# Deep Marching Cubes: Learning Explicit Surface Representations

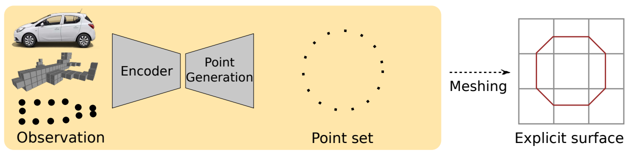
Yiyi Liao[1,2] Simon Donné[1,3] Andreas Geiger[1,4]
[1]Autonomous Vision Group, MPI for Intelligent Systems Tübingen
[2]Institute of Cyber-Systems and Control, Zhejiang University
[3]imec - IPI - Ghent University [4]CVG Group, ETH Zürich
{yiyi.liao,simon.donne,andreas.geiger}@tue.mpg.de

## Abstract

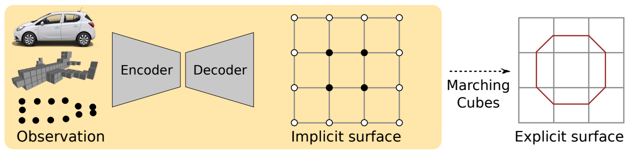
*Existing learning based solutions to 3D surface prediction cannot be trained end-to-end as they operate on intermediate representations (e.g., TSDF) from which 3D surface meshes must be extracted in a post-processing step (e.g., via the marching cubes algorithm). In this paper, we investigate the problem of end-to-end 3D surface prediction. We first demonstrate that the marching cubes algorithm is not differentiable and propose an alternative differentiable formulation which we insert as a final layer into a 3D convolutional neural network. We further propose a set of loss functions which allow for training our model with sparse point supervision. Our experiments demonstrate that the model allows for predicting sub-voxel accurate 3D shapes of arbitrary topology. Additionally, it learns to complete shapes and to separate an object's inside from its outside even in the presence of sparse and incomplete ground truth. We investigate the benefits of our approach on the task of inferring shapes from 3D point clouds. Our model is flexible and can be combined with a variety of shape encoder and shape inference techniques.*

## 1. Introduction

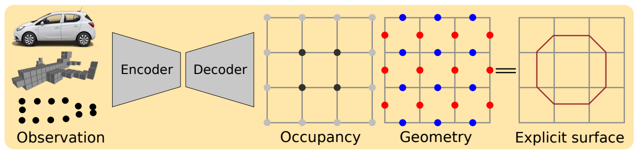
3D reconstruction is a core problem in computer vision, yet despite its long history many problems remain unsolved. Ambiguities or noise in the input require the integration of strong geometric priors about our 3D world. Towards this goal, many existing approaches formulate 3D reconstruction as inference in a Markov random field [2, 21, 41, 46] or as a variational problem [17, 47]. Unfortunately, the expressiveness of such prior models is limited to simple local smoothness assumptions [2, 17, 21, 47] or very specialized shape models [1, 15, 16, 42]. Neither can such simple priors resolve strong ambiguities, nor are they able to reason about missing or occluded parts of the scene. Hence, existing 3D reconstruction systems work either in narrow domains where specialized shape knowledge is available, or require well captured and highly-textured environments.



(a) **Sparse Point Prediction** (e.g., [12])



(b) **Implicit Surface Prediction** (e.g., [35, 45])



(c) **Explicit Surface Prediction** (ours)

Figure 1: **Illustration** comparing point prediction (a), implicit surface prediction (b) and explicit surface prediction (c). The encoder is shared across all approaches and depends on the input (we use point clouds in this paper). The decoder is specific to the output representation. All trainable components are highlighted in yellow. Note that only (c) can be trained end-to-end for the surface prediction task.

However, the recent success of deep learning [19, 20, 38] and the availability of large 3D datasets [5, 6, 9, 26, 37] nourishes hope for models that are able to learn powerful 3D shape representations from data, allowing reconstruction even in the presence of missing, noisy and incomplete observations. And indeed, recent advances in this area [7, 12, 18, 24, 34, 36, 39, 40] suggest that this goal can ultimately be achieved.

Existing 3D representation learning approaches can be classified into two categories: voxel based methods and point based methods, see Fig. 1 for an illustration. Voxel based methods [34,35,45] use a grid of voxels as output representation and predict either voxel occupancy [34, 45] or a truncated signed distance field (TSDF) which implicitly determines the surface [35]. Point based methods [12] directly regress a fixed number of points as output. While voxel and point based representations are easy to implement, both require a post processing step to retrieve the actual 3D surface mesh which is the quantity of interest in 3D reconstruction. For point based methods, meshing techniques such as Poisson surface reconstruction [25] or SSD [4] can be employed. In contrast, implicit voxel based methods typically use marching cubes [29] to extract the zero level set.

As both techniques cannot be trained end-to-end for the 3D surface prediction task, an auxiliary loss (e.g., Chamfer distance on point sets, $\ell_1$ loss on signed distance field) must be used during training. However, there are two major limitations in this setup: firstly, while implicit methods require 3D supervision on the implicit model, the ground truth of the implicit representation is often hard to obtain, e.g., in the presence of a noisy and incomplete point cloud or when the inside and outside of the object is unknown. Secondly, these methods only optimize an auxiliary loss defined on an intermediate representation and require an additional post-processing step for surface extraction. Thus they are unable to directly constrain the properties of the predicted surface.

In this work, we propose *Deep Marching Cubes (DMC)*, a model which predicts explicit surface representations of arbitrary topology. Inspired by the seminal work on Marching Cubes [29], we seek for an end-to-end trainable model that directly produces an explicit surface representation and optimizes a geometric loss function. This avoids the need for defining auxiliary losses or converting target meshes to implicit distance fields. Instead, we directly train our model to predict surfaces that agree with the 3D observations. We demonstrate that direct surface prediction can lead to more accurate reconstructions while also handling noise and missing observations. Besides, this allows for separating inside from outside even if the ground truth is sparse or not watertight, as well as easily integrating additional priors about the surface (e.g., smoothness). We summarize our contributions as follows:

- We demonstrate that Marching Cubes is not differentiable with respect to topological changes and propose a modified representation which is differentiable.
- We present a model for end-to-end surface prediction and derive appropriate geometric loss functions. Our model can be trained from unstructured point clouds and does not require explicit surface ground truth.
- We propose a novel loss function which allows for separating an object's inside from its outside even when learning with sparse unstructured 3D data.
- We apply our model to several surface prediction tasks and demonstrate its ability to recover surfaces even in the presence of incomplete or noisy ground truth.

Our code and data is available on the project website[1].

## 2. Related Work

**Point Based Representations:** Point based representations have a long history in robotics and computer graphics. However, the irregular structure complicates the usage of point clouds in deep learning. Qi et al. [31] proposed PointNet for point cloud classification and segmentation. Invariance wrt. the order of the points is achieved by means of a global pooling operation over all points. As global pooling does not preserve local information, a hierarchical neural network that applies PointNet recursively on a nested partitioning of the input point set has been proposed in follow-up work [33]. Fan et al. [12] proposed a model for sparse 3D reconstruction, predicting a point set from a single image. While point sets require less parameters to store compared to dense volumetric grids, the maximal number of points which can be predicted is limited to a few thousand due to the simple fully connected decoder. In contrast to the method proposed in this paper, an additional post-processing step [4, 25] is required to "lift" the 3D point cloud to a dense surface mesh.

**Implicit Surface Representations:** Implicit surface representations are amongst the most widely adopted representations in 3D deep learning as they can be processed by means of standard 3D CNNs. By far the most popular representation are binary occupancy grids which have been applied to a series of discriminative tasks such as 3D object classification [30, 32], 3D object detection [38] and 3D reconstruction [7, 13, 34, 44, 45]. Its drawback, however, is obvious: the accuracy of the predicted reconstruction is limited to the size of a voxel. While most existing approaches are limited to a resolution of $32^3$ voxels, methods that exploit adaptive space partitioning techniques [18, 39] scale up to $256^3$ or $512^3$ voxel resolution. Yet, without sub voxel estimation, the resulting reconstructions exhibit voxel-based discretization artefacts. Sub voxel precision can be achieved by exploiting the truncated signed distance function (TSDF) [8] as representation where each voxel stores the truncated signed distance to the closest 3D surface point [10, 28, 35].

While the aforementioned works require post-processing for isosurface extraction, e.g., using Marching Cubes [29], here we propose an end-to-end trainable solution which integrates this step into one coherent model. This allows for training the model directly using point based supervision and geometric loss functions. Thus, our model avoids the need for converting the ground truth point cloud or mesh

[1]https://avg.is.tue.mpg.de/research_projects/deep-marching-cubes

into an intermediate representation (e.g., TSDF) and defining auxiliary loss functions. It is worth noting that this conversion is not only undesirable but often also very difficult, i.e., when learning from unordered point sets or non-watertight meshes for which inside/outside distinction is difficult. Our approach avoids the conversion step and instead infers such relationships using weak prior knowledge.

**Explicit Surface Representations:** Compared to implicit surface representations, explicit surface representations are less structured as they are typically organized as a set of vertices and faces, complicating their deployment in deep learning. Several works consider the problem of shape classification and segmentation by defining neural networks which operate on the graph spanned by the edges and vertices of a 3D mesh [3, 14, 43]. However, these methods assume a fixed input graph while in 3D reconstruction the graph (i.e., mesh) itself needs to be inferred. Very limited results have been presented for mesh based inference, and existing works are restricted by a fixed 3D topology or mild deviations from a 3D template. Rezende et al. [34] predict a small number of vertices using a fully connected network. Each vertex is constrained to move along a pre-defined line. Thus, their method is limited to very simple convex shapes (they consider spheres, cuboids and cylinders) with a small number of vertices. Kong et al. [27] predict a mesh by deforming the vertices of a nearest neighbor CAD model, resulting in predictions close to the original shape templates. Kanazawa et al. [23] also predict meshes, however their method is specialized to human body shapes.

Our goal is to overcome these difficulties by combining voxel and mesh based representations. Our decoder operates in a volumetric space, but predicts the local face parameters of the surface mesh. Compared to the aforementioned methods, our representation is scalable regarding the number of vertex points while allowing for arbitrary topologies and the prediction of non-convex shapes. No shape templates are required at test time and the model generalizes well to unseen shape categories.

## 3. Deep Marching Cubes

We tackle the problem of predicting an explicit surface representation (i.e., a mesh) directly from raw observations (e.g., a mesh, point cloud, volumetric data or an image). Existing works formulate this problem as the prediction of an intermediate signed distance representation using an auxiliary (typically $\ell_1$) loss [10, 35], followed by applying the Marching Cubes (MC) algorithm [29]. In this work we aim at making this last step differentiable, hence allowing for end-to-end training using surface based geometric loss functions.

We first provide a formal introduction to the Marching Cubes algorithm [29]. We then demonstrate that backpropagation through this algorithm is intractable and propose a modified differentiable representation which avoids these problems. We exploit this representation as a *Differentiable Marching Cubes Layer (DMCL)* in a neural network for end-to-end surface prediction of arbitrary topology.

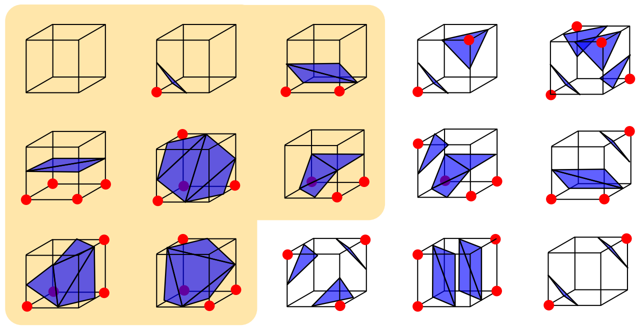
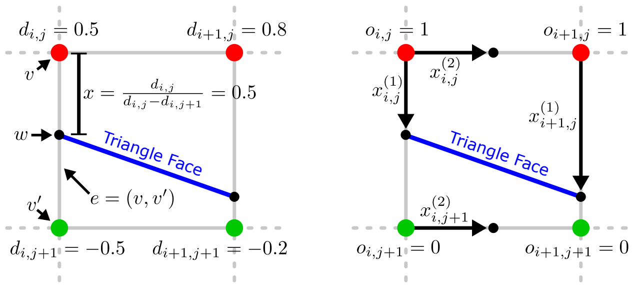


Figure 2: **Mesh Topology.** The $2^8 = 256$ topologies can be grouped into 15 equivalence classes due to rotational symmetry. In this paper, we consider only the singly connected topologies (highlighted in yellow).



(a) Marching Cubes (b) Differentiable MC

Figure 3: **Representation** used by Marching Cubes (a) and the proposed Differentiable Marching Cubes (b). The former uses an implicit surface representation based on signed distances $\mathbf{D}$ while the latter exploits an explicit surface representation which is parameterized in terms of occupancy probabilites $\mathbf{O}$ and vertex displacements $\mathbf{X}$.

### 3.1. Marching Cubes

The Marching Cubes (MC) algorithm extracts the zero level set of a signed distance field and represents it as a set of triangles. It comprises two steps: estimation of the topology (i.e., the number and connectivity of triangles in each cell of the volumetric grid) and the prediction of the vertex locations of the triangles, determining the geometry.

More formally, let $\mathbf{D} \in \mathbb{R}^{N \times N \times N}$ denote a (discretized) signed distance field obtained using volumetric fusion [8] or predicted by a neural network [10, 35] where $N$ denotes the number of voxels along each dimension. Let further $d_{\mathbf{n}} \in \mathbb{R}$ denote the $\mathbf{n}$'th element of $\mathbf{D}$ where $\mathbf{n} = (i, j, k) \in \mathbb{N}^3$ is a multi-index ($i, j, k$ correspond to the 3 dimensions of $\mathbf{D}$). As $\mathbf{D}$ is a signed distance field, $|d_{\mathbf{n}}|$ is the distance between voxel $\mathbf{n}$ and its closest surface point. Without loss of generality, let us assume that $d_\mathbf{n} > 0$ if voxel $\mathbf{n}$ is located inside an object and $d_\mathbf{n} < 0$ otherwise. The zero level set of the signed distance field $\mathbf{D}$ defines the surface which can be represented by means of a triangular mesh $\mathbf{M}$. This mesh $\mathbf{M}$ can be extracted from $\mathbf{D}$ using the Marching Cubes (MC) algorithm [29] which iterates ("marching") through all *cells* of the grid connecting the *voxel* centers and inserts triangular faces whenever a sign change is detected[2]. More specifically, MC performs the following two steps:

First, the cell's surface topology $\mathbf{T}$ is determined based on the sign of $d_\mathbf{n}$ at its 8 corners. $\mathbf{T}$ can be represented as a binary tensor $\mathbf{T} \in \{0,1\}^{2\times 2\times 2}$ where each element represents a corner. The total number of configurations equals $2^8 = 256$, see Fig. 2 for an illustration. A vertex is created in case of a sign change of the distance values of two adjacent corners of the cell (i.e., corners connected by an edge). The vertex is placed at the edge connecting both corners.

In a second step, the vertex location of each triangular face along the edge is determined using linear interpolation. More formally, let $x \in [0,1]$ denote the relative location of a triangle vertex $w$ along edge $e = (v, v')$ where $v$ and $v'$ are the corresponding edge vertices as illustrated in Fig. 3a. In particular, let's assume $x = 0$ if $w = v$ and $x = 1$ if $w = v'$. Let further $d \in \mathbb{R}$ and $d' \in \mathbb{R}$ denote the signed distance values at $v$ and $v'$, respectively. In the Marching Cubes algorithm, $x$ is determined from $d$ and $d'$ as the zero crossing of the linear interpolant of $d$ and $d'$. This interpolant is given as $f(x) = d + x(d' - d)$. Setting $f(x) = 0$ yields $x = d/(d - d')$, see also Fig. 3a.

**Discussion:** Given the MC algorithm, can we construct a deep neural network for end-to-end surface prediction? Indeed, we could try to construct a deep neural network which predicts a signed distance field that is converted into a triangular mesh using MC. We could then compare this surface to a ground truth surface or point cloud and backpropagate errors through the MC layer and the neural network. Unfortunately, this approach is intractable for two reasons:

- First, $x = d/(d - d')$ is singular at $d = d'$, thus preventing topological changes during training. However, the topology is unknown at training time if a point cloud or a partial mesh is used as input. Instead, the network needs to learn the topology *during* training.

- Second, observations affect only grid cells in their immediate vicinity, i.e., they act solely on cells where the surface passes through. Thus gradients are not propagated to cells further away from the predicted surface.

[2]We distinguish **voxels** and **cells** in this paper: voxels are the regular representation used by occupancy maps, while cells are displaced by a distance of 0.5 voxels and connect the voxel centers. Marching cubes as well as our algorithm operates on the edges and vertices of these cells.

To circumvent these problems we propose a modified *differentiable* representation which separates the mesh topology from the geometry. In contrast to predicting signed distance values, we predict 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. In addition, we predict a vertex location for every edge of each cell. The combination of both implicitly defined topology and vertex location defines a distribution over meshes which is differentiable and can be used for backpropagation. The second problem can be tackled by introducing appropriate loss functions on the occupancy and the vertex location variables.

Note that predicting occupancies instead of distance values is not a limitation as the surface computed via MC does not depend on cells further away. Similar to MC, our representation is flexible in terms of the output topology.

## 3.2. Differentiable Marching Cubes

We now formalize our *Differentiable Marching Cubes Layer (DMCL)*. Let again $\mathbf{n} = (i, j, k) \in \mathbb{N}^3$ denote a multi-index into a 3D tensor and let $\mathbf{1} = (1, 1, 1)$ index the first element of the tensor. Let $\mathbf{O} \in [0,1]^{N\times N\times N}$ denote the occupancy field and let $\mathbf{X} \in [0,1]^{N\times N\times N\times 3}$ denote the vertex displacement field predicted by a neural network (see Section 3.3 for details on the network architecture). Let $o_\mathbf{n} \in [0,1]$ denote the $\mathbf{n}$'th element of $\mathbf{O}$, representing the occupancy probability of that voxel with $o = 1$ if the voxel is occupied. Similarly, let $\mathbf{x_n} \in [0,1]^3$ denote the $\mathbf{n}$'th element of $\mathbf{X}$, representing the displacements of the triangle vertices along the edges associated with $\mathbf{x_n}$. Note that $\mathbf{x_n}$ is a 3-dimensional vector as we need to specify one vertex displacement for each dimension of the 3D space (see Fig. 3b). Let $w$ denote a vertex of the output mesh located on edge $e = (v, v')$. As before, we have $x = 0$ if $w = v$ and $x = 1$ if $w = v'$. In other words, $w$ is displaced linearly between $v$ and $v'$ based on $x$.

The topology can be implicitly defined via the occupancy variables. We consider the predictions of the neural network $o_\mathbf{n} \in [0,1]$ as parameters of a Bernoulli distribution

$$p_\mathbf{n}(t) = (o_\mathbf{n})^t (1 - o_\mathbf{n})^{1-t} \tag{1}$$

where $t \in \{0,1\}$ is a random variable and $p_\mathbf{n}(t)$ is the probability of voxel $\mathbf{n}$ being occupied ($t = 1$) or unoccupied ($t = 0$). Let now $\{o_\mathbf{n}, \dots, o_{\mathbf{n}+\mathbf{1}}\}$ denote the $2^3 = 8$ occupancy variables corresponding to the 8 corners of the $\mathbf{n}$'th grid cell. Let further $\mathbf{T} \in \{0,1\}^{2\times 2\times 2}$ denote a binary random tensor representing the topology. The probability for topology $\mathbf{T}$ at grid cell $\mathbf{n}$ is the product of the 8 occupancy probabilities at its corners

$$p_\mathbf{n}(\mathbf{T}) = \prod_{\mathbf{m}\in\{0,1\}^3} (o_{\mathbf{n}+\mathbf{m}})^{t_\mathbf{m}} (1 - o_{\mathbf{n}+\mathbf{m}})^{1-t_\mathbf{m}} \tag{2}$$

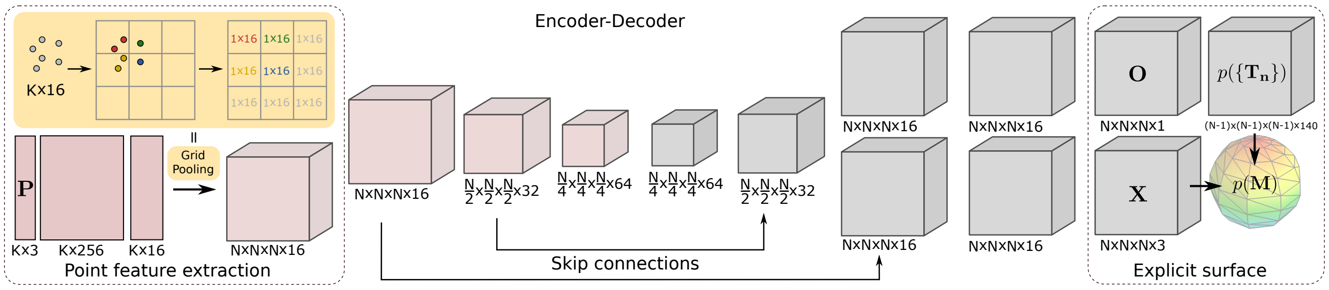


Figure 4: **Network Architecture.** The input point cloud $\mathbf{P}$ is converted into a volumetric representation using grid pooling. The grid pooling operation (highlighted in yellow) takes as input a set of $K$ points with their $D = 16$ dimensional feature maps and performs max pooling within each cell. Empty cells are associated with the zero vector. The pooled features are processed by an encoder-decoder network with skip connections. The decoder has two heads: one for occupancies $\mathbf{O}$ and one for vertex displacements $\mathbf{X}$. All details of the architecture can be found in the supplementary material.

with $t_{\mathbf{m}} \in \{0, 1\}$ denoting the $\mathbf{m}$'th element of $\mathbf{T}$. Note that jointly with the vertex displacement field $\mathbf{X}$, the distribution over topologies $p_{\mathbf{n}}(\mathbf{T})$ at cell $\mathbf{n}$ defines a distribution over triangular meshes within cell $\mathbf{n}$. Considering all cells $\mathbf{n} \in \mathcal{T}$ we obtain a distribution over meshes in the entire grid as

$$p(\{\mathbf{T_n}|\mathbf{n} \in \mathcal{T}\}) = \prod_{\mathbf{n}\in\mathcal{T}} p_{\mathbf{n}}(\mathbf{T_n}) \tag{3}$$

where $\mathcal{T} = \{1, \ldots, N-1\}^3$ and the vertex displacements $\mathbf{X}$ are fixed to the predictions of the neural net.

**Remark:** While the total number of possible topologies within a voxel is $2^8 = 256$, many of them represent disconnected meshes. As those are unlikely to occur in practice given a fine enough voxel resolution, in this paper we only consider the 140 singly connected topologies (highlighted in yellow in Fig. 2) and renormalize (2) accordingly.

## 3.3. Network Architecture

This section describes our complete network architecture which integrates the Differentiable Marching Cubes Layer described in the previous section as a final layer for explicit surface prediction. We adopt an encoder-decoder architecture as illustrated in Fig. 4. The encoder extracts features from the raw observations and the decoder predicts an explicit surface. In this paper we consider a 3D point cloud $\mathbf{P} \in \mathbb{R}^{K\times 3}$ with $K$ points as input. However, note that the encoder could be easily adapted to other types of observations including 3D volumetric information or 2D images.

Our point cloud encoder is a variation of PointNet++ [33] which is invariant to the local point ordering while retaining local information. Similar to PointNet++, we first extract a local feature vector for each point using fully connected layers. The major difference is that our feature representation is tightly coupled with the discrete structure of the voxel grid. While PointNet++ recursively samples points for grouping, we group all points falling into a voxel into one set and apply pooling within this voxel. Thus, we retain the regular grid structure of the decoder which allows for exploiting skip connections in our model (see Fig. 4).

The result of the grid pooling operation is fed into a standard 3D encoder-decoder CNN for increasing the size of the receptive field. This subnetwork is similar to the one used in [11,36] and comprises convolution, pooling and unpooling layers as well as ReLU non-linearities. Following common practice, we exploit skip connections to preserve details. The decoder head is split into two branches, one for estimating the occupancy probabilities $\mathbf{O}$ and one for predicting the vertex displacement field $\mathbf{X}$. A sigmoid layer is added to both $\mathbf{O}$ and $\mathbf{X}$ to ensure valid probabilities between 0 and 1 for $\mathbf{O}$, and valid vertex displacements for $\mathbf{X}$. The distribution over topologies is given by equation (3). For more details we refer to the supplementary material.

## 3.4. Loss Functions

At training time, our goal is to minimize the distance between the ground truth point cloud and the predicted surface mesh $\mathbf{M}$. Note that our model predicts a distribution over surface meshes $p(\mathbf{M})$ thus we minimize the expected surface error. We add additional constraints to regularize the occupancy variables and the smoothness of the estimated mesh. Our loss function decomposes into four parts

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) = w_1 \sum_{\mathbf{n}} \mathcal{L}^{\text{mesh}}_{\mathbf{n}}(\boldsymbol{\theta}) + w_2\, \mathcal{L}^{\text{occ}}(\boldsymbol{\theta}) + \\ w_3 \sum_{\mathbf{n}\sim\mathbf{m}} \mathcal{L}^{\text{smooth}}_{\mathbf{n,m}}(\boldsymbol{\theta}) + w_4 \sum_{\mathbf{n}\sim\mathbf{m}} \mathcal{L}^{\text{curve}}_{\mathbf{n,m}}(\boldsymbol{\theta})\end{aligned} \tag{4}$$

where $\boldsymbol{\theta}$ represents the parameters of the neural network in Fig. 4, $\{w_i\}$ are the weights of the loss function and $\mathbf{n} \sim \mathbf{m}$ denotes the set of adjacent cells in the grid. Each part of this loss function will be described in the following paragraphs.

**Point to Mesh Loss:** We first introduce a geometric loss which measures the compatibility of the predicted 3D surface mesh with respect to the observed 3D points. Let $\mathbf{Y}$ denote the set of observed 3D points (i.e., the ground truth) and let $\mathbf{Y_n} \subseteq \mathbf{Y}$ denote the set of observed points falling into cell $\mathbf{n}$. As our model predicts a distribution of topologies $p_\mathbf{n}(\mathbf{T})$ and hence also meshes at every cell $\mathbf{n}$, we seek to minimize the expected error with respect to this distribution. More formally, we have

$$\mathcal{L}_\mathbf{n}^{\text{mesh}}(\boldsymbol{\theta}) = \mathbb{E}_{p_\mathbf{n}(\mathbf{T}|\boldsymbol{\theta})}\left[\sum_{\mathbf{y}\in\mathbf{Y_n}} \Delta(\mathbf{M_n}(\mathbf{T}, \mathbf{X}(\boldsymbol{\theta})), \mathbf{y})\right] \quad (5)$$

where $\mathbf{y} \in \mathbb{R}^3$ is an observed 3D point, $\mathbf{M_n}(\mathbf{T}, \mathbf{X})$ represents the mesh induced by topology $\mathbf{T}$ and vertex displacement field $\mathbf{X}$ at cell $\mathbf{n}$, and $\Delta(\mathbf{M}, \mathbf{y})$ denotes the point-to-mesh distance if . The point-to-mesh distance is calculated by finding the triangle closest to $\mathbf{y}$ in terms of euclidean ($\ell_2$) distance. Note that in contrast to losses defined on implicit surface representations (e.g., TSDF), the loss in (5) directly measures the geometric error of the inferred mesh.

While (5) ensures that the inferred mesh covers all observations the converse is not true. That is, surface predictions far from the observations are not penalized as long as all observations are covered by the predicted surface mesh. Unfortunately, such a penalty is not feasible in our case as the ground truth may be incomplete. We therefore add a small constant loss on all non-empty topologies for cells without observed points. Moreover, we introduce additional loss functions that prefer simple solutions in the following paragraphs. In particular, these constraints enforce free-space at the boundary of the volume and smoothness of the surface.

**Occupancy Loss:** As mentioned above, the occupancy status is ambiguous when considering unstructured 3D point clouds as observations. That is, flipping the occupied with the free voxels will result in exactly the same geometric loss as only the distance to the surface can be measured, but no information about what is inside or outside is present in the data. However, we observe that for most scenes objects are surrounded by free space, thus we can safely assume that the 6 faces of the cube bounding the 3D scene are unoccupied. Defining a prior for occupied voxels is more challenging. One could naïvely assume that the center of the bounding cube must be occupied, yet this is not true in general. Thus, we relax this assumption by encouraging a sub-volume inside the scene to be occupied. More formally, we have:

$$\mathcal{L}^{\text{occ}}(\boldsymbol{\theta}) = \frac{1}{|\Gamma|}\sum_{\mathbf{n}\in\Gamma} o_\mathbf{n}(\boldsymbol{\theta}) + w(1 - \frac{1}{|\Omega|}\sum_{\mathbf{n}\in\Omega} o_\mathbf{n}(\boldsymbol{\theta})) \quad (6)$$

where $\Gamma$ denotes the boundary of the scene cube (i.e., all voxels on its six faces) and $\Omega$ denotes a sub-volume inside the cube (e.g., half the size of the scene cube). Minimizing the first term of (6) encourages the boundary voxels to become unoccupied. Minimizing the second term enforces a region within the scene cube to become occupied depending on the adaptive weight $w$, which decreases with the number of high confident occupied voxels in the scene.

**Smoothness Loss:** Note that both $\mathcal{L}^{\text{mesh}}$ as well as $\mathcal{L}^{\text{occ}}$ act only locally on the volume. To propagate occupancy information within the volume, we therefore introduce an additional smoothness loss. In particular, we assume that the majority of all neighboring voxels take the same occupancy state. This assumption is justified by the fact that transitions happen only at the surface of an object (covering the minority of voxels). We therefore introduce the following pairwise loss, encouraging occupancy smoothness:

$$\mathcal{L}_{\mathbf{n},\mathbf{m}}^{\text{smooth}} = |o_\mathbf{n}(\boldsymbol{\theta}) - o_\mathbf{m}(\boldsymbol{\theta})| \quad (7)$$

**Curvature Loss:** Similarly to the smoothness loss on the occupancy variables we can encourage smoothness of the predicted mesh geometry. This is particularly important if the ground truth point cloud is sparse and noisy as assumed in this paper. We therefore add a curvature loss which enforces smooth transitions between adjacent cells by minimizing the expected difference in normal orientation:

$$\mathcal{L}_{\mathbf{n},\mathbf{m}}^{\text{curve}}(\boldsymbol{\theta}) = \mathbb{E}_{p_{\mathbf{n},\mathbf{m}}(\mathbf{T},\mathbf{T}'|\boldsymbol{\theta})}\left[\varphi_{\mathbf{n},\mathbf{m}}(\mathbf{T}, \mathbf{T}', \mathbf{X}(\boldsymbol{\theta}))\right] \quad (8)$$

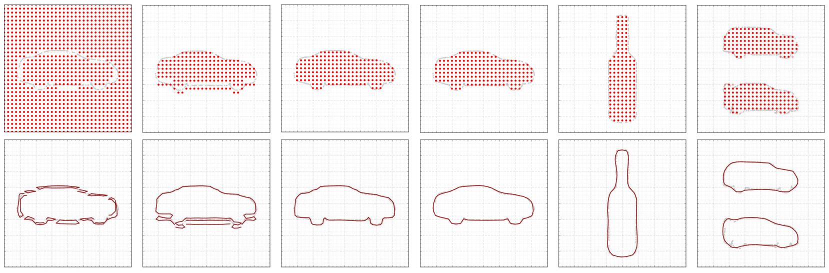
Here, $p_{\mathbf{n},\mathbf{m}}(\mathbf{T}, \mathbf{T}'|\boldsymbol{\theta}) = p_\mathbf{n}(\mathbf{T}|\boldsymbol{\theta})\, p_\mathbf{m}(\mathbf{T}'|\boldsymbol{\theta})$ is the joint distribution over the topologies of voxel $\mathbf{n}$ and voxel $\mathbf{m}$. Furthermore, $\varphi_{\mathbf{n},\mathbf{m}}(\cdot)$ denotes a function which returns the squared $\ell_2$ distance between the normals of the faces in cell $\mathbf{n}$ and $\mathbf{m}$ which are connected by a joint edge, and 0 if the faces in both cells are not topologically connected.

## 4. Experimental Evaluation

In this section, we first thoroughly evaluate the effectiveness and robustness of the proposed method in 2D. Then we demonstrate the ability of our method to predict 3D meshes from 3D point clouds.

### 4.1. Model Validation in 2D

For clarity, we validate our model in 2D before we consider the 3D case. In 2D, the total number of topologies reduces to $2^4 = 16$ as illustrated in the supplementary material. We rendered silhouettes of 1547 different car instances from ShapeNet [5], which we split into 1237 training samples and 310 test samples. We randomly sampled 300 points from the silhouette boundaries which we feed as input to the network. We use a voxel grid of size $N \times N \times N$ with $N = 32$ throughout all of our experiments. All other hyperparameters are specified in the supplementary material. For evaluation, we use Chamfer distance, accuracy and completeness. We follow common practice [22] and specify all measures as distances, thus lower accuracy / completeness values indicate better results.


(a) $\mathcal{L}^{\text{mesh}}$ (b) $+\mathcal{L}^{\text{occ}}$ (c) $+\mathcal{L}^{\text{smooth}}$ (d) $+\mathcal{L}^{\text{curve}}$ (e) Car→Bot. (f) Topology

| | Chamfer | Acc. | Comp. | Hamming |
|---|---|---|---|---|
| $\mathcal{L}^{\text{mesh}}$ | 0.339 | 0.388 | 0.289 | 83.69% |
| $+\mathcal{L}^{\text{occ}}$ | 0.357 | 0.429 | 0.285 | 4.67% |
| $+\mathcal{L}^{\text{smooth}}$ | 0.240 | 0.224 | 0.255 | 0.56% |
| $+\mathcal{L}^{\text{curve}}$ | 0.245 | 0.219 | 0.272 | 0.53% |

(g) Quantitative Results (Lower is Better)

Figure 5: **2D Ablation Study.** (a)-(d)+(g) show our results when incrementally adding the loss functions of (4). (e)+(f) demonstrate the ability of our model to generalize to novel categories (train: car, test: bottle) and more complex surface topologies (in this case, two separated objects). The top row shows the input points in gray and the estimated occupancy field $\mathbf{O}$ with red indicating occupied voxels. The bottom row shows the most probable surface $\mathbf{M}$ in red.

**Ablation Study:** We first validate the effectiveness of each component of our loss function in Fig. 5. Starting with the point to mesh loss $\mathcal{L}^{\text{mesh}}$, we incrementally add the occupancy loss $\mathcal{L}^{\text{occ}}$, smoothness loss $\mathcal{L}^{\text{smooth}}$ and curvature loss $\mathcal{L}^{\text{curve}}$. We evaluate the quality of the predicted mesh by measuring the Chamfer distance in voxels, which considers both accuracy and completeness of the predicted mesh. For this experiment, we also evaluated the Hamming distance between our occupancy prediction and the ground truth occupancy to assess the ability of our model in separating inside from outside. Using only $\mathcal{L}^{\text{mesh}}$, the network predicts multiple surfaces around the true surface and fails to predict occupancy (a). Adding the occupancy loss $\mathcal{L}^{\text{occ}}$ allows the network to separate inside from outside, but still leads to fragmented surface boundaries (b). Adding the smoothness loss $\mathcal{L}^{\text{smooth}}$, removes these fragmentations (c). The curvature loss $\mathcal{L}^{\text{curve}}$ further enhances the smoothness of the surface without decreasing performance. Thus, we adopt the full model in the following evaluation.

**Generalization & Topology:** To demonstrate the flexibility of our approach, we apply our model trained on the category "car" to point clouds from the category "bottle". As evidenced by Fig. 5e, our model generalizes well to novel categories; it learns local shape representations rather than capturing purely global shape properties. Fig. 5f shows that our method, trained and tested with multiple separated car instances also handles complex topologies, correctly separating inside from outside, even when the center voxel is not occupied, validating the robustness of our occupancy loss.

**Model Robustness:** In practice, 3D point cloud measurements are often noisy or incomplete due to sensor occlusions. In this section, we demonstrate that our method is able to reconstruct surfaces even in the presence of noisy and incomplete observations. Note that this is a challenging problem which is typically not considered in learning-based approaches to 3D reconstruction which assume that the ground truth is densely available. We vary the level of noise and completeness in Table 1 and Table 2. For moderate levels of noise, the predicted mesh degrades only slightly. Moreover, our model correctly predicts the shape of the car in Table 2 even though information within an angular range of up to 45° was not available during training.

| | Chamfer | Accuracy | Complete. |
|---|---|---|---|
| $\sigma = 0.00$ | 0.245 | 0.219 | 0.272 |
| $\sigma = 0.15$ | 0.246 | 0.219 | 0.273 |
| $\sigma = 0.30$ | 0.296 | 0.267 | 0.325 |

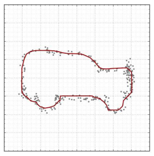


Table 1: **Robustness wrt. Noisy Ground Truth.**

| | Chamfer | Accuracy | Complete. |
|---|---|---|---|
| $\theta = 15°$ | 0.234 | 0.210 | 0.257 |
| $\theta = 30°$ | 0.250 | 0.227 | 0.273 |
| $\theta = 45°$ | 0.308 | 0.261 | 0.354 |

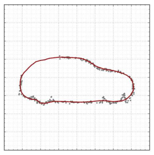


Table 2: **Robustness wrt. Incomplete Ground Truth.**

## 4.2. 3D Shape Prediction from Point Clouds

In this section, we verify the main hypothesis of this paper, namely if end-to-end learning for 3D shape prediction is beneficial wrt. regressing an auxiliary representation and extracting the 3D shape in a postprocessing step. Towards this goal, we compare our model to two baseline methods which regress an implicit representation as widely adopted in the 3D deep learning literature [7, 13, 34, 44, 45], as well as to the well-known Screened Poisson Surface Reconstruction (PSR) [25]. Specifically, given the same point cloud encoder as introduced in Section 3.3, we construct two baselines which predict occupancy and Truncated Signed Distance Functions (TSDFs), respectively, followed by classical Marching Cubes (MC) for extracting the meshes. For a fair comparison, we use the same decoder architecture as our occupancy branch and predict at the same resolution (32 × 32 × 32 voxels). We apply PSR with its default pa-

| Resolution | Method | Chamfer | Accuracy | Complete. |
|---|---|---|---|---|
| $32^3$ | Occ. + MC | 0.407 | 0.246 | 0.567 |
| | TSDF + MC | 0.412 | 0.236 | 0.588 |
| | wTSDF + MC | 0.354 | 0.219 | 0.489 |
| | PSR-5 | 0.352 | 0.405 | 0.298 |
| | Ours | **0.218** | **0.182** | **0.254** |
| $256^3$ | PSR-8 | 0.198 | 0.196 | 0.200 |

Table 3: **3D Shape Prediction from Point Clouds.**

rameters[3]. While the default resolution of the underlying grid (with reconstruction depth $d = 8$) is $256 \times 256 \times 256$ we also evaluate PSR with $d = 5$ (and hence a $32 \times 32 \times 32$ grid as in our method) for a fair comparison.

Again, we conduct our experiments on the ShapeNet dataset, but this time we directly use the provided 3D models. More specifically, we train our models jointly on objects from 3 classes (bottle, car, sofa). As ShapeNet models comprise interior faces such as car seats, we rendered depth images and applied TSDF fusion at a high resolution ($128 \times 128 \times 128$ voxels) for extracting clean meshes and occupancy grids. We randomly sampled points on these meshes which are used as input to the encoder as well as observations. Note that training the implicit representation baselines requires dense ground truth of the implicit surface / occupancy grid while our approach only requires a sparse unstructured 3D point cloud for supervision. For the input point cloud we add Gaussian noise with $\sigma = 0.15$ voxels.

Table 3 shows our results. All predicted meshes are compared to the ground truth mesh extracted from the TSDF at $128 \times 128 \times 128$ voxels resolution. Here, wTSDF refers to a TSDF variant where higher importance is given to voxels closer to the surface resulting in better meshes.

Our method outperforms both baseline methods and PSR in all three metrics given the same resolution. This validates our hypothesis that directly optimizing a surface loss leads to better surface reconstructions. Note that our method infers occupancy using only unstructured points as supervision while both baselines require this knowledge explicitly.

A qualitative comparison is shown in Fig. 6. Our method significantly outperforms the baseline methods in reconstructing small details (e.g., wheels of the cars in rows 1-4) and thin structures (e.g., back of the sofa in rows 6+8). The reason for this is that implicit representations require discretization of the ground truth while our method does not. Furthermore, the baseline methods fail completely when the ground truth mesh is not closed (e.g., car underbody is missing in row 4) or has holes (e.g., car windows in row 2). In this case, large portions of the space are incorrectly labeled free space. While the baselines use this information directly as training signal, our method uses a surface-based

[3]PSR: https://github.com/mkazhdan/PoissonRecon;
We use Meshlab to estimate normal vectors as input to PSR.

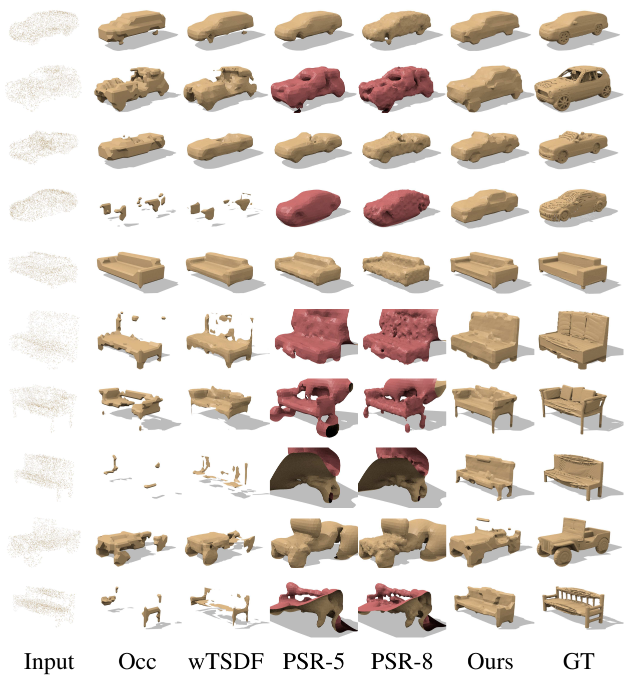


Figure 6: **3D Shape Prediction from Point Clouds.** Surfaces are colored: the outer surface is yellow, the inner red.

loss. Thus it is less affected by errors in the occupancy ground truth. Even though PSR-8 beats our method on completeness given its far higher resolution, it is less robust to noisy inputs compared to PSR-5, while our method handles the trade-off between reconstruction and robustness more gracefully. Furthermore, PSR sometimes flips inside and outside (rows 2+4+6+7) as estimating oriented normal vectors from a sparse point set is a non-trivial task.

We also provide some failure cases of our method in the last two rows of Fig. 6. Our method might fail on very thin surfaces (row 9) or connect disconnected parts (row 10) although in both cases our method still convincingly outperforms the other methods. Those failures are caused by the rather low-resolution output (a $32^3$ grid), which could be addressed using octree networks [18, 35, 36, 39].

## 5. Conclusion

We proposed a flexible framework for learning 3D mesh prediction. We demonstrated that training the surface prediction task end-to-end leads to more accurate and complete reconstructions. Moreover, we showed that surface-based supervision results in better predictions in case the ground truth 3D model is incomplete. In future work, we plan to adapt our method to higher resolution outputs using octrees techniques [18, 36, 39] and integrate our approach with other input modalities like the ones illustrated in Fig. 1.

**Acknowledgements:** Yiyi Liao was partially supported by NSFC under grant U1509210.

## References

[1] S. Bao, M. Chandraker, Y. Lin, and S. Savarese. Dense object reconstruction with semantic priors. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013. 1

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 23:2001, 1999. 1

[3] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *Signal Processing Magazine*, 34(4):18–42, 2017. 3

[4] F. Calakli and G. Taubin. SSD: smooth signed distance surface reconstruction. *Computer Graphics Forum*, 30(7):1993–2002, 2011. 2

[5] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *arXiv.org*, 1512.03012, 2015. 1, 6

[6] S. Choi, Q. Zhou, S. Miller, and V. Koltun. A large dataset of object scans. *arXiv.org*, 1602.02481, 2016. 1

[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 *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 1, 2, 7

[8] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *ACM Trans. on Graphics (SIGGRAPH)*, 1996. 2, 3

[9] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Niessner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[10] A. Dai, C. R. Qi, and M. Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 3

[11] A. Dosovitskiy, P. Fischer, E. Ilg, P. Haeusser, C. Hazirbas, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2015. 5

[12] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[13] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016. 2, 7

[14] K. Guo, D. Zou, and X. Chen. 3d mesh labeling via deep convolutional neural networks. In *ACM Trans. on Graphics (SIGGRAPH)*, 2015. 3

[15] F. Güney and A. Geiger. Displets: Resolving stereo ambiguities using object knowledge. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1

[16] C. Haene, N. Savinov, and M. Pollefeys. Class specific 3d object shape priors using surface normals. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. 1

[17] C. Haene, C. Zach, A. Cohen, R. Angst, and M. Pollefeys. Joint 3D scene reconstruction and class segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013. 1

[18] C. Häne, S. Tulsiani, and J. Malik. Hierarchical surface prediction for 3d object reconstruction. *arXiv.org*, 1704.00710, 2017. 1, 2, 8

[19] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *arXiv.org*, 1703.06870, 2017. 1

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[21] H. Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 30(2):328–341, 2008. 1

[22] R. R. Jensen, A. L. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs. Large scale multi-view stereopsis evaluation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. 6

[23] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. *arXiv.org*, 1712.06584, 2017. 3

[24] A. Kar, S. Tulsiani, J. Carreira, and J. Malik. Category-specific object reconstruction from a single image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1

[25] M. M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Trans. on Graphics (SIGGRAPH)*, 32(3):29, 2013. 2, 7

[26] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Trans. on Graphics (SIGGRAPH)*, 36(4), 2017. 1

[27] C. Kong, C.-H. Lin, and S. Lucey. Using locally corresponding cad models for dense 3d reconstructions from a single image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[28] L. Ladicky, O. Saurer, S. Jeong, F. Maninchedda, and M. Pollefeys. From point clouds to mesh using regression. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 2

[29] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM Trans. on Graphics (SIGGRAPH)*, 1987. 2, 3, 4

[30] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2015. 2

[31] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[32] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[33] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 2, 5

[34] D. J. Rezende, S. M. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3d structure from images. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 1, 2, 3, 7

[35] G. Riegler, A. O. Ulusoy, H. Bischof, and A. Geiger. OctNetFusion: Learning depth fusion from data. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2017. 1, 2, 3, 8

[36] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 5, 8

[37] T. Schöps, J. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[38] S. Song and J. Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 2

[39] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 1, 2, 8

[40] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[41] A. O. Ulusoy, M. Black, and A. Geiger. Patches, planes and probabilities: A non-local prior for volumetric 3d reconstruction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[42] A. O. Ulusoy, M. Black, and A. Geiger. Semantic multi-view stereo: Jointly estimating objects and voxels. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[43] P. Wang, Y. Gan, Y. Zhang, and P. Shui. 3d shape segmentation via shape fully convolutional networks. *Computers & Graphics*, 1702.08675, 2017. 3

[44] 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 *Advances in Neural Information Processing Systems (NIPS)*, 2016. 2, 7

[45] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 2, 7

[46] K. Yamaguchi, D. McAllester, and R. Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2014. 1

[47] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust tv-l1 range image integration. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2007. 1