Deep Implicit Surface Point Prediction Networks

Rahul Venkatesh[1] Tejan Karmali[2] Sarthak Sharma[3] Aurobrata Ghosh[3]
R. Venkatesh Babu[2] László A. Jeni[1*] Maneesh Singh[3*]
[1]Carnegie Mellon University, Pittsburgh, PA, USA [2]Indian Institute of Science, Bengaluru, India
[3]Verisk Analytics, Jersey City, NJ, USA

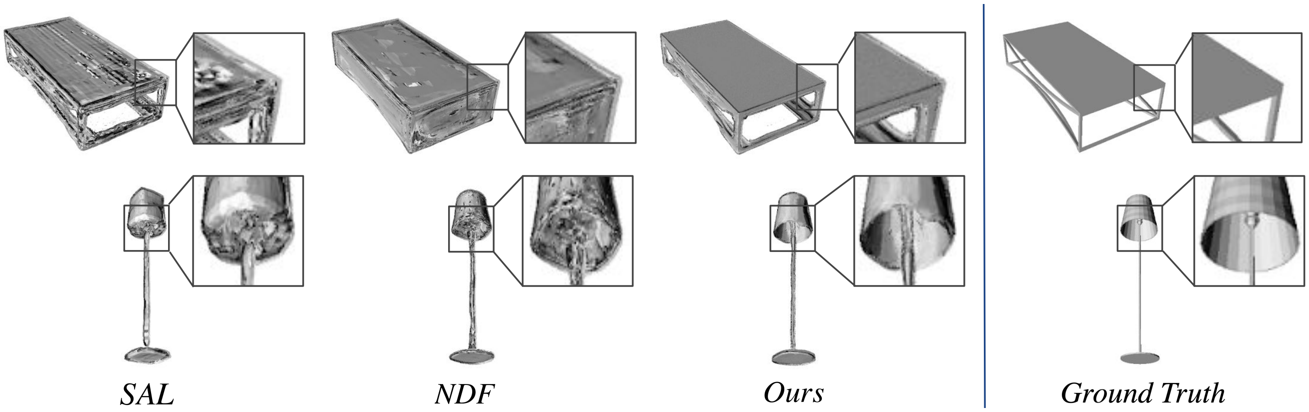


Figure 1: Our novel implicit shape representation can model complex surfaces with high-fidelity. ***Row 1:*** Recovering visually pleasing surfaces in comparison to prior state-of-the-art SAL [2] and NDF [9]. ***Row 2:*** Results on a representative open shape, where we correctly model the shape, as opposed to SAL [2], which closes up regions that are meant to be open.

Abstract

Deep neural representations of 3D shapes as implicit functions have been shown to produce high fidelity models surpassing the resolution-memory trade-off faced by the explicit representations using meshes and point clouds. However, most such approaches focus on representing closed shapes. Unsigned distance function (UDF) based approaches have been proposed recently as a promising alternative to represent both open and closed shapes. However, since the gradients of UDFs vanish on the surface, it is challenging to estimate local (differential) geometric properties like the normals and tangent planes which are needed for many downstream applications in vision and graphics. There are additional challenges in computing these properties efficiently with a low-memory footprint. This paper presents a novel approach that models such surfaces using a new class of implicit representations called the closest surface-point (CSP) representation. We show that CSP allows us to represent complex surfaces of any topology (open or closed) with high fidelity. It also allows for accurate and efficient computation of local geometric properties. We further demonstrate that it leads to efficient implementation of downstream algorithms like sphere-tracing for rendering the 3D surface as well as to create explicit mesh-based representations. Extensive experimental evaluation on the ShapeNet dataset validate the above contributions with results surpassing the state-of-the-art. Code and data are available at https://sites.google.com/view/cspnet.

1. Introduction

High fidelity representation and rendering of potentially open 3D surfaces with complex topology from raw sensor data (images, point clouds) finds application in vision, graphics and animation industry [19]. Therefore, in recent years deep learning based methods for 3D reconstruction of objects have garnered significant interest [36, 32, 29].

Explicit 3D shape representations such as point clouds, voxels, triangles or quad meshes pose challenges in reconstructing surfaces with arbitrary topology [31]. Moreover, the ability to capture details of such representations are limited by predefined structure (like number of vertices for meshes) or memory and computational footprint

[*] indicates two authors equally advised

(for voxels and point clouds). Several implicit shape representations using deep neural networks have been proposed [32, 29, 13, 7, 2, 9] to alleviate these shortcomings.

Recent approaches use a distance function as the implicit representation. For example, DeepSDF [32] use a Signed Distance Function (SDF) as the implicit representation where the sign represents the inside/outside of the surface being modeled. Not only does this limit DeepSDF to modeling closed surfaces, the ground truth needs to be watertight (*closed*) as well. Since most 3D shape datasets [5] have non-watertight (*open*) shapes, preprocessing is needed to artificially close such shapes and make them watertight [29] - a process which is known to result in a loss of fidelity [21]. To overcome this problem, methods such as SAL [2] seek to learn surface representations directly from raw unoriented point clouds. However, such methods also make an assumption that the underlying surface represented by the point cloud is closed, leading to learnt representations necessarily describing closed shapes [3].

NDF [9] overcomes this limitation by using an unsigned distance function (*UDF*) based implicit representation and achieves state-of-the-art performance on 3D shape representation learnt directly from the unprocessed ShapeNet dataset. However, *UDFs* have a fundamental limitation. Since the gradient of the *UDF* vanishes on the surface, direct estimation of local, differential geometric properties like the tangent plane and the surface normal becomes noisy and loses fidelity. This results in a loss of performance on downstream tasks like rendering and surface reconstruction [22] as well as those like registration [34] and segmentation [17] where normal estimates play a vital role.

An additional issue is that for the above methods, the estimation of differential geometric properties needs a backward pass leading to increased memory footprint and time complexity. This becomes a challenge for applications which require fast rendering on devices with limited memory (e.g. tiled rendering on hand-held devices [12]), or for robotics tasks such as real-time path planning where fast normal estimates in 3D space play an essential role [14, 30].

To address these challenges, we introduce a novel implicit shape representation called the *Closest Surface-Point* (*CSP*) function, which for a given query point returns the closest point on the surface. We demonstrate that *CSP* can model open and closed shapes of arbitrary topology, and in contrast to NDF, allows for the easy computation of differential geometric properties like the tangent plane and the surface normal. Moreover, as opposed to existing implicit representations and demonstrated later, it can efficiently recover normal information with a forward pass. A comparative summary of the properties discussed above for *CSP* and the most related art is presented in Table 1. We also present a panel of illustrative results in Fig. 1 which clearly demonstrates the higher fidelity with which complex surfaces are represented by *CSP* when compared with *SAL* and *NDF*.

Method	Learning from Triangle Soups	Representation Power			Single pass normal estimation
		Open Shapes	Complex Topology	High Fidelity	
DeepSDF [32]	✗	✗	✓	✓	✗
SAL [2]	✓	✗	✓	✗	✗
NDF [9]	✓	✓	✓	✗	✗
CSP (Ours)	✓	✓	✓	✓	✓

Table 1: Comparison between *CSP* and closely related arts.

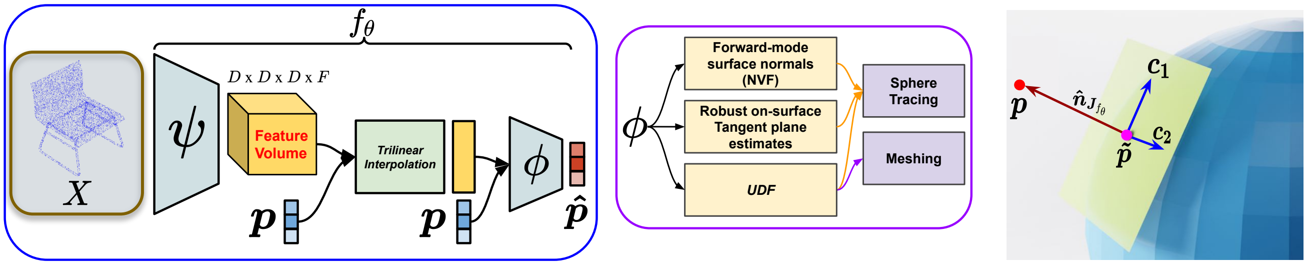
Finally, we show that due to the above benefits, *CSP* is not only a potential method of choice for learning high fidelity 3D representations of complex topologies (open as well as closed) from the raw data, but also for many downstream applications. For this, we present (a) a fast and memory efficient rendering algorithm using an adaptation of sphere-tracing for *CSPs* that leverages the accurate surface normal estimates that *CSP* provides, and, (b) since it is often required to extract an explicit surface representation [1], we present a coarse to fine meshing algorithm for *CSPs*, that can recover high-fidelity meshes faster than prior methods [9]. To summarize, our contributions are:

- *CSP*: A high fidelity representation capable of modelling both open and closed shapes, allowing for efficient estimation of differential geometric properties of the surface (Sec. 3.1) - an advancement over NDF.
- Normal estimation with a forward pass that significantly accelerates speed and memory efficiency of rendering (Sec. 3.2).
- A faster multi-resolution surface extraction technique (Sec. 3.3.2) to extract meshes from *CSP*, achieving better speed and quality than existing techniques [9].

2. Related Work

Explicit Shape Representations. Explicit approaches primarily use voxels, meshes or point clouds for representing 3D shapes. Voxels provide a direct extension of pixels to 3D, allowing easier extension of image processing methods for 3D shape analysis. Several initial shape representation works are built upon this idea [28, 11, 38]. Drawbacks of using voxel representations are limited output resolution, and higher computational and memory requirements. Mesh representations address some of these issues [18, 41, 16, 23], although the details captured are still limited by the choice for the number of vertices. Point clouds provide a more compact and sparser representation of surface geometry, but do not yield high-fidelity reconstruction and cannot reliably model arbitrary topology.

Implicit Shape Representations. Modern implicit shape representation approaches use deep neural network models to implicitly represent a shape by either (1) classifying a (query) point as inside/outside a shape [29] (delineated by the modeled surface), or by (2) the signed (or unsigned) distance of the point to the surface [32], where the sign indi-



(a) Overview of the proposed system

(b) Estimation of local geometric properties

Figure 2: ***Left:*** Network architecture of CSPNet (Sec. 3.1.1). A point cloud X is the input to the volume encoder ψ to obtain a feature volume. The shape decoder ϕ is conditioned on it to obtain closest surface-point $\hat{\boldsymbol{p}}$ for each query point $\boldsymbol{p} \in \mathbb{R}^3$. Next, we show how *CSP* enables extraction of both *UDF*, *NVF*, and tangent plane; which are further utilised for applications like rendering (via Sphere tracing in Sec. 3.3.1) and meshing (in Sec. 3.3.2). ***Right:*** Surface-normal estimation via the method described in Sec. 3.2.1. $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ refer to the basis of the tangent plane and $\hat{\boldsymbol{n}}_{\boldsymbol{J}_{f_\theta}}$ is the normal estimated.

cates inside or outside.

Hybrid explicit/implicit representations are proposed [7, 13], where the implicit function is a union of inside/outside classifier hyper-planes. BSP-Net [7] uses a binary space partitioning network to model a convex decomposition of the 3D shape, the union of which defines a watertight separation of the inside/outside of the shape. CvxNet [13], also proposes a convex decomposition using hyper-planes but with a double representation of a complex primitive.

The methods described above [7, 13, 32, 29, 32] can only represent closed surfaces, with an additional requirement that the training data also comprises of *closed* watertight shapes, which often results in non-trivial loss of fine details [21]. SAL [2] provides a partial solution to this problem by proposing an unsigned similarity function to learn from ground truth unprocessed triangle soups, but eventually infers a shape representation which can only model closed shapes. NDF [9] overcomes this limitation by using a *UDF* to represent both open and closed shapes, but cannot easily provide high fidelity estimates of differential geometry (surface normal, tangent plane) due to the vanishing gradient of the *UDF* on the surface.

In contrast, *CSPs* can model arbitrary topologies (both open and closed) while also allowing for simple, efficient and high-fidelity computation of differential surface geometry as opposed to the prior art [9, 2, 7, 29, 32, 13].

3. Approach

In this section we present the proposed shape representation, and how it can be used for downstream applications. A schematic of the system architecture is presented in Fig. 2. We start below with first defining the proposed implicit shape representation (Sec. 3.1) and deep neural network model for the same (Sec. 3.1.1). Then, we present approaches to estimate the local geometric properties of the surface, e.g. the tangent plane and the surface normal (Sec. 3.2) and finally propose algorithms for using *CSP* for downstream applications like rendering (Sec. 3.3.1) and meshing (Sec. 3.3.2).

3.1. Shape Representation

Given a surface $\mathcal{S} \subset \mathbb{R}^3$ and a (query) point $\boldsymbol{p} \in \mathbb{R}^3$, we define the Closest Surface Point *CSP* as a function $\mathit{CSP}(\boldsymbol{p}) : \mathbb{R}^3 \longmapsto \mathcal{S}$, such that

$$\mathit{CSP}(\boldsymbol{p}) = \tilde{\boldsymbol{p}},\ s.t.\ \tilde{\boldsymbol{p}} = \operatorname*{argmin}_{\boldsymbol{p_s} \in \mathcal{S}} \|\boldsymbol{p} - \boldsymbol{p_s}\|_2 \tag{1}$$

where $\tilde{\boldsymbol{p}}$ is the closest point on the surface $\mathcal{S}$ to the query point $\boldsymbol{p}$. Given the closest surface-point, *UDF* can be trivially calculated as:

$$\mathit{UDF}(\boldsymbol{p}) = \|\boldsymbol{p} - \tilde{\boldsymbol{p}}\|_2 \tag{2}$$

3.1.1 CSPNet: A Deep Neural *CSP* Model

We model *CSP* using a deep neural network which we demonstrate to be robust to training with noisy data generated from a noisy triangle soup. We illustrate the complete architecture in Fig. 2. There are two main components of the proposed CSPNet.

Volume Encoder. For any input 3D shape, the volume encoder ψ produces a feature volume which is isotopic[1] to the volume enclosing the input shape. Each feature voxel encodes properties of the surface from the vantage point of the voxel. For the implementation in this paper, we follow the architecture of Convolutional Occupancy Networks [33] for the volume encoder ψ. The encoder takes as input the entire point cloud for the input shape and produces a volumetric feature encoding. More specifically, PointNet [36] is

[1] Having the same topology as the enclosing volume (not the input 3D shape)

used to encode point features. To get volumetric features, a voxel grid is constructed and voxel features are computed by (average) pooling features for the points that correspond to the voxel under consideration. This is followed by a 3D U-Net which produces the final encoding of the feature volume, resulting in a feature of dimension F for each voxel.

Shape Decoder. The feature corresponding to the input query point $\boldsymbol{p}$ is sampled from the 3D feature volume using trilinear interpolation, and passed into the shape decoder ϕ along with query point $\boldsymbol{p}$. The shape decoder ϕ uses the features encoding the shape to predict the surface point closest to $\boldsymbol{p}$. Here on, we will use f_θ and g_θ to denote the DNN approximations to the *CSP* and the *UDF* functions respectively, where θ denotes the union of parameters of both encoder and decoder. Note that the output of CSPNet, f_θ, directly provides an estimate for *CSP* while the estimate for the *UDF*, g_θ, is obtained as $\|\boldsymbol{p} - f_\theta(\boldsymbol{p})\|_2$ using (2).

3.2. Differential Surface Geometry

For any query point $\boldsymbol{p}$, CSPNet directly provides us with an estimate of both the closest point on the surface $f_\theta(\boldsymbol{p})$ as well as the unsigned distance to it, $g_\theta(\boldsymbol{p})$. However, in addition, a variety of downstream applications in vision [22, 34], robotics [14, 30], graphics [12], and animation [40] need estimates of local differential properties of the surface like the tangent plane and normal at any surface point. We show below how we can easily estimate these properties.

3.2.1 Using the Jacobian

Let $\boldsymbol{p}$ be any query point and $\tilde{\boldsymbol{p}} \in \mathcal{S}$ be its closest point on the surface $\mathcal{S}$. Further, let $\boldsymbol{J}_{f_\theta}(\tilde{\boldsymbol{p}})$ denote the Jacobian of f_θ at $\tilde{\boldsymbol{p}}$. Let δ be the unsigned distance from $\boldsymbol{p}$ to $\tilde{\boldsymbol{p}}$ and $\boldsymbol{d}$ be the surface normal at $\tilde{\boldsymbol{p}}$. Then, we get the following approximation using the first-order Taylor series expansion:

$$\begin{aligned} \boldsymbol{p} &= \tilde{\boldsymbol{p}} + \delta \cdot \boldsymbol{d} \\ f_\theta(\boldsymbol{p}) &= f_\theta(\tilde{\boldsymbol{p}} + \delta \cdot \boldsymbol{d}) \\ \tilde{\boldsymbol{p}} &\approx \tilde{\boldsymbol{p}} + \delta \cdot \boldsymbol{J}_{f_\theta}(\tilde{\boldsymbol{p}}) \cdot \boldsymbol{d} \\ 0 &\approx \delta \cdot \boldsymbol{J}_{f_\theta}(\tilde{\boldsymbol{p}}) \cdot \boldsymbol{d} \end{aligned} \tag{3}$$

The last equation shows that (to a first order approximation of the surface), the surface normal $\boldsymbol{d}$ lies in the null space of the Jacobian $\boldsymbol{J}_{f_\theta}(\tilde{\boldsymbol{p}})$ while the span of the Jacobian provides the tangent space of the surface. This is illustrated in the Fig. 2b and is intuitively clear since along the direction perpendicular to the surface, the CSP function does not change, giving the same closest surface point. The tangent space and the normal to the surface both can be estimated using singular value decomposition (SVD).

However, computation of Jacobian requires a backward pass through CSPNet. Prior works which differentiate the distance function on the zero level-set (i.e the surface) [32] also need a backward pass. Even so, since the derivative of *UDFs* vanish at the surface, NDF estimates the normals close to the surface [9] leading to some loss in fidelity.

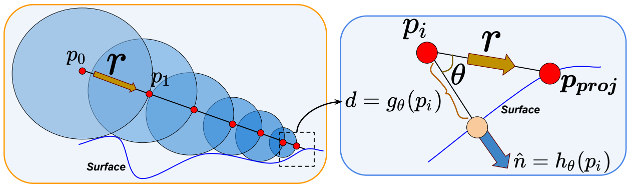


Figure 3: ***Left:*** An illustration of the Sphere Tracing procedure described in Section. 3.3.1. ***Right:*** Leveraging the *NVF* for obtaining more accurate ray-scene intersections.

3.2.2 Forward Mode Normal Estimation

In certain applications like rendering, sphere tracing is used to obtain a point on the surface and it is needed to quickly and efficiently estimate the normal at the point of intersection [12]. We can use the Jacobian approach presented in the previous section but it requires a backward pass.

An alternate approach for obtaining a fast approximation for the surface normal, using a forward pass from a query point $\boldsymbol{p}$ close to but not on the surface is by using the Normal Vector Field (*NVF*) defined as follows:

$$\mathit{NVF}(\boldsymbol{p}) = \frac{\boldsymbol{p} - \tilde{\boldsymbol{p}}}{\mathit{UDF}(\boldsymbol{p}) + \epsilon} \tag{4}$$

where ϵ is a small value to avoid division by 0 when $\boldsymbol{p}$ is on the surface. We represent the corresponding estimate for *NVF* by h_θ as $(\boldsymbol{p} - f_\theta(\boldsymbol{p}))/(g_\theta(\boldsymbol{p}) + \epsilon)$. We refer to this method of estimating normals as forward-mode normal estimation. Since there is no backward pass involved, it is faster than the previous methods. We demonstrate the utility of this approach in Sec. 4.2 and validate its performance both in terms of accuracy and speed via extensive experimental evaluation. More generally, fast estimation of the *NVF* at off-surface locations is vital to robotics applications such as path planning in distance fields [14, 30] and hand tracking [39].

3.3. Rendering and Meshing

In this section, we describe techniques for rendering surfaces and extracting topologically consistent meshes from the the learnt representation. Note that this process is important for many downstream vision applications such as shape analysis [24] and graphics applications such as rendering novel scenes under changed illumination, texture, or camera viewpoints [35].

3.3.1 Sphere Tracing *CSP*

Sphere tracing [20] is a standard technique to render images from a distance field that represents the shape. To create an image, rays are cast from the focal point of the camera, and their intersection with the scene is computed using sphere tracing. Roughly speaking, irradiance/radiance computations are performed at the point of intersection to obtain the color of the pixel for that ray.

The sphere tracing process can be described as follows: given a ray, $\boldsymbol{r}$, originating at point, $\boldsymbol{p_0}$, iterative marching along the ray is performed to obtain its intersection with the surface. In the first iteration, this translates to taking a step along the ray with a step size of *UDF*$(\boldsymbol{p_0})$ to obtain the next point $\boldsymbol{p_1} = \boldsymbol{p_0} + \boldsymbol{r} \cdot g_\theta(\boldsymbol{p_0})$. Since $g_\theta(\boldsymbol{p_0})$ is the smallest distance to the surface, the line segment $[\boldsymbol{p_0}, \boldsymbol{p_1}]$ of the ray is guaranteed not to intersect the surface ($\boldsymbol{p_1}$ can touch but not transcend the surface). The above step is iterated i times till $\boldsymbol{p_i}$ is ϵ close to the surface. The i-th iteration is given by $\boldsymbol{p_i} = \boldsymbol{p_{i-1}} + g_\theta(\boldsymbol{p_{i-1}})$ and the stopping criteria $g_\theta(\boldsymbol{p_i}) \leq \epsilon$.

Note that the above procedure can be used to get close to the surface but does not obtain a point on the surface. Once we are close enough to the surface, we can use a local planarity assumption (without loss of generality) to obtain the intersection estimate. This is illustrated in Figure 3 and is obtained in the following manner: if we stop the sphere tracing of the *CSP* at a point $\boldsymbol{p_i}$, we evaluate the *NVF* at that point as $\hat{\boldsymbol{n}} = h_\theta(\boldsymbol{p_i})$, and compute the cosine of the angle between the *NVF* and the ray direction. The estimate is then obtained as $\boldsymbol{p_{proj}} = \boldsymbol{p_i} + \boldsymbol{r} \cdot \frac{g_\theta(\boldsymbol{p_i})}{\boldsymbol{r}^T \hat{\boldsymbol{n}}}$.

3.3.2 From *CSP* to Meshes

Sphere tracing *CSP*, described in the previous section, can only be used to render a view of the shape. Thus, the extracted surface is immutable and cannot be used for applications such as 3D shape modeling, analysis and modification [24]. Explicit 3D mesh representations are more amenable for such applications. In this section, we propose an approach to extract a 3D mesh out of the learnt *CSP*.

A straightforward way to extract a mesh from an implicit representation is to create a high-resolution 3D distance grid and using the marching cubes algorithm [26] on this grid. However, as discussed in [29] this process is computationally expensive at high-resolutions, as we need to densely evaluate the grid. In [29] a method for multi-resolution surface extraction technique is proposed by hierarchically creating a binary occupancy grid by conducting inside/outside tests for a binary classifier based implicit representation.

However, *CSPs* cannot perform inside/outside tests. Hence we propose a novel technique to hierarchically divide the distance grid using edge lengths of the voxel grid cubes. We illustrate the procedure in Fig. 4. Starting with a voxel grid at some initial resolution, we obtain a high resolution distance grid and perform marching cubes on the grid using a small positive threshold to get the final mesh. A voxel is chosen for subdivision if any of its eight corners have the predicted *UDF* value $g_\theta(\boldsymbol{x}) < h_i$, where h_i is the edge length of the voxel grid at the $i'th$ level. The voxels that are not chosen for subdivision are simply discarded in the next level. Using this procedure, we quickly obtain a high-resolution distance grid, which is converted to a mesh using marching cubes. We find that our algorithm selects a few false positive voxels in the final resolution, but these are effectively pruned out in the final mesh by using a small positive threshold in the marching cubes [26] step.

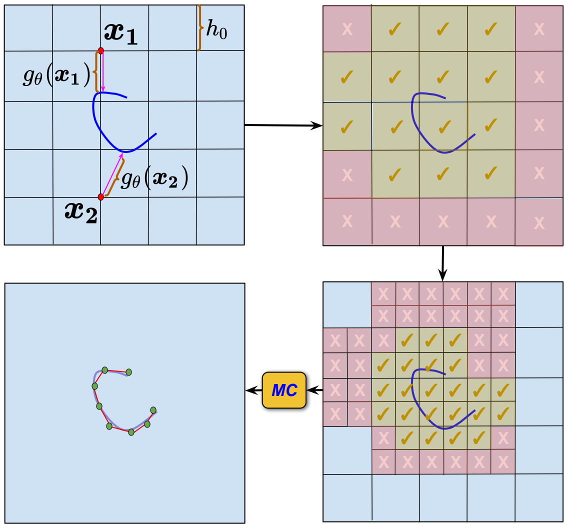


Figure 4: Multi-Resolution surface extraction for *CSP* (described in Section. 3.3.2). At each level of hierarchy, we show the voxels selected for subdivision. Note that there are some false positive voxels selected close to the shape which get eliminated in the next hierarchical level (Step 2 to Step 3), and are pruned out by using a small positive threshold while meshing the distance-grid in the Marching cubes (MC) step.

Note that marching cubes on an unsigned distance grid leads to the creation of a mesh with a double/crusted surface, given that there are two isosurfaces satisfying the small positive threshold. Such a representation is desirable for many applications [6, 10] as even the thinnest real world objects have surfaces with some infinitesimal thickness. On the other hand, extracting a mesh that represents a single (true) isosurface is desirable from a compactness standpoint, and is an interesting avenue for future work.

4. Experiments

In this section, we validate the different parts of our proposed system outlined in Fig. 2 against a selection of prior art. First we demonstrate the superiority of the proposed implicit shape representation (*CSP*: Sec. 3.1) on the task of surface reconstruction from point clouds. Next, we validate the proposed methods for extracting local surface properties such as surface normals (described in Sec. 3.2). Finally, we test the novel sphere-tracing algorithm for *CSPs* and the coarse to fine meshing algorithm (described in Sec. 3.3).

Baselines. Most existing methods such as Occupancy Networks [29], DeepSDF [32], Deep Marching Cubes [25] and IF-Net [8] only work on watertight (i.e. closed) shapes. We compare against these methods to verify that our method retains performance on closed shapes, in addition to being able to model open shapes. On the other hand, for comparing performance on raw/unprocessed shapes, we choose SAL [2] and NDF [9]. While these methods can work with non-watertight (i.e. unprocessed) ground truth, they still require a backward pass to estimate surface normals, which leads to an added computational and memory footprint. Additionally, while NDF can reconstruct both open as well as closed shapes, it is unable to guarantee plane reproduction and accurate normal estimates on the surface of the shape. In the following sections, we show empirically that our method addresses these challenges.

4.1. Shape Representation

In this section, we demonstrate the representational power of our model on ShapeNet dataset [5]. We consider the task of surface reconstruction from point clouds, and first evaluate on closed shapes to verify that our proposed generic shape representation yields comparable performance to state-of-the art methods which are solely meant to work on watertight shapes [32, 29, 15]. Second, we evaluate our method against NDF and SAL on raw, unprocessed shapes. Before describing the results, we present the evaluation metrics that we consider.

4.1.1 Evaluation metrics

A common practice for evaluating 3D reconstruction pipelines is the chamfer distance metric [32, 2, 9]. However, as discussed in some prior work [42], this metric does not reflect the perceptual quality of the rendered image. Moreover, for applications such as relighting [27] it is desirable to obtain surface normal maps by directly rendering the isosurface using sphere-tracing, as opposed to extracting a mesh. Clearly, there is a need to evaluate implicit shape representations on the perceptual quality of their isosurfaces rendered via sphere-tracing. Therefore, in addition to the chamfer distance we propose new metrics (outlined below) which are designed to capture these properties.

Depth Error (DE). First we evaluate the mean absolute error (*MAE*) between the ground truth and the estimated depth map obtained by sphere-tracing the learnt representation. This error is evaluated only on the "valid" pixels, which we define as the pixels having non-infinite depth (foreground) in both the ground truth and estimated depth map. This metric captures the *accuracy of ray-surface intersection*.

Normal Cosine Similarity (NCS). We also evaluate the cosine similarity between the sphere-traced normal map and the ground truth normal map for the valid pixels. Since the surface normals play a vital role in rendering, this metric is informative of the *fidelity* of the rendered surface.

Pixel-Space IOU. Finally, since both Depth Error and NCS are evaluated only on the valid pixels, they do not quantify whether the *geometry of the final shape* is correct. Therefore, we also evaluate Pixel-Space IOU,

$$IOU = \frac{\#\mathit{Valid\,Pixels}}{\#\mathit{Invalid\,Pixels} + \#\mathit{Valid\,Pixels}} \tag{5}$$

Here the invalid pixels are those which have non-infinite depth (foreground) in either the ground truth depth map or the estimated depth map but not both. Note that for the proposed metrics, we render the shape from 6 views (uniformly sampled on sphere) to capture all the regions of the surface.

4.1.2 Data creation

We first normalize each mesh in the ShapeNet dataset to $[-0.5, 0.5]$ using the steps followed in ONet [29]. For each shape, we densely sample a set of 0.25M points, denoted by the set $\mathcal{V}$, to represent the set of surface points. Similar to the startegy used in DeepSDF [32], training points $\mathcal{P}$ are obtained for each shape by uniformly sampling 0.025M points as well as perturbing the set $\mathcal{V}$ with a gaussian noise of $2.5\mathrm{e}{-4}$ and $2.5\mathrm{e}{-3}$. Finally, the ground truth for each trainin points $\boldsymbol{p} \in \mathcal{P}$ is computed by finding its nearest surface point $\tilde{\boldsymbol{p}} \in \mathcal{V}$ to construct the training pair $(\boldsymbol{p}, \tilde{\boldsymbol{p}})$.

4.1.3 Training

Note that, we only train f_θ, and g_θ and h_θ can be derived from it in the same way *UDF* and *NVF* can be derived from *CSP* in eq. 2 and eq. 4 respectively. Given $f_\theta(\boldsymbol{p}|X) = \phi(\psi(X), \boldsymbol{p})$, as the training objective, we simply use the squared L_2 loss between the estimated closest surface-point $f_\theta(\boldsymbol{p}|X)$ and the ground-truth $\tilde{\boldsymbol{p}}(= \mathit{CSP}(\boldsymbol{p}|X))$

$$\mathcal{L}_{\mathit{CSP}} = \frac{1}{|\mathcal{P}|} \sum_{\boldsymbol{p} \in \mathcal{P}} ||f_\theta(\boldsymbol{p}|X) - \tilde{\boldsymbol{p}}||_2^2 \tag{6}$$

4.1.4 Evaluation on closed shapes

We convert all the ShapeNet 3D models to closed shapes by following the steps in [37]. Following this, we run our data creation process (outlined in Sec. 4.1.2).

After training our proposed surface reconstruction pipeline, we compare to the selected prior art outlined earlier and report the results in Table. 2. We find that our class agnostic model performs on par with NDF.

Method	Chamfer-*L2* ($\times 10^4$)
PSGN [15]	4.0
ONet [29]	4.0
DMC [25]	1.0
CON [33]	0.95
IF-Net [8]	0.2
NDF [9]	**0.05**
CSP (Ours)	0.1

Table 2: Results: closed shapes.

4.1.5 Evaluation on unprocessed shapes

In addition to closed shapes, *CSP* can also represent shapes of arbitrary topology. Therefore, we also train on unprocessed ShapeNet 3D models and evaluate performance using the metrics defined in Sec. 4.1.1. We compare against SAL and NDF, which are methods that can learn representations from raw/unprocessed ground truth. [1] This comparison is reported in Table 3. We find that *CSP* marginally outperforms NDF on chamfer, depth and IOU metrics, but yields a significant improvement on the surface normals metric owing to the useful plane reproduction property of *CSP*. Additionally, SAL clearly suffers on all metrics, given that it learns a signed distance function (closed shape) even for surfaces that are open. This behavior can also be confirmed in the qualitative results shown in row 2 of Fig. 1.

Method	Chamfer-*L2* $\times 10^4 \downarrow$	Depth $\downarrow$	Normal $\uparrow$	IOU $\uparrow$
SAL [2]	2.25	0.025	0.84	0.96
NDF [9]	1.73	0.018	0.86	0.97
CSP (Ours)	**1.28**	**0.014**	**0.92**	**0.98**

Table 3: Results on unprocessed shapes. We evaluate both on chamfer distance metric as well as the three additional metrics defined in Sec. 4.1.1. For all methods, we obtained normals by leveraging first order information. We use the protocol followed in NDF for computing chamfer distance.

4.2. Local Surface Properties

In Sec. 3.2, we described various strategies to estimate surface normals using the learnt implicit representation. We refer to the strategy using the Jacobian as *CSP (jac.)* and the one using the forward pass (eqn. 4) as *CSP (fwd.)*.

Similar to NDF, this latter approach approximates surface normals using off-surface points close to the surface (where $\boldsymbol{p} \approx \tilde{\boldsymbol{p}}$) by stepping back along the ray at its point of intersection with the surface. More concretely, given a ray $\boldsymbol{r}$ which intersects with the surface at $\boldsymbol{p_{int}}$ (at the end of sphere-tracing), the normal is computed by stepping back along the ray by some scalar value α. Thus, $\hat{n}_{\boldsymbol{p_{int}}} = \nabla_{\boldsymbol{p_{int}}} g_\theta(\boldsymbol{p_{int}} - \alpha \cdot \boldsymbol{r})$. Note here that α is a hyperparameter which is sensitive to the curvature of the surface, and NDF chooses a constant $\alpha = 0.005$. However, we observe that choosing a single α for all shapes is sub-optimal given that surfaces can have varying curvatures.

To investigate the sensitivity of the system to varying α, we record normal cosine similarity vs different values of α in Table 4. It can be clearly seen that *CSP (jac.)* has higher quality normal estimates for points on the surface (i.e. $\alpha = 0$), given its tangent plane reproduction property, as opposed to NDF and *CSP (fwd.)* which do not. It is interesting to note that although SAL learns a signed distance function that is differentiable on the surface of the shape, it still performs poorly on this metric, owing to the instability of their unsigned similarity loss, and poor geometric reconstruction on open shapes. However, we find that both NDF and *CSP (fwd.)* yield comparable performance to *CSP (jac.)* if allowed to step back along the ray ($\alpha = 0.005$). However, the normal cosine similarity is lower than *CSP (jac.)* at $\alpha = 0$, which is a definite drawback. Moreover, we find that at $\alpha = 0.005$, *CSP (fwd.)* yields similar performance compared to NDF, even though it does not use a backward pass. We report rendering speeds and memory footprint for *CSP (fwd.)* and NDF in Table 5, and we immediately find that *CSP* is superior on both fronts.

Additionally, in Table 4 we find that although $\alpha = 0.005$ yields reasonably good normals, the standard deviation is higher than those obtained by the tangent plane approximation. This clearly shows that choosing a single threshold for all shapes [9] is sub-optimal. Finally, we qualitatively compare various normal estimation strategies in Fig. 6. We find here too that *CSP (fwd.)* performs reasonably well for $\alpha = 0.005$, with *CSP (jac.)* yielding the best performance at $\alpha = 0$. Both visually and quantitatively, we find that our normal estimation strategies outperform NDF. Additionally, forward-mode surface normal estimates, *Ours (fwd.)* are faster than that of NDF while *Ours (jac.)* is comparable in speed (more analysis in supplementary).

Method	Normal Map Similarity			
	$\alpha = 0$	$\alpha = 0.002$	$\alpha = 0.005$	$\alpha = 0.05$
SAL [2]	0.84 ± 0.017	0.851 ± 0.014	0.871 ± 0.009	0.861 ± 0.01
NDF [9]	0.863 ± 0.01	0.882 ± 0.008	0.903 ± 0.006	0.891 ± 0.008
CSP (fwd.)	0.620 ± 0.12	0.873 ± 0.018	0.912 ± 0.006	$\mathbf{0.91 \pm 0.007}$
CSP (jac.)	$\mathbf{0.913 \pm 0.003}$	$\mathbf{0.915 \pm 0.003}$	$\mathbf{0.920 \pm 0.003}$	0.871 ± 0.01

Table 4: Normal estimation accuracy of various methods described in Sec. 3.2. Here α refers to the step-back distance along the ray.

4.3. Rendering and Meshing

In this section, we validate our sphere-tracing strategy and meshing algorithm (Sec. 3.3) against various baselines.

Sphere Tracing *CSP*. We compare the sphere tracing strategy described in Sec. 3.3.1 to a baseline strategy when the projection step is excluded from the algorithm. Our proposed strategy yields better depth maps (*MAE* = 0.014) than the Vanilla Sphere tracing (*MAE* = 0.016) owing to more accurate ray-scene intersection. As expected, the qualitative results (depth error maps) shown in Fig. 5 also indicate the benefit of using projection step as a part of sphere tracing *CSP*. Refer supplementary material for more visualizations.

Speed & memory footprint of rendering. In Table 5, we report the average time taken to render a 512×512 im-

[1] Since both SAL and NDF do not provide a release of pretrained class-agnostic models, we retrain them. Further, as *CSP* uses a more powerful backbone (CON [33]) than the one originally proposed in SAL, we train SAL with the CON backbone, to ensure a fair comparison.

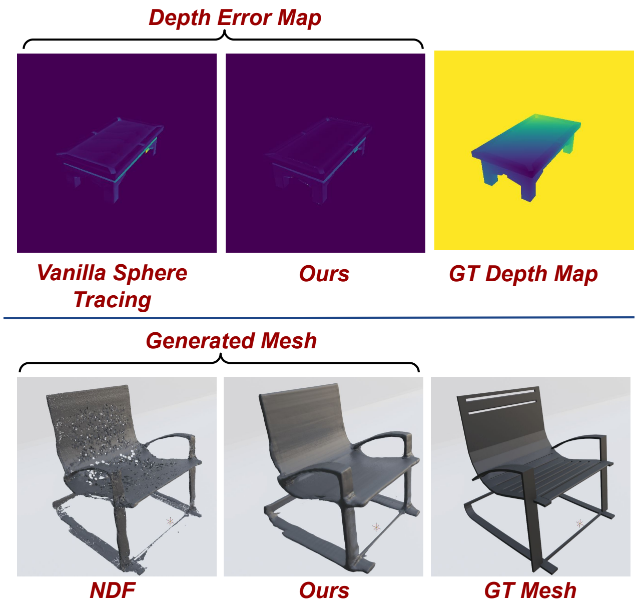


Figure 5: ***Top:*** Depth error maps comparing vanilla sphere tracing strategy vs our projection based strategy ***Bottom:*** Comparison of our meshing algorithm with that of NDF. Note that NDF displays visible artifacts, whereas our strategy reconstructs a topologically consistent mesh.

age using a memory budget of 8GiB. Since we do not rely on backward passes through the network (see definition of *NVF* in Sec. 3.2.2) higher batch sizes can be used on a fixed GPU budget, which leads to 20× faster rendering. In this manner, *CSP* provides a viable solution for applications which require real time, fast estimation of surface normals on small GPUs with limited memory [14, 14, 12]. Refer to supplementary material for more details on the specifics of the experimental setup.

Method	Rendering time	#decoder params	Memory Budget
NDF *CSP (fwd.)*	0.063s **0.003s**	1.97M 1.91M	8GiB

Table 5: Rendering times for a 512×512 image and memory footprint of the proposed method (*CSP*) against NDF.

Meshing *CSP*. Our novel coarse-to-fine meshing algorithm allows for fast conversion of *CSP* to mesh, providing a viable and fast alternative to that proposed in NDF [9]. For the representative example shown in Fig. 5, NDF's method takes 4.48s to generate the dense point cloud and an additional 104s for meshing using BPA [4][2]. On the other hand, our method takes a total of 2.50s for generating the final mesh. It is also important to note that although we use only 636k function evaluations, a higher quality mesh is recovered in comparison to NDF which uses 12M function evaluations. In Fig. 5 we show qualitative comparisons to NDF's meshing algorithm for *UDFs*. Note also that NDF's meshes display visible artifacts, given that it recovers the mesh after performing BPA on a dense point cloud (1M pts.) generated from the learnt representation. In contrast, our coarse-to-fine meshing strategy enables the application of Marching Cubes, and reconstructs a topologically consistent and visually pleasing mesh. Refer supplementary for details on the hyperparameters used.

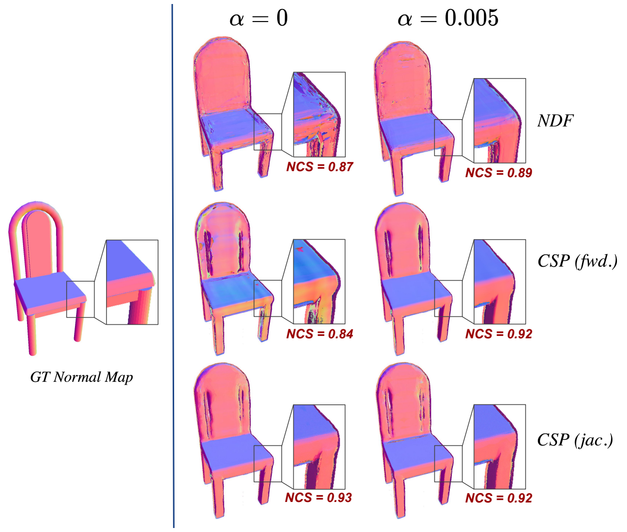


Figure 6: Comparison of normal estimation strategies outlined in Sec. 3.2. Note that *CSP (jac.)* at $\alpha = 0$ is the best performing model (0.93), with *CSP (fwd.)* at $\alpha = 0.005$ following next (0.92).

5. Conclusion

In this work, we proposed a new class of implicit representations called *CSP* that can model complex 3D objects (both open and closed surfaces), with a fidelity surpassing the state of the art. We demonstrated that *CSP* also facilitates accurate and efficient computation of local geometric properties of the surface like the tangent plane and the surface normal which enables efficient algorithms for downstream applications like surface rendering and meshing - we presented novel algorithms for both. We further showed that *CSP* yields state-of-the-art performance on the unprocessed ShapeNet dataset, surpassing prior art such as SAL [2] and NDF [9]. In summary, this work provides a strong alternative to existing methods for 3D modeling and representation by addressing fundamental problems in representing complex shapes. In the future, we expect to extend this work to infer surface representations - both geometric and photometric - from single and multi-view 2D images.

Acknowledgements This research was supported in part by NSF award No. IIS-1925281 and Sudha Murthy Chair Project, Pratiksha Trust, IISc.

[2] BPA is known to be slow [4]

References

[1] Samir Akkouche and Eric Galin. Adaptive implicit surface polygonization using marching triangles. In *Computer Graphics Forum*, volume 20, pages 67–80. Wiley Online Library, 2001. 2

[2] Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2, 3, 6, 7, 8

[3] Matan Atzmon and Yaron Lipman. {SALD}: Sign agnostic learning with derivatives. In *International Conference on Learning Representations*, 2021. 2

[4] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999. 8

[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2, 6

[6] Hsiao-Yu Chen, Arnav Sastry, Wim M van Rees, and Etienne Vouga. Physical simulation of environmentally induced thin shell deformation. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018. 5

[7] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. BSP-Net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 45–54, 2020. 2, 3

[8] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020. 6

[9] Julian Chibane, Mohamad Aymen mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21638–21652. Curran Associates, Inc., 2020. 1, 2, 3, 4, 6, 7, 8

[10] Min Gyu Choi, Seung Yong Woo, and Hyeong-Seok Ko. Real-time simulation of thin shells. In *Computer Graphics Forum*, volume 26, pages 349–354. Wiley Online Library, 2007. 5

[11] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3D object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 2

[12] Wagner T Corrêa, James T Klosowski, and Cláudio T Silva. Out-of-core sort-first parallel rendering for cluster-based tiled displays. *Parallel Computing*, 29(3):325–338, 2003. 2, 4, 8

[13] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey E. Hinton, and Andrea Tagliasacchi. CvxNet: Learnable convex decomposition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 31–41. IEEE, 2020. 2, 3

[14] Thibaud Duhautbout, Julien Moras, and Julien Marzat. Distributed 3d tsdf manifold mapping for multi-robot systems. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–8. IEEE, 2019. 2, 4, 8

[15] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 6

[16] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2

[17] Eleonora Grilli, Fabio Menna, and Fabio Remondino. A review of point clouds segmentation and classification algorithms. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:339, 2017. 2

[18] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[19] Xianfeng Han, Hamid Laga, and Mohammed Bennamoun. Image-based 3D object reconstruction: State-of-the-art and trends in the deep learning era. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2019. 1

[20] John C Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996. 4

[21] Jingwei Huang, Hao Su, and Leonidas Guibas. Robust watertight manifold surface generation method for ShapeNet models. *arXiv preprint arXiv:1802.01698*, 2018. 2, 3

[22] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 2, 4

[23] Jogendra Nath Kundu, Mugalodi Rakesh, Varun Jampani, Rahul M Venkatesh, and R. Venkatesh Babu. Appearance consensus driven self-supervised human mesh recovery. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[24] Hamid Laga, Yulan Guo, Hedi Tabia, Robert B Fisher, and Mohammed Bennamoun. *3D Shape analysis: fundamentals, theory, and applications*. John Wiley & Sons, 2018. 4, 5

[25] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018. 6

[26] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In Maureen C. Stone, editor, *SIGGRAPH*, pages 163–169. ACM, 1987. 5

[27] Robert Maier, Kihwan Kim, Daniel Cremers, Jan Kautz, and Matthias Nießner. Intrinsic3d: High-quality 3d reconstruction by joint appearance and geometry optimization with

spatially-varying lighting. In *Proceedings of the IEEE international conference on computer vision*, pages 3114–3122, 2017. 6

[28] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015. 2

[29] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 1, 2, 3, 5, 6

[30] Helen Oleynikova, Alexander Millane, Zachary Taylor, Enric Galceran, Juan Nieto, and Roland Siegwart. Signed distance fields: A natural representation for both mapping and planning. In *RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*. University of Michigan, 2016. 2, 4

[31] Junyi Pan, Xiaoguang Han, Weikai Chen, Jiapeng Tang, and Kui Jia. Deep mesh reconstruction from single RGB images via topology modification networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9964–9973, 2019. 1

[32] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1, 2, 3, 4, 6

[33] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 523–540, Cham, 2020. Springer International Publishing. 3, 6, 7

[34] François Pomerleau, Francis Colas, and Roland Siegwart. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1):1–104, 2015. 2, 4

[35] Timothy J Purcell, Ian Buck, William R Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. In *ACM SIGGRAPH 2005 Courses*, pages 268–es. 2005. 4

[36] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 3

[37] David Stutz and Andreas Geiger. Learning 3D shape completion from laser scan data with weak supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1955–1964, 2018. 6

[38] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3D: Dataset and methods for single-image 3D shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[39] A. Tagliasacchi, M. Schröder, A. Tkach, Sofien Bouaziz, M. Botsch, and M. Pauly. Robust articulated-icp for real-time hand tracking. *Computer Graphics Forum*, 34, 2015. 4

[40] Pascal Volino and Nadia Magnenat-Thalmann. Fast geometrical wrinkles on animated surfaces. In *Seventh International Conference in Central Europe on Computer Graphics and Visualization (Winter School on Computer Graphics)*, 1999. 4

[41] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *ECCV*, 2018. 2

[42] Chulin Xie, Chuxin Wang, Bo Zhang, Hao Yang, Dong Chen, and Fang Wen. Style-based point generator with adversarial rendering for point cloud completion. *arXiv preprint arXiv:2103.02535*, 2021. 6