Coupling Explicit and Implicit Surface Representations for Generative 3D Modeling

Omid Poursaeed[1,2], Matthew Fisher[3], Noam Aigerman[3], and Vladimir G. Kim[3]

[1]Cornell University [2]Cornell Tech [3]Adobe Research

Abstract. We propose a novel neural architecture for representing 3D surfaces, which harnesses two complementary shape representations: (i) an explicit representation via an atlas, i.e., embeddings of 2D domains into 3D; (ii) an implicit-function representation, i.e., a scalar function over the 3D volume, with its levels denoting surfaces. We make these two representations synergistic by introducing novel consistency losses that ensure that the surface created from the atlas aligns with the level-set of the implicit function. Our hybrid architecture outputs results which are superior to the output of the two equivalent single-representation networks, yielding smoother explicit surfaces with more accurate normals, and a more accurate implicit occupancy function. Additionally, our surface reconstruction step can directly leverage the explicit atlas-based representation. This process is computationally efficient, and can be directly used by differentiable rasterizers, enabling training our hybrid representation with image-based losses.

1 Introduction

Many applications rely on a neural network to generate a 3D geometry [12,8], where early approaches used point clouds [1], uniform voxel grids [18], or template mesh deformations [2] to parameterize the outputs. The main disadvantage of these representations is that they rely on a pre-selected discretization of the output, limiting network's ability to focus its capacity on high-entropy regions. Several recent geometry learning techniques address this limitation by representing 3D shapes as *continuous* mappings over vector spaces. Neural networks learn over a manifold of these mappings, creating a mathematically elegant and visually compelling generative models. Two prominent alternatives have been proposed recently.

The explicit surface representation defines the surface as an atlas – a collection of *charts*, which are maps from 2D to 3D, $\{f_i : \Omega_i \subset \mathbb{R}^2 \rightarrow \mathbb{R}^3\}$, with each chart mapping a 2D patch Ω_i into a part of the 3D surface. the surface S is then defined as the union of all 3D patches, $S = \cup_i f_i(\Omega_i)$. In the context of neural networks, this representation has been explored in a line of works considering atlas-based architectures [11,34] which exactly represent surfaces by having the network predict the charts $\{f_i^{\mathbf{x}}\}$, where the network also takes latent code, $\mathbf{x} \in \mathcal{X}$, as input, to describe the target shape. These predicted charts can then be queried at arbitrary 2D points, enabling approximating the resulting

surface with, e.g., a polygonal mesh, by densely sampling the 2D domain with the vertices of a mesh, and then mapping the resulting mesh to 3D via $f_i^{\mathbf{x}}$. This reconstruction step is suitable for an end-to-end learning pipeline where the loss is computed over the resulting surface. It can also be used as an input to a differentiable rasterization layer in case image-based losses are desired. On the other hand, the disadvantage of atlas-based methods is that the resulting surfaces tend to have visual artifacts due to inconsistencies at patch boundaries.

The implicit surface representation defines a volumetric function $g : \mathbb{R} \to \mathbb{R}^3$. This function is called an implicit function, with the surface S defined as its zero level set, $S = \{p \in \mathbb{R}^3 | g(p) = 0\}$. Many works train networks to predict implicit functions, either as signed distance fields [25,5], or simply occupancy values [24]. They also typically use shape descriptor, $\hat{\mathbf{x}} \in \hat{\mathcal{X}}$, as additional input to express different shapes: $g^{\hat{\mathbf{x}}}$. These methods tend to produce visually appealing results since they are smooth with respect to the 3D volume. They suffer from two main disadvantages; first, they do not immediately produce a surface, making them less suitable for end-to-end pipeline with surface-based or image-based losses; second, as observed in [25,5,24], their final output tends to produce a higher surface-to-surface distance to ground truth than atlas-based methods.

In this paper we propose to use both representations in a hybrid manner, with our network predicting both an explicit atlas $\{f_i\}$ and an implicit function g. For the two branches of the two representations we use the AtlasNet [11] and OccupancyNet [24] architectures. We use the same loses used to train these two networks (chamfer distance and occupancy, respectively) while adding novel consistency losses that couple the two representations during joint training to ensure that the atlas embedding aligns with the implicit level-set. We show the two representations reinforce one another: OccupancyNet learns to shift its level-set to align it better with the ground truth surface, and AtlasNet learns to align the embedded points and their normals to the level-set. This results in smoother normals that are more consistent with the ground truth for the atlas representation, while also maintaining lower chamfer distance in the implicit representation. Our framework enables a straightforward extraction of the surface from the explicit representation, as opposed to the more intricate marching-cube-like techniques required to extract a surface from the implicit function. This enables us to add image-based losses on the output of a differentiable rasterizer. Even though these losses are only measured over AtlasNet output, we observe that they further improve the results for *both* representations, since the improvements propagate to OccupancyNet via consistency losses. Another advantage of reconstructing surfaces from the explicit representation is that it is an order of magnitude faster than running marching cubes on the implicit representation. We demonstrate the advantage of our joint representation by using it to train 3D-shape autoencoders and reconstruct a surface from a single image. The resulting implicit and explicit surfaces are consistent with each other and quantitatively and qualitatively superior to either of the branches trained in isolation.

2 Related Work

We review existing representations for shape generation that are used within neural network architectures. While target application and architecture details might vary, in many cases an alternative representation can be seamlessly integrated into an existing architecture by modifying the layers of the network that are responsible for generating the output.

Generative networks designed for images operate over regular 2D grids and can directly extend to 3D voxel occupancy grids [18,9,3,8]. These models tend to be coarse and blobby, since the size of the output scales cubically with respect to the desired resolution. Hierarchical models [13,27] alleviate this problem, but they still tend to be relatively heavy in the number of parameters due to multiple levels of resolution. A natural remedy is to only focus on surface points, hence point-based techniques were proposed to output a tensor with a fixed number of 3D coordinates [1,29]. Very dense point clouds are required to approximate high curvature regions and fine-grained geometric details, and thus, point-based architectures typically generate coarse shapes. While polygonal meshes allow non-even tessellation, learning over this domain even with modest number of vertices remains a challenge [7]. One can predict vertex positions of a template [30], but this can only apply to analysis of very homogeneous datasets. Similarly to volumetric cases, one can adaptively refine the mesh [31,33] using graph unpooling layers to add more mesh elements or iteratively refine it via graph convolutional networks [32]. This refinement can be conditioned on images [31,32] or 3D volumes [33]. The main limitation of these techniques is that they discretize the domain in advance and allocate same network capacity to each discrete element. Even hierarchical methods only provide opportunity to save time by not exploring finer elements in feature-less regions. In contrast, continuous, functional representations enable the network to learn the discretization of the output domain.

The explicit continuous representations view 3D shapes as 2D charts embedded in 3D [11,34]. These atlas-based techniques tend to have visual artifacts related to non-smooth normals and patch misalignments. For homogeneous shape collections, such as human bodies, this can be remedied by replacing 2D charts with a custom template (e.g., a human in a T-pose) and enforce strong regularization priors (e.g., isometry) [10], however, the choice of such a template and priors limits expressiveness and applicability of the method to non-homogeneous collections with diverse geometry and topology of shapes.

Another alternative is to use a neural network to model a space probing function that predicts occupancy [24] or clamped signed distance field [25,5] for each point in a 3D volume. Unfortunately, these techniques cannot be trained with surface-based losses and thus tend to perform worse with respect to surface-to-surface error metrics.

Implicit representations also require marching cubes algorithm [23] to reconstruct the surface. Note that unlike explicit representation, where every sample lies on the surface, marching cubes requires sampling off-surface points in the volume to extract the level set. We found that this leads to a surface recon-

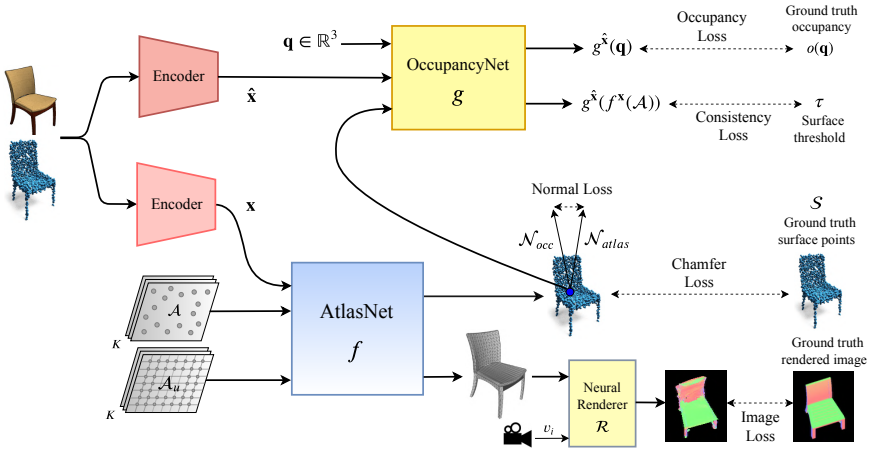

Fig. 1. Model architecture. AtlasNet and OccupancyNet branches of the hybrid model are trained with Chamfer and occupancy losses as well as consistency losses for aligning surfaces and normals. A novel image loss is introduced to further improve generation quality.

struction algorithm that is about an order of magnitude slower than an explicit technique. This additional reconstruction step, also makes it impossible to plug-in the output of the implicit representation into a differentiable rasterizer (e.g., [22,15,19]. We observe that using differentiable rasterizer to enforce additional image-based losses can improve the quality of results. Moreover, adding these losses just for the explicit output, still propagates the improvements to the implicit representation via the consistency losses.

In theory, one could use differentiable version of marching cubes [17] for reconstructing a surface from an implicit representation, however, this has not been used by prior techniques due to cubic memory requirements of this step (essentially, it would limit the implicit formulation to 32^3 grids as argued in prior work [24]). Several recent techniques use ray-casting to sample implicit functions for image-based losses. Since it is computationally intractable to densely sample the volume, these methods either interpolate a sparse set of samples [21] or use LSTM to learn the ray marching algorithm [28]. Both solutions are more computationally involved than simply projecting a surface point using differentiable rasterizer, as enabled by our technique.

3 Approach

We now detail the architecture of the proposed network, as well as the losses used within the training to enforce consistency across the two representations.

3.1 Architecture

Our network simultaneously outputs two surface representations. These two representations are generated from two branches of the network, where each branch uses a state-of-the-art architecture for the target representation.

For the explicit branch, we use AtlasNet [11]. AtlasNet represents K charts with neural functions $\{f_i^{\mathbf{x}}\}_{i=1}^{K}$, where each function takes a shape descriptor vector, $\mathbf{x} \in \mathcal{X}$, and a query point in the unit square, $\mathbf{p} \in [0,1]^2$, and outputs a point in 3D, i.e., $f_i^{\mathbf{x}} : [0,1]^2 \to \mathbb{R}^3$. We also denote the set of 3D points achieved by mapping all 2D points in $\mathcal{A} \subset [0,1]^2$ as $f^{\mathbf{x}}(\mathcal{A})$.

For the implicit branch, we use OccupancyNet [24], learning a neural function $g^{\hat{\mathbf{x}}} : \mathbb{R}^3 \to [0,1]$, which takes a query point $q \in \mathbb{R}^3$ and a shape descriptor vector $\hat{\mathbf{x}} \in \hat{\mathcal{X}}$ and outputs the occupancy value. The point q is considered occupied (i.e., inside the shape) if $g^{\hat{\mathbf{x}}} \geq \tau$, where we set $\tau = 0.2$ following the choice of OccupancyNet.

3.2 Loss Functions

Our approach centers around losses that ensure geometric consistency between the output of the OccupancyNet and AtlasNet modules. We employ these consistency loses along with each branch's original fitting loss (Chamfer and occupancy loss) that was used to train the network in its original paper. Furthermore, we take advantage of AtlasNet's output lending itself to differentiable rendering in order to incorporate a rendering-based loss. These losses are summarized in Figure 1 and detailed below.

Consistency losses. First, to favor consistency between the explicit and implicit representations, we observe that the surface generated by AtlasNet should align with the τ-level set of OccupancyNet:

$$g^{\hat{\mathbf{x}}}\left(f_i^{\mathbf{x}}(\mathbf{p})\right) = \tau, \tag{1}$$

for all charts $f_i^{\mathbf{x}}$ and at every point $\mathbf{p} \in [0,1]^2$. Throughout this subsection we assume that $\mathbf{x}$ and $\hat{\mathbf{x}}$ are describing the same shape.

This observation motivates the following surface consistency loss:

$$\mathcal{L}_{\text{consistency}} = \sum_{\mathbf{p} \in \mathcal{A}} \mathcal{H}\big(g^{\hat{\mathbf{x}}}(f_i^{\mathbf{x}}(\mathbf{p})), \tau\big). \tag{2}$$

where $\mathcal{H}(\cdot,\cdot)$ is the cross entropy function, and $\mathcal{A}$ is the set of sample points in $[0,1]^2$. More specifically, for each point p_i sampled on a 2D patch and its mapped version $g(f(p_i))$, we measure the binary cross-entropy $\tau \log(g(f(p_i))) + (1-\tau)\log(1-g(f(p_i)))$ and sum the losses for all points in the batch. Therefore, the minimum occurs at $g(f(p_i)) = \tau$ for all i. Note that the current loss function only penalizes surface points that are not on the level set of the implicit function, but does not penalize if the OccupancyNet has a zero level set far away from the AtlasNet surface. Such a loss can be added via differentiable layers that either

extract the level set of the OccupancyNet or convert AtlasNet surface to an implicit function. Finding a computationally efficient way of incorporating this loss function is a good venue for future work.

Second, we observe that the gradient of the implicit representation should align with the surface normal of the explicit representation. The normals for both representations are differentiable and their analytic expressions can be defined in terms of gradients of the network. For AtlasNet, we compute surface normal at a point $\mathbf{p}$ as follows:

$$\mathcal{N}_{\text{atlas}} = \frac{\partial f_i^{\mathbf{x}}}{\partial u} \times \frac{\partial f_i^{\mathbf{x}}}{\partial v}\bigg|_{\mathbf{p}} \tag{3}$$

The gradient of OccupancyNet's at a point $\mathbf{q}$, is computed as:

$$\mathcal{N}_{\text{occ}} = \nabla_{\mathbf{q}} g^{\hat{\mathbf{x}}}(\mathbf{q}) \tag{4}$$

We now define the normal consistency loss by measuring the misalignment in their directions (note that the values are normalized to have unit magnitude):

$$\mathcal{L}_{\text{norm}} = \left| 1 - \frac{\mathcal{N}_{\text{atlas}}}{\|\mathcal{N}_{\text{atlas}}\|} \cdot \frac{\mathcal{N}_{\text{occ}}}{\|\mathcal{N}_{\text{occ}}\|} \right| \tag{5}$$

We evaluate this loss only at surface points predicted by the explicit representation (i.e., $\mathcal{N}_{\text{occ}}$ is evaluated at $\mathbf{q} = f^{\mathbf{x}}(\mathbf{p}), \mathbf{p} \in \mathcal{A}$).

Fitting losses. Each branch also has its own fitting loss, ensuring it adheres to the input geometry. We use the standard losses used to train each of the two surface representations in previous works.

For the explicit branch, we measure the distance between the predicted surface and the ground truth in standard manner, using Chamfer distance:

$$\mathcal{L}_{\text{chamfer}} = \sum_{\mathbf{p} \in \mathcal{A}} \min_{\hat{\mathbf{p}} \in S} |f^{\mathbf{x}}(\mathbf{p}) - \hat{\mathbf{p}}|^2 + \sum_{\hat{\mathbf{p}} \in S} \min_{\mathbf{p} \in \mathcal{A}} |f^{\mathbf{x}}(\mathbf{p}) - \hat{\mathbf{p}}|^2, \tag{6}$$

where $\mathcal{A}$ be a set of points randomly sampled from the K unit squares of the charts (here $f^{\mathbf{x}}$ uses one of the neural functions f_i depending on which of the K charts the point $\mathbf{p}$ came from). S is a set of points that represent the ground truth surface.

For the implicit branch, given a set of points $\{\mathbf{q}_i\}_{i=1}^N$ sampled in 3D space, with $o(\mathbf{q}_i)$ denoting their ground-truth occupancy values, the occupancy loss is defined as:

$$\mathcal{L}_{\text{occ}} = \sum_{i=1}^{N} \mathcal{H}\big(g^{\hat{\mathbf{x}}}(\mathbf{q}_i), o(\mathbf{q}_i)\big) \tag{7}$$

Finally, for many applications visual quality of a rendered 3D reconstruction plays a very important role (e.g., every paper on this subject actually presents a rendering of the reconstructed model for qualitative evaluations). Rendering implicit functions requires complex probing of volumes, while output of the explicit representation can be directly rasterized into an image. Thus, we chose to only

include an image-space loss for the output of the explicit branch, comparing its differentiable rendering to the image produced by rendering the ground truth shape. Note that this loss to still improves the representation learned by the implicit branch due to consistency losses.

To compute the image-space loss we first reconstruct a mesh from the explicit branch. In particular, we sample a set of 2D points $\mathcal{A}_u$ on a regular grid for each of the K unit squares. Each grid defines topology of the mesh, and mapping the corners of all grids with $f^{\mathbf{x}}$ gives a triangular 3D mesh that can be used with most existing differentiable rasterizers $\mathcal{R}$ (we use our own implementation inspired by SoftRas [20]). We render 25 images from different viewpoints produced by the cross product of 5 elevation and 5 azimuth uniformly sampled angles.

The image loss is defined as:

$$\mathcal{L}_{\text{img}} = \frac{1}{25} \sum_{i=1}^{25} \|\mathcal{R}(f^{\mathbf{x}}(\mathcal{A}_u), v_i) - \mathcal{R}(\mathcal{M}_{gt}, v_i)\|^2 \tag{8}$$

in which v_i is the i^{th} viewpoint and $\mathcal{M}_{gt}$ represents the ground truth mesh. Our renderer $\mathcal{R}$ outputs a normal map image (based on per-face normals), since they capture the shape better than silhouettes or gray-shaded images.

Our final loss is a weighted combination of the fitting and consistency losses:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{occ}} + \alpha \cdot \mathcal{L}_{\text{chamfer}} + \beta \cdot \mathcal{L}_{\text{img}} + \gamma \cdot \mathcal{L}_{\text{consistency}} + \delta \cdot \mathcal{L}_{\text{norm}} \tag{9}$$

Since the loss functions measure different quantities with vastly different scales, we set the weights empirically to get the best qualitative and quantitative results on the validation set. We use $\alpha = 2.5 \times 10^4$, $\beta = 10^3$, $\gamma = 0.04$, and $\delta = 0.05$ in all experiments.

3.3 Pipeline and Training

Figure 1 illustrates the complete pipeline for training and inference: given an input image or a point cloud, the two encoders encode the input to shape features, $\mathbf{x}$ and $\hat{\mathbf{x}}$. For the AtlasNet branch, a set of points $\mathcal{A} \subset [0, 1]^2$ is randomly sampled from K unit squares. These points are concatenated with the shape feature $\mathbf{x}$ and passed to AtlasNet. The Chamfer loss is computed between $f^{\mathbf{x}}(\mathcal{A})$ and the ground truth surface points, per Equation 6. For the OccupancyNet branch, similarly to [24], we uniformly sample a set of points $\{q_i\}_{i=1}^N \subset \mathbb{R}^3$ inside the bounding box of the object and use them to train OccupancyNet with respect to the fitting losses. To compute the image loss, the generated mesh $f^{\mathbf{x}}(\mathcal{A}_u)$ and the ground truth mesh $\mathcal{M}_{gt}$ are normalized to a unit cube prior to rendering.

For the consistency loss, the occupancy function $g^{\hat{\mathbf{x}}}$ is evaluated at the points generated by AtlasNet, $f^{\mathbf{x}}(\mathcal{A})$ and then penalized as described in Equation (2). AtlasNet's normals are evaluated at the sample points $\mathcal{A}$. OccupancyNet's normals are evaluated at the corresponding points, $f^{\mathbf{x}}(\mathcal{A})$. These are then plugged into the loss described in Equation (5). We train AtlasNet and OccupancyNet

jointly with the loss function in equation (9), thereby coupling the two branches to one-another via the consistency losses.

Since we wish to show the merit of the hybrid approach, we keep the two branches' networks' architecture and training setup identical to the one used in the previous works that introduced those two networks. For AtlasNet, we sample random 100 points from each of $K = 25$ patches during training. At inference time, the points are sampled on 10×10 regular grid for each patch. For OccupancyNet, we use the 2500 uniform samples provided by the authors [24]. We use the Adam optimizer [16] with learning rates of 6×10^{-4} and 1.5×10^{-4} for AtlasNet (f) and OccupancyNet (g) respectively.

4 Results

We evaluate our network's performance on single view reconstruction as well as on point cloud reconstruction, using the same subset of shapes from ShapeNet [4] as used in Choy et al. [6]. For both tasks, following prior work (e.g., [11,24]), we use simple encoder-decoder architectures. Similarly to [24], we quantitatively evaluate the results using the chamfer-L_1 distance and normal consistency score. The chamfer-L_1 distance is the mean of the accuracy and completeness metrics, with accuracy being the average distance of points on the output mesh to their nearest neighbors on the ground truth mesh, and completeness similarly with switching the roles of source and target point sets. Note that we use the chamfer-L_2 distance for training in order to be consistent with the AtlasNet paper [11]. For evaluation, we use the chamfer-L_1 distance since it is adopted as the evaluation metric in OccupancyNet [24]. The normal consistency score is the mean absolute dot product of normals in the predicted surface and normals at the corresponding nearest neighbors on the true surface.

Single view reconstruction. To reconstruct geometry from a single-view image, we use a ResNet-18 [14] encoder for each of the two branches to encode an input image into a shape descriptor which is then fed to the branch. Using distinct encoders enables model-specific feature extraction, and we found this to slightly outperform a shared encoder. We then train end-to-end with the loss (9), on the dataset of images provided by Choy et al. [6], using batch size of 7. Note that with our method the surface can be reconstructed from either the explicit AtlasNet (AN) branch or the implicit OccupancyNet (ON) branch. We show qualitative results (Figure 2) and error metrics (Table 1) for both branches. The surface generated by our AtlasNet branch, "Hybrid (AN)," provides a visually smoother surface than vanilla AtlasNet (AN), which is also closer to the ground truth – both in terms of chamfer distance, as well as its normal-consistency score. The surface generated by our OccupancyNet branch, "Hybrid (ON)", similarly yields a more accurate surface in comparison to vanilla OccupancyNet (ON). We observe that the hybrid implicit representation tends to be better at capturing thinner surfaces (e.g., see table legs in Figure 2) than its vanilla counterpart; this improvement is exactly due to having the implicit branch indirectly trained with the chamfer loss propagated from the AtlasNet branch.

Metric	**Chamfer-L_1($\times 10^{-1}$)**									
Model	AN	ON	Hybrid		No $\mathcal{L}_{\text{img}}$		No $\mathcal{L}_{\text{norm}}$		No $\mathcal{L}_{\text{img}}, \mathcal{L}_{\text{norm}}$	
Branch			AN	ON	AN	ON	AN	ON	AN	ON
airplane	1.05	1.34	**0.91**	1.03	0.96	1.10	0.95	1.08	1.01	1.17
bench	1.38	1.50	**1.23**	1.26	1.27	1.31	1.26	1.29	1.32	1.38
cabinet	1.75	1.53	1.53	**1.47**	1.57	1.49	1.55	1.49	1.61	1.50
car	1.41	1.49	**1.28**	1.31	1.33	1.37	1.33	1.36	1.37	1.42
chair	2.09	2.06	1.96	**1.95**	2.02	2.01	1.99	1.99	2.04	2.03
display	1.98	2.58	**1.89**	2.14	1.92	2.24	1.90	2.19	1.94	2.29
lamp	3.05	3.68	**2.91**	3.02	2.93	3.09	2.91	3.06	2.99	3.21
sofa	1.77	1.81	**1.56**	1.58	1.61	1.63	1.59	1.61	1.68	1.71
table	1.90	1.82	1.73	**1.72**	1.80	1.78	1.78	1.76	1.83	1.79
telephone	1.28	1.27	**1.17**	1.18	1.22	1.21	1.19	1.19	1.24	1.24
vessel	1.51	2.01	**1.42**	1.53	1.46	1.60	1.46	1.58	1.48	1.69
mean	1.74	1.92	**1.60**	1.65	1.64	1.71	1.63	1.69	1.68	1.77
Metric	**Normal Consistency ($\times 10^{-2}$)**									
Model	AN	ON	Hybrid		No $\mathcal{L}_{\text{img}}$		No $\mathcal{L}_{\text{norm}}$		No $\mathcal{L}_{\text{img}}, \mathcal{L}_{\text{norm}}$	
Branch			AN	ON	AN	ON	AN	ON	AN	ON
airplane	83.6	84.5	85.5	**85.7**	85.3	85.6	84.8	85.3	84.3	85.0
bench	77.9	81.4	81.4	**82.5**	80.9	82.2	80.4	81.9	79.9	81.7
cabinet	85.0	88.4	88.3	**89.1**	88.1	89.0	87.2	88.7	86.8	88.6
car	83.6	85.2	86.2	**86.8**	85.8	86.5	85.3	86.0	84.9	85.8
chair	79.1	82.9	83.5	**84.0**	83.1	83.7	82.4	83.4	82.0	83.2
display	85.8	85.7	**87.0**	86.9	86.7	86.6	86.3	86.1	86.0	85.9
lamp	69.4	75.1	74.9	**76.0**	74.7	75.9	73.3	75.6	72.8	75.4
sofa	84.0	86.7	87.2	**87.5**	86.9	87.4	86.4	87.1	85.9	86.9
table	83.2	85.8	86.3	**87.4**	86.0	87.1	85.3	86.4	84.9	86.1
telephone	92.3	93.9	94.0	**94.5**	93.8	94.4	93.6	94.2	93.3	94.1
vessel	75.6	79.7	79.2	**80.6**	78.9	80.4	77.7	80.0	77.4	79.9
mean	81.8	84.5	84.9	**85.5**	84.6	85.4	83.9	85.0	83.5	84.8

Table 1. Quantitative results on single-view reconstruction. Variants of our hybrid model, with AtlasNet (AN) and OccupancyNet (ON) branches, are compared with vanilla AtlasNet and OccupancyNet using Chamfer-L_1 distance and Normal Consistenty score.

Point cloud reconstruction. As a second application, we train our network to reconstruct a surface for a sparse set of 2500 input points. We encode the set of points to a shape descriptor using a PointNet [26] encoder for each of the two branches, and train the encoder-decoder architecture end-to-end with the loss (9). We train with the same points as [24] with a batch size of 10. See results in Figure 3 and Table 2. As in the single view reconstruction task, the hybrid method surpasses the vanilla, single-branch architectures on average. While there are three categories in which vanilla OccupancyNet performs better, we note that Hybrid AtlasNet consistently outperforms vanilla AtlasNet on all categories. This indicates that the hybrid training is mostly beneficial for

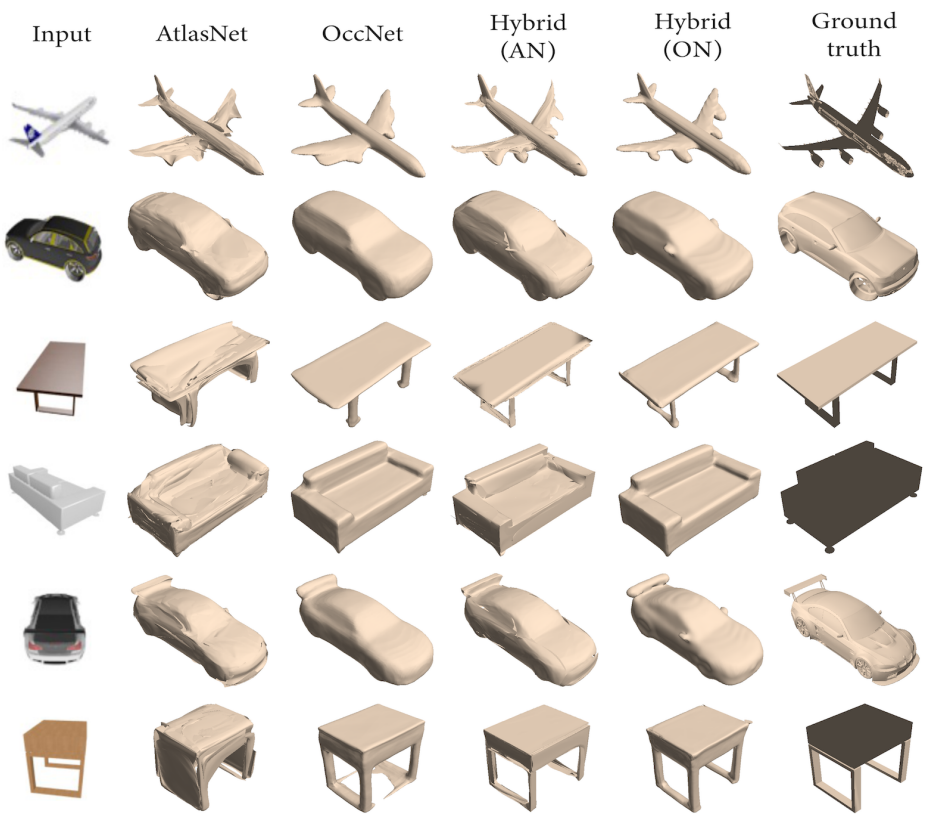

Fig. 2. Comparison of meshes generated by vanilla AtlasNet and OccupancyNet with the AtlasNet (AN) and OccupancyNet (ON) branches of our hybrid model. Compared to their vanilla counterparts, our AtlasNet branch produces results with significantly less oscillatory artifacts, and our OccupancyNet branch produces results that better preserve thin features such as the chair legs.

the implicit representation, and *always* beneficial for the explicit representation; this in turn offers a more streamlined surface reconstruction process. Additional examples are shown in the supplementary material.

Ablation study on the loss functions. We evaluate the importance of the different loss terms via an ablation study for both tasks (see Tables 1, 2). First, we exclude the image-based loss function $\mathcal{L}_{\text{img}}$. Note that even without this loss, hybrid AtlasNet still outperforms vanilla AtlasNet, attributing these improvements mainly to the consistency losses. Removing the normal-consistency loss $\mathcal{L}_{\text{norm}}$ results in decreased quality of reconstructions, especially the accuracy of normals in the predicted surface. Finally, once both terms $\mathcal{L}_{\text{img}}, \mathcal{L}_{\text{norm}}$ are removed, we observe that still, the single level-set consistency term is sufficient to boost the performance within the hybrid training.

Metric	**Chamfer-L_1** ($\times 10^{-3}$)									
Model	AN	ON	Hybrid		No $\mathcal{L}_{\text{img}}$		No $\mathcal{L}_{\text{norm}}$		No $\mathcal{L}_{\text{img}}, \mathcal{L}_{\text{norm}}$	
Branch			AN	ON	AN	ON	AN	ON	AN	ON
airplane	0.17	0.19	**0.15**	0.16	0.16	0.17	0.16	0.17	0.17	0.18
bench	0.49	**0.23**	0.31	0.25	0.34	0.25	0.33	0.24	0.37	0.24
cabinet	0.73	0.56	0.55	**0.51**	0.58	0.52	0.61	0.54	0.63	0.54
car	0.49	0.54	**0.42**	0.44	0.46	0.47	0.44	0.47	0.47	0.50
chair	0.52	**0.30**	0.36	0.33	0.39	0.33	0.38	0.33	0.41	0.32
display	0.61	0.45	0.47	**0.39**	0.50	0.41	0.48	0.40	0.52	0.42
lamp	1.53	1.35	1.42	**1.31**	1.46	1.33	1.44	1.31	1.49	1.34
sofa	0.32	0.34	**0.25**	0.26	0.28	0.29	0.26	0.27	0.30	0.31
table	0.58	0.45	0.46	**0.41**	0.48	0.42	0.47	0.42	0.50	0.43
telephone	0.22	0.12	0.14	**0.10**	0.15	0.10	0.16	0.11	0.18	0.11
watercraft	0.74	**0.38**	0.53	0.42	0.57	0.41	0.54	0.42	0.61	0.40
mean	0.58	0.45	0.46	**0.41**	0.49	0.43	0.48	0.43	0.51	0.44
Metric	**Normal Consistency** ($\times 10^{-2}$)									
Model	AN	ON	Hybrid		No $\mathcal{L}_{\text{img}}$		No $\mathcal{L}_{\text{norm}}$		No $\mathcal{L}_{\text{img}}, \mathcal{L}_{\text{norm}}$	
Branch			AN	ON	AN	ON	AN	ON	AN	ON
airplane	85.4	89.6	88.3	**90.1**	88.1	90.0	87.5	89.7	87.1	89.6
bench	81.5	**87.1**	85.6	86.7	85.3	86.8	85.0	86.8	84.7	86.9
cabinet	87.0	90.6	89.3	**91.1**	89.1	91.0	88.4	90.8	88.1	90.8
car	84.7	87.9	87.5	**88.6**	87.1	88.5	86.7	88.1	86.1	88.0
chair	84.7	**94.9**	88.7	94.3	88.2	94.5	87.6	94.5	87.1	94.6
display	89.7	91.9	91.8	**92.4**	91.5	92.3	91.0	92.2	90.8	92.1
lamp	73.1	79.5	77.1	**79.8**	76.9	79.7	76.4	79.6	76.0	79.5
sofa	89.1	92.2	91.8	**92.8**	91.6	92.7	91.3	92.5	91.0	92.4
table	86.3	91.0	88.8	**91.4**	88.6	91.3	88.3	91.3	88.0	91.2
telephone	95.9	97.3	97.4	**98.0**	97.2	97.8	96.8	97.6	96.5	97.5
watercraft	82.1	**86.7**	84.9	86.5	84.7	86.5	84.3	86.7	84.0	86.7
mean	85.4	89.8	88.3	**90.2**	88.0	90.1	87.6	90.0	87.2	89.9

Table 2. Quantitative results on auto-encoding. Variants of our hybrid model are compared with vanilla AtlasNet and OccupancyNet.

We also provide qualitative examples on how each loss term affects the quality of generated point clouds and meshes. Figure 4 illustrates impact of the image loss. Generated meshes from the AN branch are rendered from different viewpoints as shown in Figure 1. Rendered images are colored based on per-face normals. As we observe, the image loss reduces artifacts such as holes, resulting in more accurate generation. Note that our differentiable renderer uses backface culling, so the back side is not visible as the surface is oriented away from the camera.

We next demonstrate the importance of the normal consistency loss in Figure 5. We colorize the generated point clouds (from the AN branch) with ground truth normals as well as normals computed from the AN and ON branches (Equation 3 and 4). We show the change in these results between two models

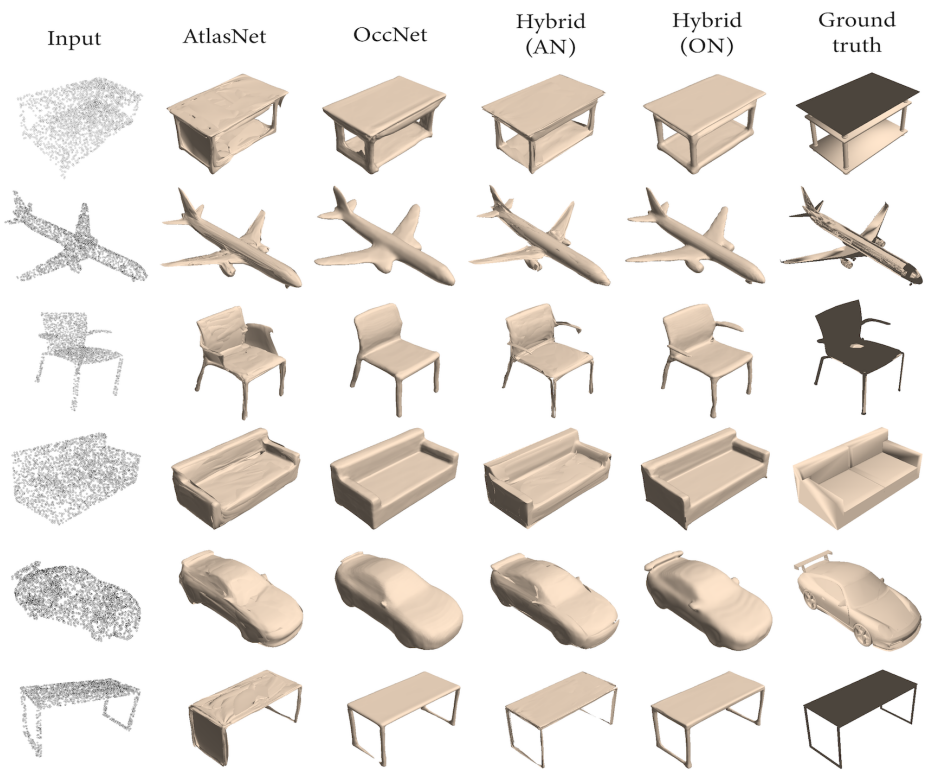

Fig. 3. Reconstructing surfaces from point clouds. Our hybrid approach better reproduces fine features and avoids oscillatory surface normals.

trained with and without normal consistency loss (Equation 5). As we observe, AN's normals are inaccurate for models without the normal loss. This is since the normals consistency loss drives AN's normals to align with ON's normals, which are generally close to the ground truth's.

Finally, Figure 6 exhibits the effect of the consistency loss. We evaluate the resulting consistency by measuring the deviation from the constraint in Equation 1, i.e., the deviation of the predicted occupancy probabilities from the threshold τ, sampled on the predicted AtlasNet surface. We then color the point cloud such that the larger the deviation the redder the point is. Evidently, for models trained without the consistency loss this deviation is significantly larger, than when the consistency loss is incorporated.

Surface reconstruction time. One drawback of the implicit representations is the necessary additional step of extracting the level set. Current approaches require sampling a large number of points near the surface which can be computationally expensive. Our approach allows to circumvent this issue by using the reconstruction from the explicit branch (which is trained to be consistent with the level set of the implicit representation). We see that the surface reconstruct-

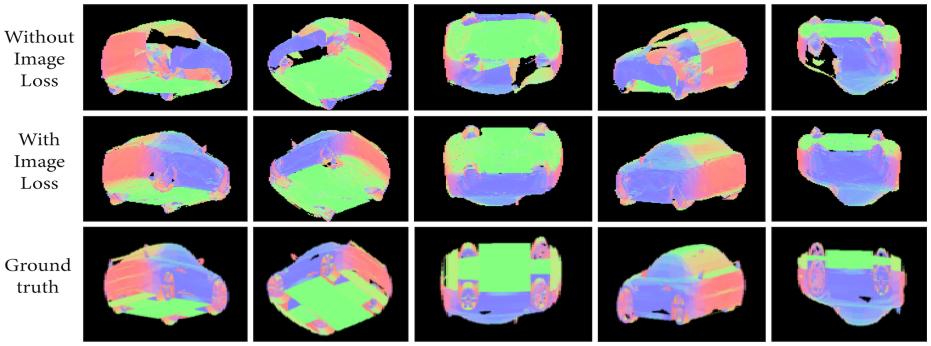

Fig. 4. Impact of the image loss. Rendered images from different viewpoints are shown for models trained with and without the image loss. Evidently, the image loss significantly improves the similarity of the output to the ground truth.

	AN (explicit)	ON (implicit)
Single-view Reconstruction	0.037	0.400
Auto-encoding	0.025	0.428

Table 3. Average surface reconstruction time (in seconds) for the explicit (AN) and implicit (ON) representations. Our approach enables to pick the appropriate reconstruction routine at inference time depending on the application needs, where the quality of the reconstructed surfaces increases due to dual training.

ing time is about order of magnitude faster for explicit representation (Table 3). Our qualitative and quantitative results suggest that the quality of the explicit representation improves significantly when trained with the consistency losses.

5 Conclusion and Future Work

We presented a dual approach for generating consistent implicit/explicit surface representations using AtlasNet and OccupancyNet in a hybrid architecture via novel consistency losses that encourage this consistency. Various tests demonstrate that surfaces generated by our network are of higher quality, namely smoother and closer to the ground truth compared with vanilla AtlasNet and OccupancyNet. A main shortcoming of our method is that it only penalizes inconsistency of the branches, but does not guarantee perfect consistency; nonetheless, the experiments conducted show that both representations significantly improve by using this hybrid approach during training.

We believe this is an important step in improving neural surface generation, and are motivated to continue improving this hybrid approach, by devising tailor-made encoders and decoders for both representations, to optimize their synergy. In terms of applications, we see many interesting future directions that

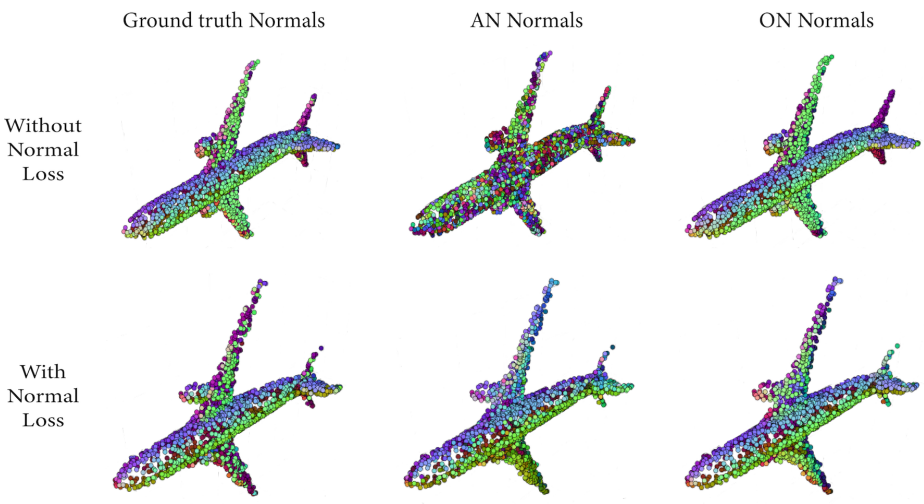

Fig. 5. Impact of the normal consistency loss. The generated point clouds are colored based on the ground truth normals, AtlasNet's (AN) normals, and OccupancyNet's (ON) normals. The results are then compared between a model trained with the normal consistency loss and a model trained without that loss; AN's normals significantly improve when the loss is incorporated, as it encourages alignment with ON's normals which tend to be close to the ground truth normals.

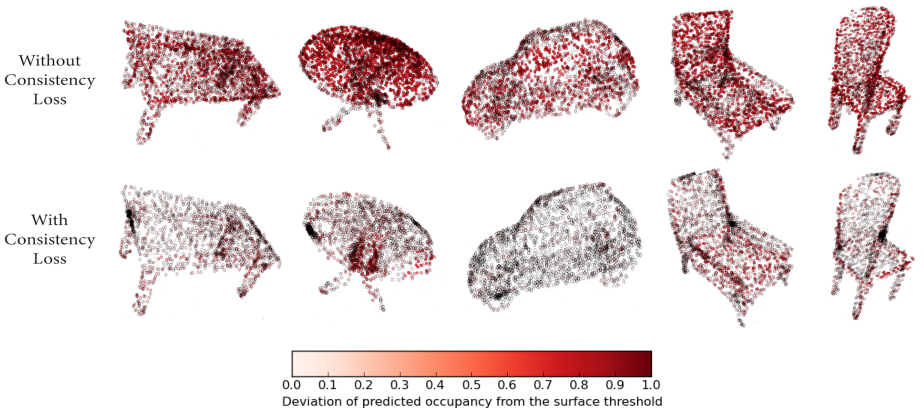

Fig. 6. Impact of the consistency loss. Each point in the generated point cloud is colored based on deviation of its predicted occupancy probability from the threshold τ, with red indicating deviation.

leverage strengths of each approach, such as using AtlasNet to texture OccupancyNet's implicit level set, or using OccupancyNet to train AtlasNet's surface to encapsulate specific input points.

6 Appendix

We visualize additional random results for single-view reconstruction and auto-encoding in Figure 7.

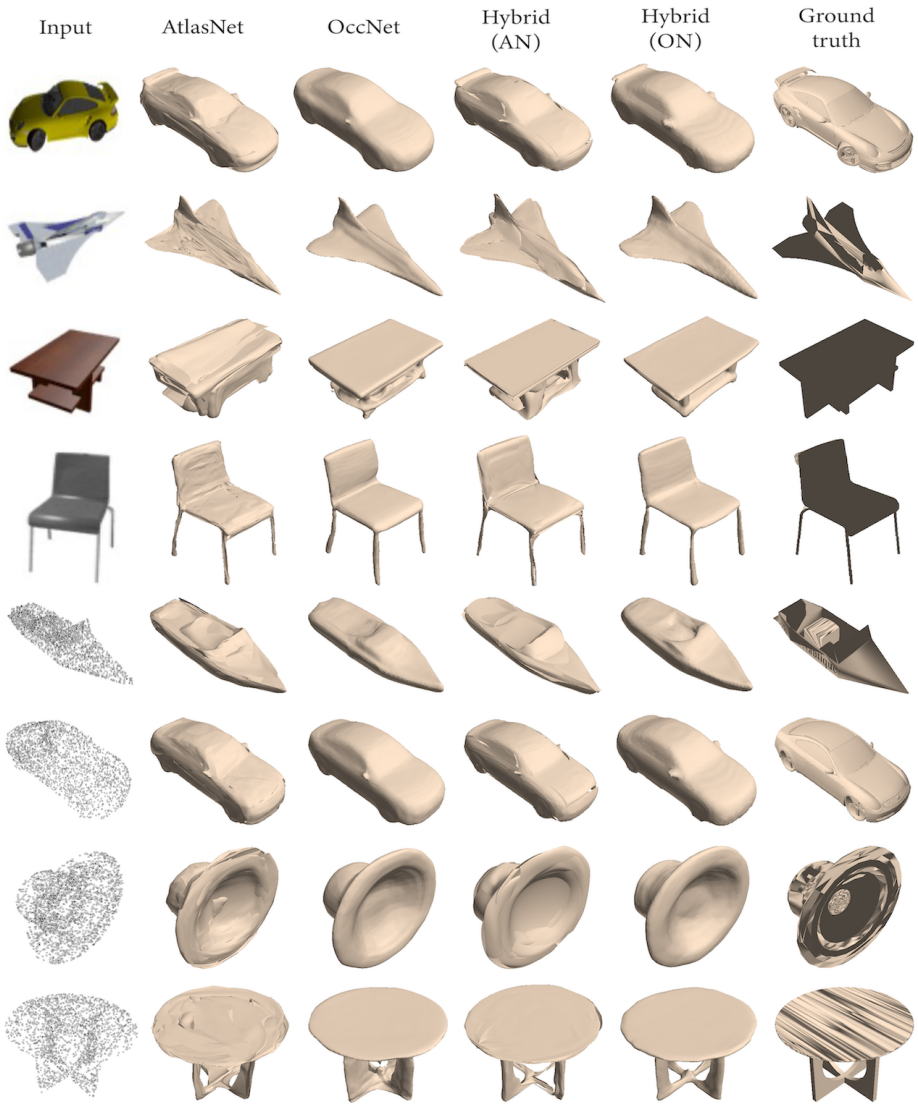

Fig. 7. Random results on reconstructing surfaces from images and point clouds.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.J.: Learning representations and generative models for 3d point clouds. ICML (2018)
2. Ben-Hamu, H., Maron, H., Kezurer, I., Avineri, G., Lipman, Y.: Multi-chart generative surface modeling. SIGGRAPH Asia (2018)
3. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Generative and discriminative voxel modeling with convolutional neural networks. CoRR **abs/1608.04236** (2016), `http://arxiv.org/abs/1608.04236`
4. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
5. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. IEEE Computer Vision and Pattern Recognition (CVPR) (2019)
6. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: European conference on computer vision. pp. 628–644. Springer (2016)
7. Dai, A., Nießner, M.: Scan2mesh: From unstructured range scans to 3d meshes. In: Proc. Computer Vision and Pattern Recognition (CVPR), IEEE (2019)
8. Dai, A., Qi, C.R., Nießner, M.: Shape completion using 3d-encoder-predictor cnns and shape synthesis. Proc. Computer Vision and Pattern Recognition (CVPR), IEEE (2017)
9. Girdhar, R., Fouhey, D.F., Rodriguez, M., Gupta, A.: Learning a predictable and generative vector representation for objects. CoRR **abs/1603.08637** (2016), `http://arxiv.org/abs/1603.08637`
10. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: 3d-coded: 3d correspondences by deep deformation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 230–246 (2018)
11. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: Atlasnet: A papier-mache approach to learning 3d surface generation. arXiv preprint arXiv:1802.05384 (2018)
12. Han, X., Laga, H., Bennamoun, M.: Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era. CoRR **abs/1906.06543** (2019), `http://arxiv.org/abs/1906.06543`
13. Häne, C., Tulsiani, S., Malik, J.: Hierarchical surface prediction for 3d object reconstruction. In: 2017 International Conference on 3D Vision (3DV). pp. 412–420. IEEE (2017)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
15. Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer (2018)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
17. Liao, Y., Donné, S., Geiger, A.: Deep marching cubes: Learning explicit surface representations. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
18. Liu, J., Yu, F., Funkhouser, T.: Interactive 3d modeling with a generative adversarial network. International Conference on 3D Vision (3DV) (2017)
19. Liu, S., Li, T., Chen, W., Li, H.: Soft rasterizer: A differentiable renderer for image-based 3d reasoning. The IEEE International Conference on Computer Vision (ICCV) (Oct 2019)

20. Liu, S., Li, T., Chen, W., Li, H.: Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 7708–7717 (2019)
21. Liu, S., Saito, S., Chen, W., Li, H.: Learning to infer implicit surfaces without 3d supervision. Neural Information Processing Systems (NeurIPS) (Oct 2019)
22. Loper, M.M., Black, M.J.: OpenDR: An approximate differentiable renderer. pp. 154–169 (Sep 2014)
23. Lorensen, W., Cline, H.: Marching cubes: A high resolution 3d surface construction algorithm. In: SIGGRAPH (1987)
24. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019)
25. Park, J.J., Florence, P., Straub, J., Newcombe, R.A., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. CVPR (2019)
26. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660 (2017)
27. Riegler, G., Osman Ulusoy, A., Geiger, A.: Octnet: Learning deep 3d representations at high resolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3577–3586 (2017)
28. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. In: Advances in Neural Information Processing Systems (2019)
29. Su, H., Fan, H., Guibas, L.: A point set generation network for 3d object reconstruction from a single image. CVPR (2017)
30. Tan, Q., Gao, L., Lai, Y.K., Xia, S.: Variational autoencoders for deforming 3d mesh models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018)
31. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2mesh: Generating 3d mesh models from single rgb images. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 52–67 (2018)
32. Wen, C., Zhang, Y., Li, Z., Fu, Y.: Pixel2mesh++: Multi-view 3d mesh generation via deformation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1042–1051 (2019)
33. Wickramasinghe, U., Remelli, E., Knott, G., Fua, P.: Voxel2Mesh: 3D Mesh Model Generation from Volumetric Data. arXiv e-prints arXiv:1912.03681 (Dec 2019)
34. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). vol. 3 (2018)