets

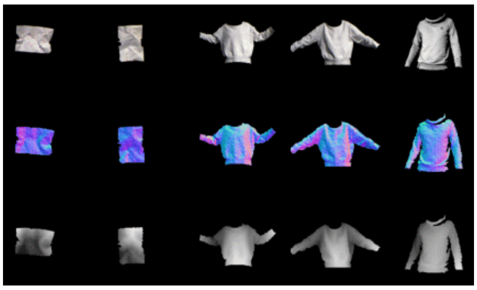
In this section, we introduce the datasets used by the networks discussed later in Section 5. These datasets contain RGB images of different objects and their 3D shape in one of the representations introduced above. Table 1 provides a summary of these datasets.

4.1. Real Texture-less Surfaces Data

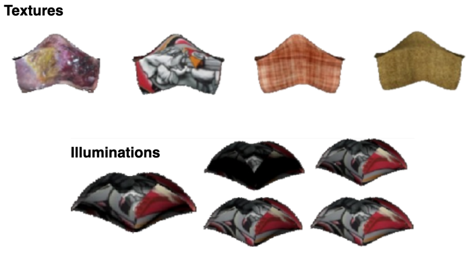
This RGB-D dataset of deformable texture-less surfaces from [3] consists of 26,445 RGB images, along with depth maps and normal maps for each image. These RGB images and depth maps were collected using a Microsoft Kinect for Xbox One device. Normal maps were computed by differentiating the depth maps. The dataset contains five different types of objects: *tshirt*, *hoody*, *sweater*, a rectangular sheet of *cloth*, and a crumpled piece of *paper*. The objects have no texture or colors on them. Figure 2a shows some samples from this dataset.

Table 1. Summary of the major 3D datasets in this paper. Most of the datasets contain textured surfaces and are generated from synthetic 3D models. Real datasets captured directly with 3D sensors are less common and smaller in size because of the difficulty associated with obtaining such that.

Dataset	Type	Surface	Texture	Groundtruth	Size
Bednarik *et al.*	Real	Deformable	No	Depth and Normal Maps	$26,445$
Golyanik *et al.*	Synthetic	Deformable	Yes	Point Clouds	$4,648$
ShapeNet [10]	Synthetic	Mixed	Yes	3D Mesh	$> 300M$
R2N2 [18]	Synthetic	Rigid	Yes	3D Meshes and Voxelized Models	50,000



(a) The texture-less surfaces dataset [3] contains RGB images and corresponding normal and depth maps for 5 different real objects.



(b) The synthetic point cloud dataset of Golyanik *et al.* has a deforming thin plate rendered with 4 different textures under 5 different illuminations.

Figure 2. Examples of images in the datasets of Bednarik *et al.* and Golyanik *et al.*, which are used to evaluate some of the networks in this paper. *(Figures adapted from [3,5])*

The *tshirt*, *hoody* and *sweater* were worn by a person who made random motions to simulate realistic creases. The sheet of *cloth* was fixed to a bar on the wall and manually deformed, and the piece of *paper* was crumpled by hand to create different depths. Different combinations of four light sources were used to create lighting variations across different recording sequences. This included three fixed lights in front of the objects on the right, left, and center, and one moving dynamic light in the room. Table 2 summarizes the number of samples of each kind of object.

Table 2. Summary of objects in the texture-less surfaces dataset [3]. Sequences of data samples were captured using a Kinect device at 5 FPS with varying lighting conditions across sequences.

	cloth	*tshirt*	*sweater*	*hoody*	*paper*
sequences	18	12	4	1	3
samples	15,799	6,739	2,203	517	1,187

4.2. Synthetic 3D Point Cloud Data

Golyanik *et al.* generated a synthetic 3D dataset in point cloud representation. Using Blender [19], they created a 3D scene with a thin plate undergoing various isometric non-linear deformations. Four kinds of textures ($endoscopy, graffiti, clothes$ and $carpet$) were mapped on to the deformed 3D model, which was then illuminated in various settings using five different light sources. The scene was viewed from five separate cameras at different angles. In this way, a total of 4648 states were generated. Each state is represented

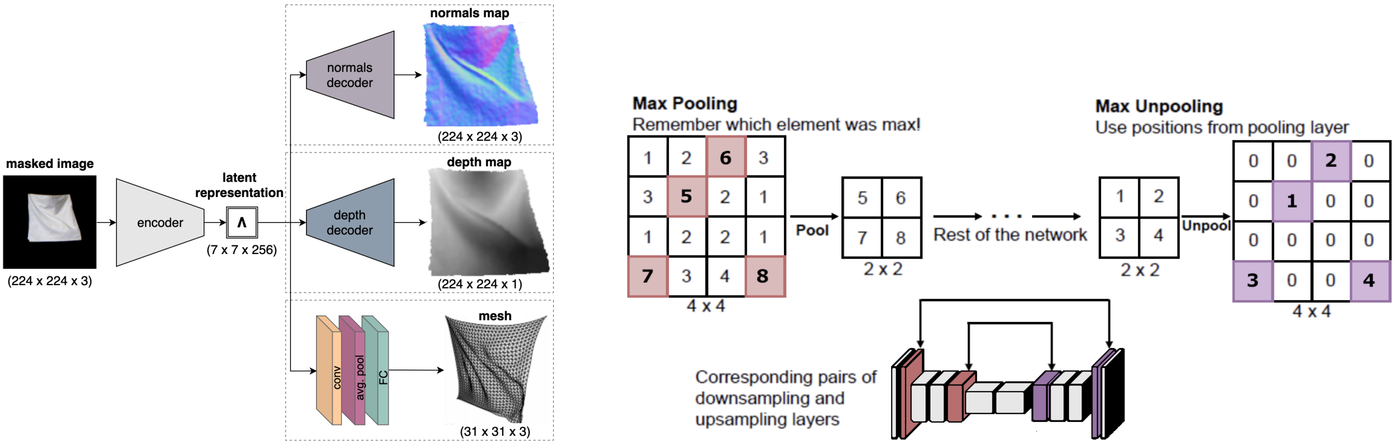


Figure 3. The texture-less surface reconstruction network [3] (left) consists of an encoder $\mathbf{\Lambda}$ that takes a masked image $\mathbf{I}_m^n$ as input and outputs a latent representation Λ. This is followed by three parallel decoders $\mathbf{\Phi}_N$, $\mathbf{\Phi}_D$, and $\mathbf{\Phi}_C$ that use Λ for reconstructing the normal map, depth map, and a 3D mesh respectively. The indices of all maxpool operations in encoder are saved when downsampling (right). These indices are later used for non-linear upsampling in corresponding decoder layers.

with 73^2 3D points sampled on a regular grid at rest and a consistent topology across states. For each state, there is also a corresponding rendered 2D image viewing the object from one of the cameras. Figure 2b shows some samples from this dataset.

4.3. ShapeNet

ShapeNet [10] is a large-scale dataset containing richly-annotated 3D CAD models organized according to the WordNet [20] hierarchy. It contains over 300M models, 220M of which are classified into 3,135 WordNet symsets [21]. ShapeNet is made up of many smaller subsets. A major subset is called ShapeNetCore that contains manually verified single clean 3D models. It has two versions, v1 and v2, covering 55 and 57 categories respectively. Processed meshes and annotations of these models can be downloaded online from the ShapeNet website [22].

4.4. R2N2 Dataset

Choy *et al.* [18] took a subset of the ShapeNetCore v1 dataset containing 50,000 models and 13 categories including $plane$, $bench$, $cabinet$, car, $chair$, $monitor$, $lamp$, $speaker$, $firearm$, $sofa$, $table$, $phone$, and $watercraft$. For each object, R2N2 dataset also makes available its own 24 renderings of the models from different viewpoints in a 137 × 137 resolution, and the 3D models themselves as meshes, point clouds and voxels.

5. Networks

This section introduces some of the recent methods proposed for reconstructing 3D surfaces from a single 2D image. These are summarized in Table 1.

5.1. Bednarik et al.

Bednarik *et al.* [3] introduced a general framework for reconstructing the 3D shape of texture-less surfaces with an encoder-decoder architecture. Using a single RGB image, they reconstruct the normal maps, depth maps, and triangulated meshes for the objects in the images. Figure 3 shows an overview of their architecture. This network has an encoder connected to three separate decoders, one each for reconstructing the normal map, depth map, and the triangulated mesh. The encoder takes an RGB image of size 224 × 224 × 3 and creates a latent representation of size 7 × 7 × 256. This encoding is fed to the three decoders.

The architecture of the encoder and the depth and normal decoders is based on SegNet [23]. The encoder has the same layers as VGG-16 [24] except for the fully-convolutional layers. However, in contrast to VGG-16, the output channels at the convolutional blocks are $32, 64, 128, 256, 264$ respectively. As the normal maps and depth maps have same spatial size as the input image, the normal and depth decoders are symmetric to the encoder with both having the same architecture except the number of channels at the final output layer; the normal decoder has 3 channels and depth decoder has 1 channel. Like SegNet, pooling indices at the max pooling layers in the encoder are saved, and used in the normal and depth decoders to perform non-linear upsampling. For the mesh decoder, a smaller network with a single convolutional layer followed by average pooling and a fully-connected layer is used.

The depth decoder is trained by minimizing the absolute difference between the predicted and groundtruth depth values of the foreground, giving the loss function

$$\mathcal{L}_D = \frac{1}{N}\sum_{n=1}^{N}\frac{\sum_i \left|\mathbf{D}_i^n - \Phi_D(\Lambda(\mathbf{I}_m^n))_i\right|\mathbf{B}_i^n}{\sum_i \mathbf{B}_i^n}, \tag{1}$$

where $\mathbf{D}^n$ is the groundtruth depth map and Φ_D is the depth decoder, which takes the encoder output on the masked input image $\Lambda(\mathbf{I}_m^n)$ and returns the predicted depth map. The absolute difference is only calculated for the foreground pixels, i.e. where the foreground mask $\mathbf{B}^n$ has the value 1, and the sum of absolute differences is averaged over all the foreground pixels.

To train the normal decoder, the angular distance between the predicted and groundtruth normal vectors and the length of the predicted normal vectors are both optimized using the loss function

$$\mathcal{L}_N = \frac{1}{N}\sum_{n=1}^{N}\frac{\sum_i \left(\kappa\mathcal{L}_a\left(\mathbf{N}_i^n, \bar{\mathbf{N}}_i^n\right) + \mathcal{L}_l\left(\bar{\mathbf{N}}_i^n\right)\right)\mathbf{B}_i^n}{\sum_i \mathbf{B}_i^n} \tag{2}$$

with

$$\mathcal{L}_a(\mathbf{N}_i^n, \bar{\mathbf{N}}_i^n) = \arccos\left(\frac{\mathbf{N}_i^n\bar{\mathbf{N}}_i^n}{\|\mathbf{N}_i^n\|\|\bar{\mathbf{N}}_i^n\| + \epsilon}\right)\frac{1}{\pi}, \tag{3}$$

$$\mathcal{L}_l(\bar{\mathbf{N}}_i^n) = \left(\|\bar{\mathbf{N}}_i^n\| - 1\right)^2 \tag{4}$$

where $\mathcal{L}_a$ is the angular distance calculated as the arccos of the cosine similarity between the predicted and groundtruth normal vectors, $\mathcal{L}_l$ is the term that prefers unit normal vectors, and κ is a hyperparameter that sets the relative influence of the two terms. Furthermore, $\mathbf{N}^n$ is the groundtruth normal map, and the $\bar{\mathbf{N}}^n = \Phi_N(\Lambda(\mathbf{I}_m^n))$ is the predicted normal map. As with depth loss, the normal loss is only calculated for foreground pixels.

Finally, for the triangulated mesh prediction, the mesh decoder optimizes the Mean Squared Error between predicted and groundtruth vertex coordinates. That is,

$$\mathcal{L}_C = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{V}\sum_{i=1}^{V}\|\mathbf{v}_i^n - \Phi_C(\Lambda(\mathbf{I}_m^n))\|^2 \tag{5}$$

As all three decoders take input from the same encoder with same latent representation, they can be trained either jointly or separately. When trained jointly, [3] shows the accuracy of the reconstruction improves because the encoder is able to learn more robust feature extractors. The texture-less dataset described in Section 4.1 was used for training and testing this network, and experiments showed poor reconstruction accuracy for 3D meshes compared to normal and depth maps.

The network was trained using the Adam optimizer [25] with a fixed learning rate of 10^{-3}, and $\kappa = 10$. The authors used Keras [26] with a Tensorflow [27] backend for implementation and published the source code. At run-time, the network takes $0.016s$ to predict both depth and normal maps together, and $0.01s$ when predicting either the depth or normal map individually.

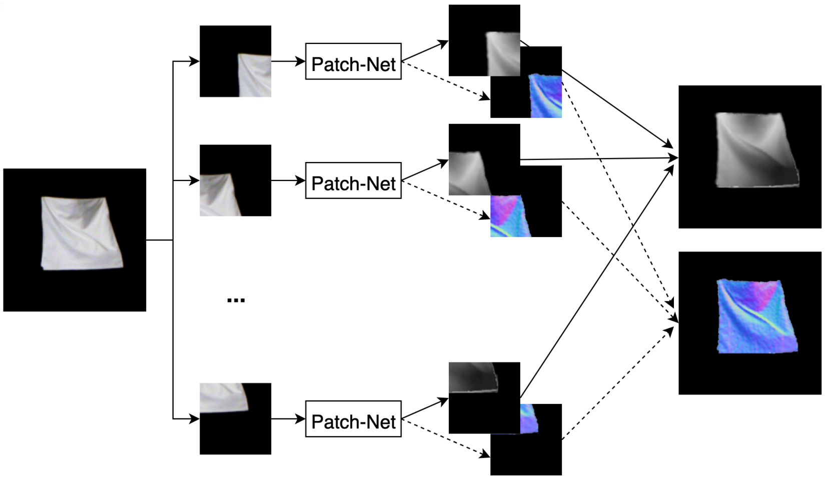


Figure 4. Patch-Net uses Bednarik *et al.*'s network with only depth and normal decoders. The input image is divided into overlapping patches, and predictions for each patch are obtained separately. Patch predictions are stitched to form the complete depth and normal maps.

5.2. Patch-Net

Tsoli and Argyros proposed a patch-based variation for better texture-less reconstruction. They take the network from [3] and change the block sizes to match VGG-16 [24], i.e. 64, 128, 256, 512, 512. They also remove the mesh decoder, keeping only the normal and depth decoders. They divide the input image into overlapping patches and get per patch reconstructions for normal and depth maps. These patches are then stitched together to get the final normal and depth maps at the input image resolution, and use bilateral filtering to smooth out inconsistencies that were not resolved by stitching. They call this network Patch-Net. Since the network expects a 224 × 224 spatial size of the input, each patch can have that size with the full image being even larger. This allows Patch-Net to get a higher resolution reconstruction than [3] with better accuracy and generalization. It uses the loss functions of Eq. 2 and Eq. 1 on each patch to compute the normal and depth loss respectively.

The network was trained using the Adam optimizer with a fixed learning rate of 10^{-3}. The authors extended the source code from [3], and trained their network on a Nvidia Titan V GPU with 12 GB memory. This code is not publicly available, and the authors do not report inference-time performance.

5.3. HDM-Net

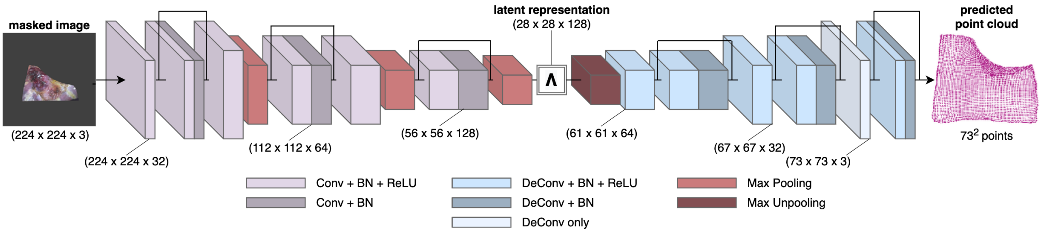


Figure 5. Overview of the HDM-Net [5] architecture. It has an encoder that takes an RGB image of size 224 × 224 × 3 and encodes it into a latent representation of size 28 × 28 × 128. This is then used by the decoder to reconstruct a 3D point cloud of the surface with 73^2 points.

The *Hybrid Deformation Model Network* (HDM-Net) [5] is another approach for reconstructing deformable surfaces from a single-view. Like [3] and Patch-Net, HDM-Net uses an encoder-decoder architecture (Figure 5), but with only one decoder instead. However, the encoder and decoder are not symmetric to each other in this network. They also have a smaller depth, with only 9 convolution layers in the encoder instead of 13 convolution layers in the VGG-16 based architectures. The upsampling in the decoder is performed using transposed convolutions, as in [28], except at the first decoder layer where a non-linear max-unpooling operation similar to [3,4,23] is used. HDM-Net directly learns the 3D shape and give a dense reconstruction of the surface of size 73 × 73 × 3 as a point cloud. It is trained on the synthetic point cloud data (Section 4.2) of a thin non-rigid plate undergoing various non-linear deformations, with a known shape at rest. Three different domain specific loss functions are used to jointly optimize the output of the network, with the goal of learning texture-dependent surface deformations, shading and contours for effective handling of occlusions.

The first loss function is a common 3D regression loss that computes the 3D error by penalizing the difference between the predicted 3D geometry $\mathbf{S}_n$ and groundtruth 3D geometry $\mathbf{S}'_n$, that is,

$$\mathcal{L}_{3D} = \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{S}'_n - \mathbf{S}_n\|^2_{\mathcal{F}} \tag{6}$$

where $\|\cdot\|_{\mathcal{F}}$ is the Frobenius norm. For each state n, the squared Frobenius norm of the difference between predicted and groundtruth geometries is calculated, and then averaged for all N states.

An isometry prior is used to constraint the regression space using an isometric loss that penalizes the roughness in the predicted surface by ensuring that neighboring vertices are located close to each other. The loss function is expressed in terms of the predicted geometry $\mathbf{S}_n$ and its smooth version $\bar{\mathbf{S}}_n$

$$\mathcal{L}_{iso.} = \frac{1}{N}\sum_{n=1}^{N} \|\bar{\mathbf{S}}_n - \mathbf{S}_n\|_{\mathcal{F}} \tag{7}$$

with

$$\bar{\mathbf{S}}_n = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{\sigma^2}\right) * \mathbf{S}_n \tag{8}$$

where $*$ is a convolution operator and σ^2 is the variance of Gaussian, and x and y stand for the point coordinates.

The third loss function optimizes the contour shapes by computing a reprojection loss. The predicted and groundtruth 3D geometries are first projected onto a 2D plane and before computing their difference as

$$\mathbf{L}_{cont.} = \frac{1}{N}\sum_{n=1}^{N} \|\tau(\pi(\mathbf{S}_n)) - \tau(\pi(\mathbf{S}'_n))\|^2_{\mathcal{F}} \tag{9}$$

where π is a differentiable 3D to 2D projection function and τ is a function that thresholds all positive values to 1 using a combination of tanh and ReLU. This gives contours as 0-1 transitions. The total loss is computed by adding all three losses with equal weights.

HDM-Net was trained for 95 epochs on a GEFORCE GTX 1080Ti GPU with 11 GB of global memory. The training relied on the PyTorch framework [29] and took 2 days to complete. At inference time, the network can reconstruct frames with a frequency of $200Hz$, or $0.005s$ per frame. The source code was not published.

5.4. IsMo-GAN

An improved version of HDM-Net is the Isometry-Aware Monocular Generative Adversarial Network (IsMo-GAN) [6] that introduces two key modifications to achieve 10-30% reduction in reconstruction error in different cases, including reconstruction of textureless surfaces. First, IsMo-GAN has an integrated Object Detection Network (OD-Net) that generates a confidence map separating background from the foreground. Secondly, IsMo-GAN is trained in an adversarial setting, which is different from the training of the simple

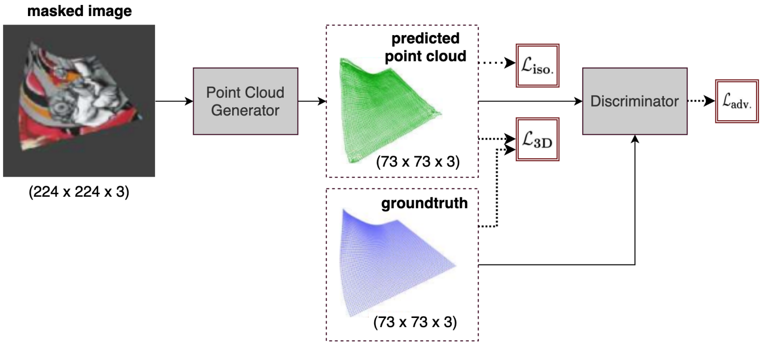


Figure 6. Overview of IsMo-GAN [6]. The generator network accepts a masked RGB image, segmented by the object detection network (OD-Net) and returns a 3D point cloud. The output and groundtruth are fed to the discriminator which serves as a surface regularizer.

auto-encoder based networks discussed in previous sections. The OD-Net is a simplified version of U-Net [28] with fewer layers than the original. It takes a 224 × 224 × 3 RGB image and outputs a grayscale confidence map indicating the position of the foreground in the image. The confidence map is binarized [30] and the target image is extracted using Suzuki *et al.*'s algorithm [31]. The masked-out input image is then passed to the Reconstruction Network (Rec-Net), which has skip connections like HDM-Net and has a similar architecture but with fewer layers. Like HDM-Net, the Rec-Net outputs a 73 × 73 × 3 size point cloud. OD-Net and Rec-Net together make up the generator of IsMo-GAN. The discriminator network consists of four convolution layers followed by a fully-connected layer and a sigmoid function. IsMo-GAN uses the LeakyReLU activation everywhere, instead of ReLU which was used in all other networks discussed previously. Figure 6 shows an overview of IsMo-GAN network.

IsMo-GAN penalizes the output of the Rec-Net with the 3D loss (Eq. 6) and isometric loss (Eq. 7) from HDM-Net, where the predicted geometry is equal to the generator output on the input image, i.e., $\mathbf{S}_n = \mathbf{G}(\mathbf{I})$. In addition to this, for adversarial training, IsMo-GAN uses cross entropy (BCE) [32], defined as

$$\mathcal{L}_{\mathbf{G}} = -\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\log\left(\mathbf{D}(\mathbf{G}(\mathbf{I}_m^n))\right) \tag{10}$$

for the generator $\mathbf{G}$, and

$$\mathcal{L}_{\mathbf{D}} = -\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left[\log\left(\mathbf{D}(\mathbf{S}'_m)\right) + \log\left(1 - \mathbf{D}(\mathbf{G}(\mathbf{I}_m^n))\right)\right] \tag{11}$$

for the discriminator $\mathbf{D}$, where M is the number of states, and N is the number images for each state. The adversarial loss is then defined as the sum of the generator and discriminator losses

$$\mathcal{L}_{adv} = \mathcal{L}_{\mathbf{G}} + \mathcal{L}_{\mathbf{D}}, \tag{12}$$

and it represents the overall objective of the training which encourages IsMo-GAN to generate more realistic surfaces. It is a key component that lets IsMo-GAN outperform HDM-Net [5] by 10-15% quantitatively as well as qualitatively on real images. The adversarial loss makes up for the undesired affects of the 3D loss and the isometry prior by acting as a novel regularizer for the surface deformations. This network is trained and evaluated on the same dataset as HDM-Net, as well as on the 3D mesh data of the *cloth* object from the subset of the Bednarik *et al.*'s real texture-less surfaces dataset.

The OD-Net and Rec-Net were both trained separately for 30 and 130 epochs respectively, using the Adam optimizer with a fixed learning rate of 10^{-3} and a batch size of 8.

IsMo-GAN was implemented using PyTorch, but the source code was not made public. It takes $0.004s$ to run an inference, which is a 20% improvement over HDM-Net.

5.5. Pixel2Mesh

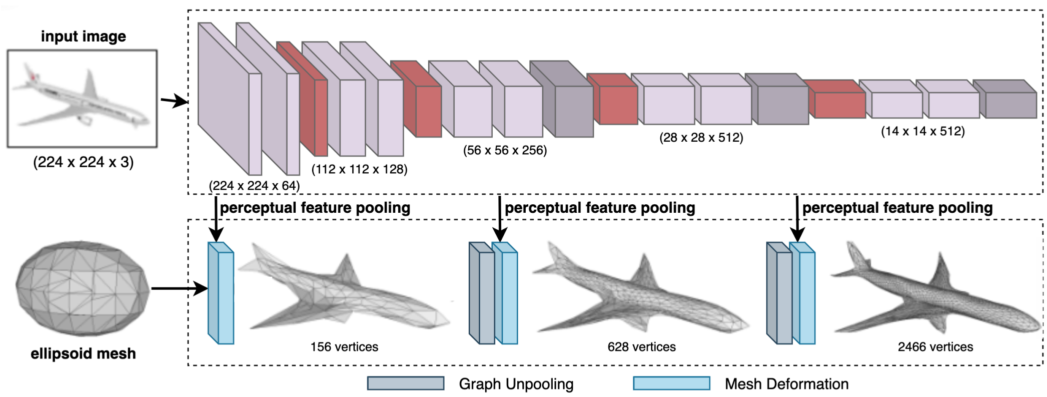


Figure 7. The Pixel2Mesh [7] network consists of two parallel networks that take an RGB image and a coarse ellipsoid 3D mesh, and learn to regress the 3D shape of the object in the image. The key contribution is the graph-based convolutions and unpooling operators in the bottom-half of the network.

Pixel2Mesh [7] is a deep learning network that reconstructs 3D shape as a triangulated mesh from a single RGB image. It was proposed in 2018 and is one of the earliest methods for monocular 3D reconstruction. Its primary idea is to use graph-based convolutions [33] to regress the mesh vertices. The network is made up of two parts; a VGG-16 based feature extractor and a graph-based convolution network (GCN). The feature extractor network takes a 224×224 image to reconstruct. And the GCN takes an ellipsoid mesh with 156 vertices and 462 edges. The feature extractor network then feeds the extracted perceptual features at different stages to the GCN in a cascaded manner, which refines the initial mesh in a coarse-to-fine manner by adding details at each stage. The GCN finally outputs a mesh with 2466 vertices (Figure 7). Each mesh deformation block in the GCN is made of 14 layers of graph based convolutions with ResNet [34] like skip connections. Their job is to optimize the position of existing vertices to get a mesh matching the object shape. This is followed by a graph unpooling layer that interpolates the mesh to increase the number of vertices.

Pixel2Mesh combines four different loss functions to optimize its weights. These include the Chamfer loss [35] to constraint the location of mesh vertices, a normal consistency loss, a Laplacian regularization to maintain the neighborhood relationships when deforming the mesh, and an edge length loss to prevent outliers. The total loss then calculated as a weighted sum of the individual losses. The network is trained and evaluated on the R2N2 subset [18] of the ShapeNet dataset [10], which consists of synthetically rendered images and 3D mesh groundtruth. The network is also qualitatively evaluated on the Stanford Online Products dataset [36], which contains real-world images of objects without any 3D labels.

Pixel2Mesh was implemented using Tensorflow and the official source code is available on GitHub. It used the Adam optimizer with a weight decay of 1^{-5} and a batch size of 1 to train for 50 epochs, with the initial learning rate of 3^{-5}. The training took 72 hours on Nvidia Titan X GPU with 12 GB memory, and the trained network can reconstruct a mesh containing 2466 vertices in 15.58ms.

5.6. Salvi et al.

A new category of networks is adding self-attention modules [37,38] to 3D reconstruction networks. Salvi *et al.* proposed one such network, which improves Occupancy Networks (ONets) [39] by adding self-attention to them. ONets consist of three parts:

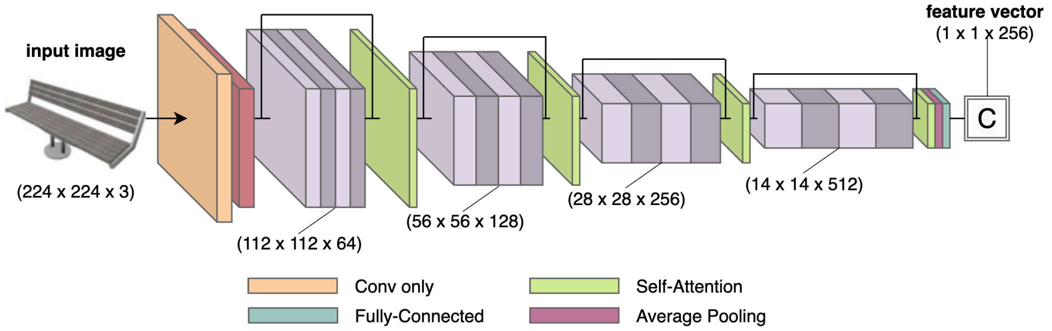


Figure 8. The "attentioned" ResNet-18 [34] network with four self-attention blocks [37] added to it. This encoder network is used by [8] to extract image features, which are fed to a decoder with five Conditional Batch Normalization blocks followed by an occupancy function.

a feature extractor, a decoder, and a continuous decision boundary function, called the occupancy function $o : \mathbb{R} \to \{0, 1\}$, that classifies each point from the space as whether or not it belongs to the surface. This provides a general 3D representation that allows extracting meshes at any resolution. ONets are an extension of the autoencoders discussed in previous sections, where the encoders functioned as feature extractors, followed by a decoder to reconstruct the 3D shape.

In the networks discussed previously as well as ONets introduced in [39], the feature extractors are based on CNNs. Standard CNNs work with local receptive fields and need very deep architectures to successfully model global dependencies. This is because the features they learn are relatively shallow and do not capture the long-range correlations in natural images. To address this limitation, self-attention modules were introduced that calculate the response at a given position as a weighted sum of the features at all positions. This allows them to efficiently model global dependencies with much smaller networks than traditional CNNs. Salvi *et al.* show that adding self-attention modules at different locations in the feature extractor can improve the performance of an Occupancy Network. When used earlier in the network, self-attention allows the network to focus more on finer details. And when used later in the network, it allows the network to extract better structural features. Figure 8 depicts one such feature extractor proposed by [8], showing a ResNet-18 [34] network with four self-attention modules.

They train their network on the synthetic R2N2 dataset [18] (see Section 4.4) using an ensemble approach, where the ensemble is made up of one specialized ONet for each object type. This is supported by their experiments which show that self-attention based ONets have better results if trained for each category separately. The network was also qualitatively evaluated on a subset of the Stanford Online Products dataset [36], which contains real images, and showed more consistent and better reconstruction of meshes when compared to existing approaches. Self-attention in decoders was not used due to computational limitations.

Adam optimizer with a learning rate of 10^{-3} and weight decay of 1^{-5} was used for training the network for $200K$ steps. All other hyperparameters were kept the same as in [39]. The source code for this network is not available.

5.7. VANet

Another network that uses the attention mechanism is the View Attention Guided Network (VANet) [9]. It uses channel-wise view attention and a dual pathway network for better reconstruction of occluded parts of the objects, and defines a unified approach for both single and multi-view reconstruction. As shown in Figure 9, the proposed architecture consists of a main pathway and an auxiliary pathway. The main path uses the first view of a scene to reconstruct a 3D mesh. If any more views are available, they are then fed to the auxiliary path, which aligns them with the main view and uses the additional information from these new views to refine the reconstructed mesh. The main view features after the

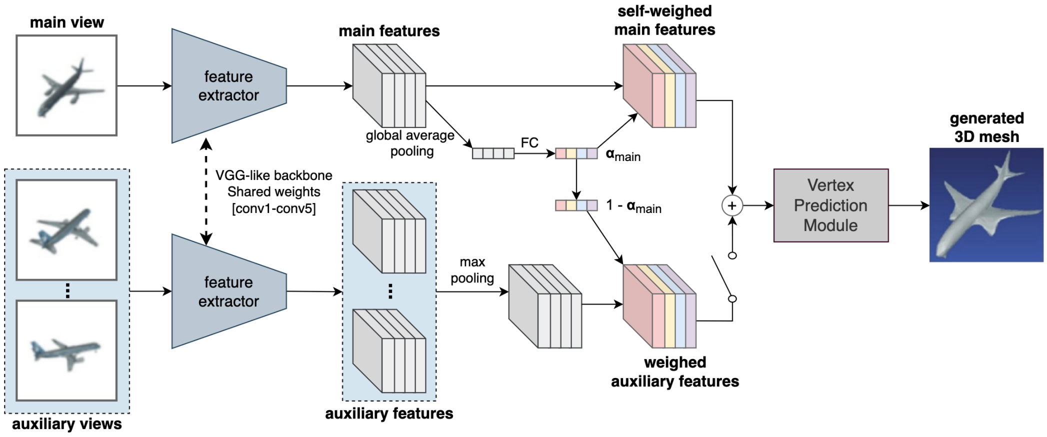


Figure 9. Overview of VANet [9], a unified approach for both single and multi-view reconstruction with a two branch architecture.

encoder are pooled along the spatial dimensions using global average pooling to get a channel descriptor of shape $1 \times 1 \times C$. This is then sent to a system of fully-connected layers followed by a sigmoid function to generate a channel-wise attention map α_{main}. These attention weights are then used to re-calibrate the computed feature maps. If auxiliary views are available, they are used to enhance the less-visible parts in the original view. A max pooling operation is used to select permutation invariant auxiliary view features, which are multiplied with $1 - \alpha_{main}$ and finally added to the main view features. These are then sent to a vertex prediction module to generate the reconstructed 3D mesh. The vertex prediction module is based on the mesh deformation module of Pixel2Mesh [7].

VANet is trained using the same four loss functions as Pixel2Mesh, and evaluated on the R2N2 subset [18] of the ShapeNet dataset [10]. Using the Adam optimizer with an initial learning rate of $2x10^{-5}$ and a batch size of 1, the network was trained for 20 epochs. It was implemented in Tensorflow but the source code was not published.

5.8. 3D-VRVT

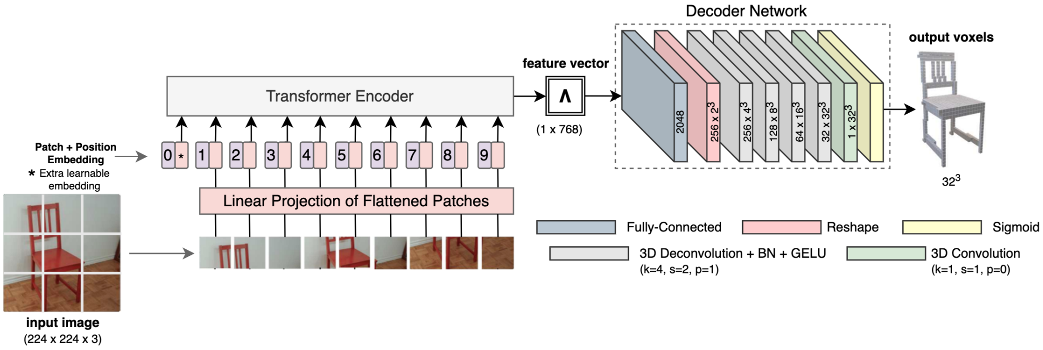


Figure 10. 3D-VRVT takes one image as input and uses a Vision Transformer encoder to extract a feature vector. This is then fed to a decoder that outputs the voxel representation of the object.

Vaswani *et al.* [37] initially proposed the Transformer architecture for natural language processing (NLP) tasks. These methods used the self-attention mechanism to let the network understand longer sequences of text to compute a representation for the whole sequence. Salvi *et al.* [8] used the self-attention mechanism from Transformers in their "attentioned" ResNet encoder to extract better features for 3D reconstruction. However, their input is not sequential (Section 5.6). Dosovitskiy *et al.* [40] proposed a novel architecture called Vision Transformers that breaks down images into patches and treats those patches as part of a sequence. Using a linear projection, vector embeddings for each patch are obtained. This sequence of patch embeddings is then fed to a Transformer network.

Inspired by this, Li and Kuang [41] proposed a Vision Transformer-based network (Figure 10) for reconstructing voxels from a single image. They call this network 3D-VRVT.

3D-VRVT uses a Vision Transformer as an encoder, that takes a $224x224$ RGB image as input and produces a feature vector of size 768 which is fed to a decoder network. The decoder network has a fully-connected layer that upscales the feature vector to 2048 and then reshapes it into a 3D tensor of shape $256x2^3$. This is followed by four 3D deconvolutions with a kernel size of 4, stride 2, and padding 1 that iteratively refine the 3D grid until it has the resolution $32x32^3$. Each deconvolution operation is also followed by a 3D batch normalization and a GELU activation function. Then, a final deconvolution with kernel size 1 is applied to get a grid of $1x32^3$. This is passed through a sigmoid activation function before getting the final voxel output.

The network was trained on the ShapeNet dataset. It used an SGD optimizer and a warm-up cosine annealing learning rate with a momentum of 0.9. The learning rate ranged between 2^{-5} and 2^{-3}. The training relied on a PyTorch implementation and continued for 600 epochs on Nvidia Titan V GPU, including 10 warm-up epochs. At test time, it takes $8.82ms$ to reconstruct an object with this network.

6. Comparison

We described different 3D reconstruction methods in this paper, which were trained by their authors on various datasets and evaluated on different error metrics. In this section, we first formally define the various error metrics used for evaluation of 3D reconstruction methods. We then describe comparable experiments on similar datasets by different networks, and report and compare performance of those methods.

6.1. Metrics

1. *Depth Error* ($\mathcal{E}_D$): The depth error metric is used to compute the accuracy of depth map predictions. Let $\Theta_{\mathbf{K}}$ and $\Theta'_{\mathbf{K}}$ be the point clouds associated with the predicted and groundtruth depth maps respectively, with the camera matrix $\mathbf{K}$. To remove the inherent global scale ambiguity [42] in the prediction, $\Theta_{\mathbf{K}}$ is aligned to groundtruth depth map $\mathbf{D}'$ to get an aligned point cloud $\bar{\Theta}_{\mathbf{K}}$ as

$$\bar{\Theta}_{\mathbf{K}} = \Omega(\Theta_{\mathbf{K}}, \mathbf{D}') \tag{13}$$

where Ω is the Procrustes transformation [43]. Then, the depth error $\mathcal{E}_D$ is calculated as

$$\mathcal{E}_D = \frac{1}{N} \sum_{n=1}^{N} \frac{\sum_i \|\Theta'_{\mathbf{K}} - \bar{\Theta}_{\mathbf{K}}\| \mathbf{B}_i^n}{\sum_i \mathbf{B}_i^n}. \tag{14}$$

Note that the foreground mask $\mathbf{B}$ in the equation ensures that the error is only calculated for foreground pixels. Smaller depth errors are preferred.

2. *Mean Angular Error* ($\mathcal{E}_{MAE}$: The mean angular error $\mathcal{E}_{MAE}$ metric is used to calculate the accuracy of normal maps, by computing the average difference between the predicted and groundtruth normal vectors. The angular errors for all samples are calculated using Eq. 3, and then averaged for all samples. Smaller angular errors indicate better predictions.
3. *Volumetric IoU* ($\mathcal{E}_{IOU}$): The Intersection over Union (IoU) metric for meshes is calculated as the volume of the intersection of groundtruth and predicted meshes, divided by the volume of their union. Larger values are better.
4. *Chamfer Distance* ($\mathcal{E}_{CD}$): Chamfer distance is a measure of similarity between two point clouds. It takes the distance of each point into account by finding, for each point in a point cloud, the nearest point in the other cloud, and summing their squared distances.

$$\mathcal{E}_{CD} = \frac{1}{|\Theta|} \sum_{x \in \Theta} \min_{y \in \Theta'} \|x - y\|^2 + \frac{1}{|\Theta'|} \sum_{x \in \Theta'} \min_{y \in \Theta} \|x - y\|^2 \tag{15}$$

Table 1: Summary of all 3D reconstruction networks discussed in this paper.

Literature	Architecture	Output	Method	Dataset Type	Runtime [s]
Bednarik *et al.* [3] (2018)	VAE with one encoder and three decoders	normal map, depth map, and 3D mesh	based on SegNet [23] with VGG-16 [24] backbone	real, deformable, texture-less surfaces	0.016
Patch-Net [4] (2019)	VAE with one encoder and two decoders	normal and depth maps	converts image to patches, gets 3D shape of patches using [3], and stitches them together	real, deformable, texture-less surfaces	-
Hybrid Deformation Model Network (HDM-Net) [5] (2018)	VAE with one encoder and one decoder	3D point cloud	simple autoencoder with skip connections like ResNet [34], combines 3D regression loss with an isometry prior and a contour loss	synthetic, deformable, well-textured surfaces	0.005
Isometry-Aware Monocular Generative Adversarial Network (IsMo-GAN) [6] (2019)	GAN with two sequential VAEs as generator and a simple CNN as discriminator	3D point cloud	integrates an OD-Net to segment foreground, and trains in an adversarial setting along with 3D loss and isometry prior from [5]	synthetic, deformable, well-textured surfaces and real, deformable, texture-less surfaces	0.004
Pixel2Mesh [7] (2018)	two-lane network with a feature extractor and a graph based mesh predictor (GCN)	3D mesh	feature extractor based on VGG-16 [24], feeds cascaded features to the GCN that uses graph convolutions	synthetic, rigid, well-textured surfaces	0.016
View Attention Guided Network (VANet) [9] (2021)	two-lane feature extractor for both single or multi-view reconstruction, followed by a mesh prediction network	3D mesh	uses channel-wise attention and information from all available views to extract features, which are then sent to a Pixel2Mesh-based [7] mesh vertex predictor	synthetic, rigid, well-textured surfaces	-
Salvi *et al.* [8] (2020)	VAE based on ONets [39] with self-attention [37] in encoder	parametric representation	ResNet-18 [34] encoder with self-attention modules, followed by a decoder and an occupancy function	synthetic, rigid, well-textured surfaces	-
3D-VRVT [41] (2021)	encoder-decoder architecture	voxel grid	encoder based on Vision Transformers [40], followed by a decoder made up of 3D deconvolutions	both synthetic and real, rigid, well-textured surfaces	0.009

where $\|\cdot\|_2$ is the square of Euclidean distance. A smaller CD score indicates a better value.

5. *Chamfer-L1* ($\mathcal{E}_{CD_1}$): The Chamfer distance (CD) has a high computational cost for meshes because of a large number of points, so an approximation called Chamfer-L1 is defined. It uses L1-norm instead of the Euclidean distance [8]. Smaller values are preferred.
6. *Normal Consistency* ($\mathcal{E}_{NC}$): The normal consistency score is defined as the average absolute dot product of normals in one mesh and normals at the corresponding nearest neighbors in the other mesh. It is computed similarly to Chamfer-L1 but the L1-norm is replaced with the dot product of the normal vectors on one mesh with their projection on the other mesh [8]. Normal consistency shows how similar the shapes of two volumes are, and is useful in cases such as where two meshes might overlap significantly, giving a high IoU, but have a different surface shape. Higher normal consistency is preferred.
7. *Earth Mover's Distance* ($\mathcal{E}_{EMD}$):
   The Earth Mover's Distance computes the cost of transforming one one pile of dirt, or one probability distribution, into another. It was introduced in [44] as a metric for image retrieval. In case of 3D reconstruction, it computes the cost of transforming the set of predicted vertices into the groundtruth vertices. The lower the cost, the better the prediction.
8. *F-score* ($\mathcal{E}_F$):
   The F-score evaluates the distance between object surfaces [2,45]. It is defined as the harmonic mean between precision and recall. Precision measures reconstruction accuracy by counting the percentage of predicted points that lie within a certain distance from the ground truth. Recall measures completeness by counting the percentage of points on the ground truth that lie within a certain distance from the prediction. The distance threshold τ can be varied to control the strictness of the F-score. In the results reported in this paper, $\tau = 10^{-4}$.

6.2. Experiments

The RGB-D dataset consisting of normal and depth maps of real data [3] was used by Bednarik *et al.* and Patch-Net in their experiments. This dataset contains five different texture-less surfaces. Experiments were conducted where the network was trained with samples from one surface and then evaluated on samples from another surface. For example, in the experiment "cloth-hoody", the network was trained on the *cloth* object and evaluated on the *hoody* object. The depth and angular errors for these experiments are summarized in Table 3. Patch-Net outperforms [3] in almost all experiments on both the metrics.

Table 3. The texture-less surfaces dataset [3] is used to compare the performance of different depth and normal map reconstruction methods. 128 × 128 size patches were used in the Patch-Net.

Metric	$\mathcal{E}_D$ (mm) ↓		$\mathcal{E}_{MAE}$ (degrees) ↓	
Method	Bednarik *et al.*	Patch-Net [4]	Bednarik *et al.*	Patch-Net [4]
cloth-cloth	17.53 ± 5.50	**12.80 ± 4.45**	17.37 ± 12.51	**14.72 ± 3.39**
tshirt-tshirt	17.18 ± 18.58	**13.70 ± 3.83**	**18.07 ± 12.71**	18.63 ± 4.43
cloth-tshirt	26.26 ± 7.72	**22.74 ± 7.20**	25.74 ± 15.81	**24.29 ± 3.80**
cloth-sweater	38.93 ± 10.36	**30.10 ± 10.00**	31.52 ± 19.07	**27.94 ± 4.79**
cloth-hoody	43.22 ± 24.81	**31.09 ± 8.73**	32.54 ± 21.15	**29.73 ± 2.52**
cloth-paper	24.16 ± 7.15	**14.53 ± 4.48**	35.53 ± 22.16	**24.52 ± 5.96**

The HDM-Net and IsMo-GAN networks were evaluated on the synthetically generated point cloud dataset consisting of a thin, deforming plate with various textures and under different illuminations [5]. The 3D error $\mathcal{E}_{3D}$ and its standard deviation over a set of frames $\mathcal{E}_\sigma$ are reported in Table 4. Results are reported for each of the four textures in the network, under a constant illumination.

Table 4. Comparison of different point cloud reconstruction methods using the synthetic thin plate dataset [5] under one illumination.

Metric	$\mathcal{E}_{3D}$ (mm) ↓		$\mathcal{E}_\sigma$ (mm) ↓	
Method	HDM-Net [5]	IsMo-GAN [6]	HDM-Net [5]	IsMo-GAN [6]
endoscopy	48.50	**33.60**	**13.50**	14.80
graffiti	49.90	**33.30**	22.00	**20.80**
clothes	48.90	**35.30**	26.40	**24.20**
carpet	144.20	**110.50**	26.90	**26.80**
mean	72.88	**53.18**	22.20	**21.65**

For a small subset of the *cloth* object in the texture-less dataset, groundtruth 3D meshes are also provided. For this cloth data, 3D error is reported for [3], HDM-Net and IsMo-GAN. In addition, [6] removed the textures from surfaces in the synthetic thin plate data to create another texture-less data. They report the 3D error on this data for HDM-Net and IsMo-GAN. We summarize these results for texture-less mesh and point cloud reconstruction in Table 5.

Table 5. Comparison of mesh reconstruction from texture-less data. As can be seen, IsMo-GAN outperforms [3] and HDM-Net on the real texture-less cloth data by 26.5% and 10.5% respectively, and it outperforms HDM-Net on the synthetic texture-less thin plate data by 31.9%

Metric	$\mathcal{E}_{3D}$ (mm) ↓		
Method	**Bednarik *et al.***	**HDM-Net** [5]	**IsMo-GAN** [6]
cloth [3]	21.48	17.65	**15.79**
plate [5,6]	-	99.40	**67.70**

For most other experiments, the Choy *et al.* subset of the ShapeNet dataset [10], described in Section 4.4, is used. For each of the 13 objects in this dataset, various error metrics are reported for the task of mesh reconstruction. Table 6 summarizes these results, reporting the F-score, Earth Mover's Distance, and the Chamfer Distance metrics. Salvi *et al.* report the IoU, Chamfer-L1 and Normal Consistency metrics on the same dataset. These results are summarized in Table 7.

7. Discussion

There are many different methods for 3D reconstruction from 2D images. In this paper, we discussed several of them - including those that use depth maps, normal maps, point clouds, surface meshes or volumetric data. Each of these approaches has its own advantages and disadvantages, but no one method is perfect. This makes the task of 3D reconstruction an ill-posed problem, which requires careful consideration when choosing a method to use. In single-view 3D reconstruction, this becomes even more challenging because the network has to reconstruct the shape of surfaces that may not even be visible in the image at all.

We discussed two methods that reconstruct 3D shape of texture-less surfaces. In general, the more distinctive textures an image contains, the easier it is to reconstruct in 3D. With only a single RGB image - and that containing surfaces with no distinctive textures - dense reconstruction in 3D can be very difficult. That is why Bednarik *et al.* and Patch-Net only reconstruct the so-called 2.5D shape in form of normal and depth maps. They both use a very similar network architecture, but [4] uses a patch-based strategy instead of reconstructing the whole image together. By doing so, they are able to reduce the depth error by 25.3% and the mean angular error of the surface normals by 13.0%. Later,

Table 6. Comparison of (**A**) Pixel2Mesh [7] and (**B**) VANet [9] on the Choy *et al.* subset of the ShapeNet dataset [10]. VANet outperforms Pixel2Mesh in all metrics, with 10-40% improvement.

Metric	$\mathcal{E}_F$ (%) ↑		$\mathcal{E}_{EMD}$ ↓		$\mathcal{E}_{CD}$ ↓	
Method	**A**	**B**	**A**	**B**	**A**	**B**
plane	71.12	**77.01**	0.579	**0.486**	0.477	**0.304**
bench	57.57	**67.69**	0.965	**0.770**	0.624	**0.362**
cabinet	60.39	**63.30**	2.563	**1.575**	0.381	**0.327**
car	67.86	**69.53**	1.297	**1.185**	0.268	**0.235**
chair	54.38	**60.74**	1.399	**0.957**	0.610	**0.443**
monitor	51.39	**60.35**	1.536	**1.269**	0.755	**0.459**
lamp	48.15	**56.26**	1.314	**1.086**	1.295	**0.879**
speaker	48.84	**53.49**	2.951	**2.283**	0.739	**0.562**
firearm	73.20	**77.24**	0.667	**0.473**	0.453	**0.333**
sofa	51.90	**56.83**	1.642	**1.376**	0.490	**0.400**
table	66.30	**70.78**	1.480	**1.173**	0.498	**0.334**
phone	70.24	**72.27**	0.724	**0.573**	0.421	**0.298**
watercraft	55.12	**62.12**	0.814	**0.718**	0.670	**0.450**
mean	59.73	**65.20**	1.380	**1.071**	0.591	**0.414**

Table 7. Comparison of (**A**) Pixel2Mesh [7], (**C**) Salvi *et al.* and 3D-VRVT [41] on the Choy *et al.* subset of the ShapeNet dataset [10]. With self-attention after ResNet layers, [8] improves IoU by 25%, Normal Consistency by 8.9%, and reduces the average Chamfer-L1 distance approximately 11 times. [41] further improves the IoU by 9% with a Vision Transformer architecture containing multi-headed self-attention.

Metric	$\mathcal{E}_{IoU}$ ↑			$\mathcal{E}_{CD_1}$ ↓		$\mathcal{E}_{NC}$ ↑	
Method	**A**	**C**	**D**	**A**	**C**	**A**	**C**
plane	0.420	**0.645**	0.608	0.187	**0.011**	0.759	**0.868**
bench	0.323	0.493	**0.563**	0.201	**0.016**	0.732	**0.813**
cabinet	0.664	0.737	**0.794**	0.196	**0.016**	0.834	**0.876**
car	0.552	0.761	**0.855**	0.180	**0.014**	0.756	**0.855**
chair	0.396	0.534	**0.553**	0.265	**0.021**	0.746	**0.829**
monitor	0.490	0.520	**0.555**	0.239	**0.026**	0.830	**0.863**
lamp	0.323	0.379	**0.436**	0.308	**0.045**	0.666	**0.722**
speaker	0.599	0.660	**0.725**	0.285	**0.028**	0.782	**0.839**
firearm	0.402	0.527	**0.597**	0.164	**0.012**	0.718	**0.804**
sofa	0.613	0.689	**0.716**	0.212	**0.019**	0.820	**0.866**
table	0.395	0.535	**0.617**	0.218	**0.019**	0.784	**0.861**
phone	0.661	0.754	**0.805**	0.149	**0.012**	0.907	**0.937**
watercraft	0.397	0.568	**0.604**	0.212	**0.018**	0.699	**0.801**
mean	0.480	0.600	**0.654**	0.216	**0.019**	0.772	**0.841**

Shimada *et al.* used a subset of the same texture-less surfaces data to directly reconstruct 3D meshes. They report results for HDM-Net and their own IsMo-GAN network, showing 17.8% and 26.5% improvement respectively over [3]'s original texture-less surfaces method for meshes.

HDM-Net, like [3], is an encoder-decoder network and has a similar VGG-16 backbone but fewer convolutional layers. The major difference in the encoder is the ResNet-like skip connections. In [3], the decoder consisted of only one convolutional layer, followed by pooling and a fully connected layer. HDM-Net uses a much larger decoder with transposed convolutions for upsampling. Finally, while [3] for its mesh decoder used a simple mean squared error loss function, HDM-Net combines that with two more loss functions that enforce isometry and contour constraints. The skip connections, a larger decoder, and domain-specific loss functions help HDM-Net get a significantly improved reconstruction compared to [3]. IsMo-GAN takes the isometry constraints from HDM-Net and uses an adversarial training setting to improve the mesh reconstruction accuracy for texture-less surfaces even further.

HDM-Net and IsMo-GAN also evaluate their methods on a synthetic dataset of a deforming plate with a variety of textures. In addition to the adversarial training, IsMo-GAN's integrated Object Detection Network and the choice of LeakyReLU activation instead of ReLU help it outperform HDM-Net when reconstructing a 3D point clouds by 10-30% in various experiments.

Next, we talked about two methods that reconstruct 3D meshes using Choy *et al.*'s rendered images of the ShapeNet dataset; Pixel2Mesh [7] and VANet. Both use a dual-lane architecture with separate roles for each lane. Pixel2Mesh has one lane for feature extraction, which feeds cascaded input to the other lane that performs the reconstruction. On the other hand, VANet's two lanes are both for feature extraction, and have shared weights. VANet can be used for both single and multi-view reconstruction, and the first lane extracts features from the main view while the second from all auxiliary views. The feature vectors extracted from both lanes are concatenated and sent to another module which has an architecture similar to Pixel2Mesh's mesh reconstruction lane. This uses graph based convolutions to generate a mesh from extracted features. By primarily improving their feature extractor, which uses channel-wise attention, VANet was then able to use the same reconstruction network as Pixel2Mesh to get up to 40% reduction of the average Chamfer distance error (see Table 6).

Finally, we discussed Salvi *et al.*'s attentioned Occupancy Network for reconstructing the 3D shape as a continuous function. The motivation for this type of output is to have a standard method for efficiently representing a 3D shape. In this network, an encoder with ResNet-18 backbone is augmented with several self-attention blocks to improve comprehension of global dependencies in the input images. This is then passed to a decoder network and an occupancy function to get the final output, which can then be used to obtain a mesh at any resolution. The mesh output is compared to Pixel2Mesh using several metrics, and Salvi *et al.* get 11 times smaller average Chamfer-L1 distance than Pixel2Mesh. No metrics directly comparing this method with VANet are reported, but as this also uses the same ShapeNet dataset, and the reported Chamfer-L1 distane is a variant of the standard Chamfer distance reported by VANet, it can be assumed that self-attention in VANet's encoder would improve reconstruction accuracy of that network, as well. However, as [8] showed in some of their experiments, the benefit from self-attention modules was highest when used at the early layers of the encoder as opposed to the later layers.

8. Conclusions

This paper reviewed several methods for monocular 3D reconstruction and concluded that it remains an ill-posed problem. This is primarily because of a lack of a standardized, one-size-fits-all method of representing 3D shapes, as well as non-availability of standard datasets that cover the whole range of objects to be reconstructed. For example, while there are a lot of large-scale synthetic datasets, real-world 3D datasets are rare and smaller on average. This is because of the inherent difficulty of capturing real 3D data. When it comes to texture-less surfaces, there are even fewer datasets available, and those that are available only contain RGB-D data. Standard datasets for deformable texture-less surfaces with groundtruth point clouds or meshes provided are not available. This is why most texture-less reconstruction methods only reconstruct depth maps or surface normals.

As discussed before, most networks in 3D reconstruction are made up of encoder-decoder architectures, which in turn are based on semantic segmentation networks like SegNet and UNet. Various enhancements in these networks have shown promise, including the use of domain specific losses that enforce various shape constraints on the surfaces.

An example of this is the isometry prior used by some networks. Other networks have shown that using ResNet-like skip connections improves the representation power of the network, giving better results. Attention-based methods have also shown good results, with self-attention modules when compared with ResNet's skip connections, improving the performance of previous methods manifolds. However, self-attention is an expensive process which cannot be practically used everywhere in the encoder-decoder networks. Instead, their use is limited to the earlier layers of the encoder. Like self-attention, the use of max pooling indices from the encoder layers in the decoder layers is also an expensive process as it requires additional memory at each forward pass. Some networks have successfully used interpolation and transposed convolutions for upsampling, eliminating the need of pooling indices from the encoder. Using an adversarial training in a GAN-based network also showed significantly better results. The GAN-based network also performed significantly faster than all other methods reviewed in this paper, generating point clouds at 250 Hz frequency, thus making it suitable for runtime 3D reconstruction. Tranformers are also gaining traction for vision tasks, with one network using Vision Trnasformers for single-view 3D reconstruction. This network shows promising results at reconstructing a low-resoultion voxel grid, and improves the IoU on the ShapeNet dataset by 9% compared to other state-of-the-art methods.

In the future, 3D reconstruction research can benefit from more datasets, especially for texture-less surfaces and from real-world objects. For depth maps and surface normals, simple RGB-D cameras like Microsoft Kinect may be used, but it is generally very challenging to collect real-world datasets of the same size as can be generated synthetically. Collecting groundtruth point clouds or 3D meshes from real-world objects poses an even bigger challenge because of more specialized equipment required. Synthetic datasets focusing on texture-less surfaces in particular, and including both deforming and rigid surfaces, can be generated using software like Blender. In addition to dataset creation, methods to deal with the bottlenecks created by self-attention modules and pooling indices also need to be studied. The full potential of Transformers in context of single-view 3D reconstruction is also still unknown. Finally, the use of adversarial training and generator-discriminator networks for 3D reconstruction needs to be further explored.

Author Contributions: writing—original draft preparation, M.S.U.K., M.Z.A.; writing—review and editing, M.S.U.K., M.Z.A., M.L.; supervision and project administration, A.P., D.S. All authors have read and agreed to the submitted version of the manuscript.

Funding: The work leading to this publication has been partially funded by the European project INFINITY under Grant Agreement ID 883293.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

2D	Two-Dimensional
3D	Three-Dimensional
CAD	Computer-Aided Design
CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
NLP	Natural Language Processing
RGB	Red-Green-Blue
RGB-D	Red-Green-Blue-Depth
RNN	Recurrent Neural Network
VAE	Variational Autoencoder
VOC	Visual Object Classes

References

1. Bautista, M.A.; Talbott, W.; Zhai, S.; Srivastava, N.; Susskind, J.M. On the generalization of learning-based 3d reconstruction. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 2180–2189.
2. Tatarchenko, M.; Richter, S.R.; Ranftl, R.; Li, Z.; Koltun, V.; Brox, T. What do single-view 3d reconstruction networks learn? Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 3405–3414.
3. Bednarik, J.; Fua, P.; Salzmann, M. Learning to reconstruct texture-less deformable surfaces from a single view. 2018 International Conference on 3D Vision (3DV). IEEE, 2018, pp. 606–615.
4. Tsoli, A.; Argyros, A.A. Patch-Based Reconstruction of a Textureless Deformable 3D Surface from a Single RGB Image. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, 2019.
5. Golyanik, V.; Shimada, S.; Varanasi, K.; Stricker, D. HDM-Net: Monocular Non-Rigid 3D Reconstruction with Learned Deformation Model. *CoRR* **2018**, *abs/1803.10193*, [1803.10193].
6. Shimada, S.; Golyanik, V.; Theobalt, C.; Stricker, D. IsMo-GAN: Adversarial Learning for Monocular Non-Rigid 3D Reconstruction. *CoRR* **2019**, *abs/1904.12144*, [1904.12144].
7. Wang, N.; Zhang, Y.; Li, Z.; Fu, Y.; Liu, W.; Jiang, Y.G. Pixel2mesh: Generating 3d mesh models from single rgb images. Proceedings of the European conference on computer vision (ECCV), 2018, pp. 52–67.
8. Salvi, A.; Gavenski, N.; Pooch, E.; Tasoniero, F.; Barros, R. Attention-based 3D Object Reconstruction from a Single Image. 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020, pp. 1–8.
9. Yuan, Y.; Tang, J.; Zou, Z. Vanet: a View Attention Guided Network for 3d Reconstruction from Single and Multi-View Images. 2021 IEEE International Conference on Multimedia and Expo (ICME), 2021, pp. 1–6. doi:10.1109/ICME51207.2021.9428171.
10. Chang, A.X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; Xiao, J.; Yi, L.; Yu, F. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.
11. Zollhöfer, M.; Garrido, J.T.P.; Pérez, D.B.T.B.P.; Stamminger, M.; Theobalt, M.N.C. State of the Art on Monocular 3D Face Reconstruction, Tracking, and Applications. *STAR* **2018**, *37*.
12. Yuniarti, A.; Suciati, N. A review of deep learning techniques for 3D reconstruction of 2D images. 2019 12th International Conference on Information & Communication Technology and System (ICTS). IEEE, 2019, pp. 327–331.
13. Han, X.F.; Laga, H.; Bennamoun, M. Image-based 3D object reconstruction: State-of-the-art and trends in the deep learning era. *IEEE transactions on pattern analysis and machine intelligence* **2019**, *43*, 1578–1604.
14. Laga, H. A survey on deep learning architectures for image-based depth reconstruction. *arXiv preprint arXiv:1906.06113* **2019**.
15. Liu, C.; Kong, D.; Wang, S.; Wang, Z.; Li, J.; Yin, B. Deep3D reconstruction: methods, data, and challenges. *Frontiers of Information Technology & Electronic Engineering* **2021**, *22*, 652–672.
16. Maxim, B.; Nedevschi, S. A survey on the current state of the art on deep learning 3D reconstruction. 2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP). IEEE, 2021, pp. 283–290.
17. Fu, K.; Peng, J.; He, Q.; Zhang, H. Single image 3D object reconstruction based on deep learning: A review. *Multimedia Tools and Applications* **2021**, *80*, 463–498.
18. Choy, C.B.; Xu, D.; Gwak, J.; Chen, K.; Savarese, S. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. Proceedings of the European Conference on Computer Vision (ECCV), 2016.
19. Community, B.O. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.
20. Miller, G.A. WordNet: A Lexical Database for English. *Communications of the ACM* **1995**, *38*, 39–41.
21. Griffiths, D.; Boehm, J. A review on deep learning techniques for 3D sensed data classification. *Remote Sensing* **2019**, *11*, 1499.
22. Team, S.R. About ShapeNet.
23. Badrinarayanan, V.; Handa, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling. *CoRR* **2015**, *abs/1505.07293*, [1505.07293].
24. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* **2014**.

25. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. International Conference on Learning Representations (ICLR), 2015.
26. Chollet, F.; others. Keras, 2015.
27. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; Zheng, X. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.
28. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR* **2015**, *abs/1505.04597*, [1505.04597].
29. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Wallach, H.; Larochelle, H.; Beygelzimer, A.; dAlché-Buc, F.; Fox, E.; Garnett, R., Eds.; Curran Associates, Inc., 2019; pp. 8024–8035.
30. Otsu, N. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics* **1979**, *9*, 62–66.
31. Suzuki, S.; others. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing* **1985**, *30*, 32–46.
32. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems* **2014**, *27*.
33. Bronstein, M.M.; Bruna, J.; LeCun, Y.; Szlam, A.; Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* **2017**, *34*, 18–42.
34. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *CoRR* **2015**, *abs/1512.03385*, [1512.03385].
35. Fan, H.; Su, H.; Guibas, L.J. A point set generation network for 3d object reconstruction from a single image. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 605–613.
36. Oh Song, H.; Xiang, Y.; Jegelka, S.; Savarese, S. Deep metric learning via lifted structured feature embedding. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 4004–4012.
37. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Advances in neural information processing systems* **2017**, *30*.
38. Zhang, H.; Goodfellow, I.; Metaxas, D.; Odena, A. Self-attention generative adversarial networks. International conference on machine learning. PMLR, 2019, pp. 7354–7363.
39. Mescheder, L.; Oechsle, M.; Niemeyer, M.; Nowozin, S.; Geiger, A. Occupancy networks: Learning 3d reconstruction in function space. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4460–4470.
40. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; others. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* **2020**.
41. Li, X.; Kuang, P. 3D-VRVT: 3D Voxel Reconstruction from A Single Image with Vision Transformer. 2021 International Conference on Culture-oriented Science & Technology (ICCST). IEEE, 2021, pp. 343–348.
42. Eigen, D.; Puhrsch, C.; Fergus, R. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems* **2014**, *27*.
43. Stegmann, M.B.; Gomez, D.D. A brief introduction to statistical shape analysis. *Informatics and mathematical modelling, Technical University of Denmark, DTU* **2002**, *15*.
44. Rubner, Y.; Tomasi, C.; Guibas, L.J. The earth mover's distance as a metric for image retrieval. *International journal of computer vision* **2000**, *40*, 99–121.
45. Knapitsch, A.; Park, J.; Zhou, Q.Y.; Koltun, V. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)* **2017**, *36*, 1–13.