NeeDrop: Self-supervised Shape Representation from Sparse Point Clouds using Needle Dropping

Alexandre Boulch[1] Pierre-Alain Langlois[2] Gilles Puy[1] Renaud Marlet[1,2]

[1]Valeo.ai, Paris, France [2]LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France

Abstract

There has been recently a growing interest for implicit shape representations. Contrary to explicit representations, they have no resolution limitations and they easily deal with a wide variety of surface topologies. To learn these implicit representations, current approaches rely on a certain level of shape supervision (e.g., inside/outside information or distance-to-shape knowledge), or at least require a dense point cloud (to approximate well enough the distance-to-shape). In contrast, we introduce NeeDrop, an self-supervised method for learning shape representations from possibly extremely sparse point clouds. Like in Buffon's needle problem, we "drop" (sample) needles on the point cloud and consider that, statistically, close to the surface, the needle end points lie on opposite sides of the surface. No shape knowledge is required and the point cloud can be highly sparse, e.g., as lidar point clouds acquired by vehicles. Previous self-supervised shape representation approaches fail to produce good-quality results on this kind of data. We obtain quantitative results on par with existing supervised approaches on shape reconstruction datasets and show promising qualitative results on hard autonomous driving datasets such as KITTI.

1. Introduction

Learning-based approaches for 3D reconstruction from sparse 3D data have recently attracted of lot of interest. As opposed to classical approaches [3], such as Poisson reconstruction [37], these methods enable prior knowledge to be used to enrich the representation of information-deficient inputs, e.g., low density point clouds or partial scene views.

Most of the recent learning-based methods for shape reconstruction from point clouds fall into two categories. A first category produces an *explicit* or *parametric* representation of the shape: point cloud, voxel or mesh. For instance, some may deform a geometric primitive or a template mesh [30, 31], e.g., a planar patch or a sphere. The topology of both the template and the reconstruction are thus identical, which is a significant limitation. The second category of methods operates on an *implicit* formulation of the shapes. These methods build a continuous function over the 3D space, either based on occupancies [46], or on signed [48] or unsigned [13] distance functions. They are not bound by the topology of a template but require an extra-processing step for mesh extraction. Beyond shape reconstruction, several approaches also tackle the problem of shape generation by encoding shapes in a low-dimensional latent space with a constraint on the distribution of the latent variables. All existing learning-based approaches use a certain level of supervision during training, either using the information of distance to the shape or using the knowledge of which points fall inside or outside the shape.

In this work, we propose what we believe is the first *self-supervised* approach for shape reconstruction from *sparse* point cloud. It is self-supervised in the sense that it does not need to learn from actual shapes, or even from densely sampled point clouds; sparse, noisy, and even partial point clouds are enough. And it reconstructs a shape in the sense that an actual mesh can directly be produced from the occupancy field we predict, e.g., using a Marching Cubes algorithm [44], as opposed to approaches that only produce renderings or generate points on the underlying surface [8].

Like some other methods, we build a shape representation in a latent space and predict an implicit occupancy field. An actual surface can then be easily extracted as the zero-level set of the function. However, unlike most existing methods, we only use point clouds as input for learning, instead of meshes. We thus do not have to worry about shape watertightness, which is a major concern for occupancy-based methods training on meshes because most shape datasets collect good-looking but actually ill-formed meshes, thus requiring a significant preprocessing. Furthermore, we show that our method can deal with highly sparse input point clouds without surface supervision, such as point clouds captured by Lidars on moving vehicles, whereas all existing methods that use point clouds as input (except [8]) assume that points are dense enough so that the distance to the shape (supervision) can be closely ap-

proximated by the distance to the point cloud, which leads to failures to learn from sparse inputs.

Our method is inspired by Buffon's needle problem [41] where one wants to compute the probability that a needle dropped on a wooden floor with parallel strips lies across two strips. Similarly, we drop (sample) needles on the point cloud such that needles built on input points have a high probability of crossing the surface, while needles constructed a bit away from input points have a low probability. Our main contributions are:

- a new loss for self-supervised shape reconstruction formulated via needle dropping on the point cloud;
- the use of this loss to learn only from point clouds how to predict shape representations;
- a overall method that can intrinsically deal with highly sparse and partial point cloud, at train and test time;
- the validation of our method on both synthetic data and real point clouds for which no supervision is possible.

The paper is organized as follows: section 2 presents related work; section 3 describes the loss function and the network used for shape reconstruction; section 4 presents experiments showing the performance of our method and its comparison with competing techniques.

2. Related work

Representing shapes, with applications such as reconstruction and generation, has been widely studied. This section only presents *learning*-based methods; classical (non-learning-based) methods are surveyed in [3]. We classify related work based on the type of shape representation used and on the level of supervision used at training time.

Point clouds are a common way to represent a shape. They can be obtained, e.g., directly from depth sensors, or via photogrammetry. Pioneered by PointNet-based architectures [51, 53], learning-based methods have reached the state of the art in multiple tasks such as classification and semantic segmentation [6,42,61]. Point clouds have also been successful for shape generation from images [23, 33, 50] or from a prior distribution [68]. Yet, a point cloud remains a sparse representation of the underlying surface and, when generating points, their number is often a fixed parameter.

Voxels allow to directly adapt techniques developed on 2D pixels to 3D data. They are used in many tasks ranging from classification and semantic segmentation [45, 52, 66] to shape generation [60, 64], representation [34, 36, 55] or completion [18, 59]. Represented as an occupancy grid, the reconstructed surface however suffers from quantization effects, which can be mitigated using truncated signed distance functions [16, 18, 40, 57, 59].

Besides, using voxels may rapidly lead to a high memory consumption as their number grows cubicly with the size of the scene. A greater accuracy, with finer voxel grids, may be obtained at the cost of a slow training process [15, 63, 66] or by integrating surface extraction for occupancy [43]. Sparse convolutions [14,27,28] or multi-resolution approaches [32, 64] such as octrees [56, 57, 60] can be used to reduce the memory footprint and scale to a complete scene [69].

Meshes describe a shape as a set of vertices and faces. They are the representation of choice in computer graphics and computer-aided design as they are very memory efficient and easily allow geometric operations and rendering with texture. Geometric deep learning [7] exploits their graph structure for classification and segmentation [20, 24].

A sub-category of methods operating on meshes deform one or several shape templates [31, 49, 54, 65]. While giving good results when the templates and the shapes are relatively similar, the resulting meshes necessarily have the same topology as the templates, which limits the complexity of the shapes that can be modeled. Besides, they may suffer from self-intersections caused by wide deformations. Template deformation can also be mixed with voxels, to refine a coarse reconstruction [26]. Other approaches directly predict sets of vertices and faces [17]. The result may not be continuous and can require additional mesh fixing steps.

Implicit representations model a closed shape via a continuous function of the 3D space. An actual surface is then extracted as a levelset of the function (or its gradient).

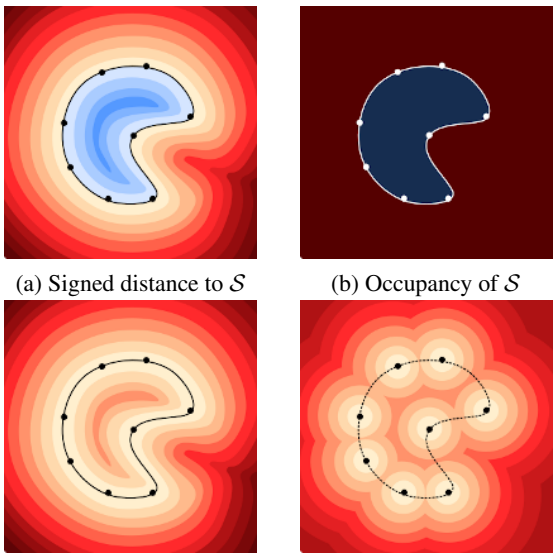
Most existing approaches associate to each point $\mathbf{x} \in \mathbb{R}^3$ a signed distance [1, 2, 11, 29, 35, 47, 48] or an occupancy value [12, 21, 25, 46]. Such implicit representations lead to the reconstruction of closed surfaces (with an inside and an outside) without template-topology limitations. In this work, the network is trained to output an occupancy value indicating on which side of the shape a query point is.

Unsigned distance fields [8, 13] have the advantage to be able to model open surfaces too. But they do not directly yield a mesh; they generate a dense point cloud, that is then given to a meshing algorithm to generate an actual surface.

Some hybrid methods predict a set of convex polytopes with one implicit function per polytope, to produce a piecewise representation of a shape [39, 62]. Others reconstruct a globale mesh by predicting voxel occupancy as well as a local mesh configuration [43].

Supervision level is key differentiating factor too. The full supervision of the real, signed distance to the shape (Fig. 1(a)) is typically tackled by a regression loss [43, 48]. Full supervision of the occupancy information (Fig. 1(b)) leads to networks predicting if a point is inside or outside the shape [12, 46]. A weaker supervision only provides the unsigned distance to the shape [13] (Fig. 1(c)). Finally, in the fully self-supervised setting, which is used in this work, only the distance to the input points is available, thus providing only an approximate distance to the shape (Fig. 1(d)).

Several methods fall in this last category. SAL [1] learns a sign-agnostic distance and a specific network initialization favors a sign change when crossing the surface, then allow-



(a) Signed distance to $\mathcal{S}$ (b) Occupancy of $\mathcal{S}$

(c) Unsigned distance to $\mathcal{S}$ (d) Distance to nearest point

Figure 1. Different kinds of supervision.

ing surface extraction. But sign change is not guaranteed. In contrast, needles enforce it at loss level. Besides, as underlined in [13], SAL tackles single objects reconstruction and often fails in multi-object reconstruction settings.

SALD [2] improves SAL by adding a term favoring supervised gradients on the surface to enforce sign change. However, it introduces an extra hyperparameter to balance reconstruction and regularization, which can be hard to tune because the balance depends on point density, whereas our two loss terms are homogeneous and simply summed.

IGR [29], on the contrary, only supervises (a null) distance on the surface but favors a norm-1 gradient everywhere. Here again, a hyperparameter balances reconstruction and regularization, with the same caveat as SALD.

Point cloud sparsity. These methods use very dense point clouds: 500k points for SALD [2], 16k for SAL [1] and 8k for IGR [29]. Distance to the nearest point (Fig 1(d)) then becomes a very good approximation of the unsigned distance (Fig. 1(c)). On the contrary, our method can cope with very sparse point clouds (300 points in our experiments), much more like in Fig. 1(d). Yet, the sparse point cloud setting is investigated in SAL [1] for single shape reconstruction. However, we observed in our experiments (Tab. 1) that SAL [1] and IGR [29] fail to produce meshes when trained on a collection of shapes with a very small number of input points. (No code is available for SALD.)

Only ShapeGF [8] handles very sparse point clouds, but although the paper talks of "surface extraction", it does not easily and directly produce a mesh. It actually predicts a shape gradient field that can be used to sample points on the underlying surface using stochastic gradient ascent (and a number of extra parameters), with a density whose uniformity cannot be controlled though. A separate meshing step of those noisy points is then required. (Ray-casting, with extra parameters too, can also produce nice renderings directly from the gradient field, as it also yields normals, which soften noise in rendered pixels.) In contrast, we generate an occupancy field, from which a mesh can be easily extracted [46]. The authors, in their supplementary material [8], mention failure cases coming from local minima and saddle points (where gradients are close to zero), as well as double surfaces arising when meshing, and they leave the improvement of surface extraction to future work.

Reconstruction with partial point clouds. Most methods focus on complete shapes, for which a full supervision is possible, or at least can be approximated. Some methods can deal with partial data as input, but still rely on full shape supervision when training, as DeepSDF [48]. It is also the case of single-view reconstruction methods, that take an RGB image as input and reconstruct the whole shape [30,46]. In [58], a network is trained from volumetric completion from RGBD data with supervision on the volume to be reconstructed. In contrast, our method does not require any level of supervision beyond sparse points and can use partial point clouds when training and testing.

Needles. OccNet [46] traces rays of possibly infinite length to close shapes, while our needles are bounded. DeepSDF [48] has a kind of needle to sign a distance ($\pm\eta$) w.r.t. a viewpoint, whereas our needles have uniform orientations and apply to self-supervised surface reconstruction.

3. Self-supervised reconstruction

3.1. Implicit shape representation

Like previous work aiming at predicting an implicit shape representation, our goal is to approximate the ideal function $f^0_{\mathcal{S}}$ such that some $\alpha \in \mathbb{R}$ level-set of $f^0_{\mathcal{S}}$ is the surface S of the target shape $\mathcal{S}$:

$$S = \{\mathbf{x} \in \mathbb{R}^3 | f^0_{\mathcal{S}}(x) = \alpha\} \tag{1}$$

For signed distance estimation [1, 48], $\alpha = 0$: negative values are inside the shape (by convention), and positive values are outside.

We consider here $f^0_{\mathcal{S}}$ to be an occupancy function,

$$f^0_{\mathcal{S}} : \mathbb{R}^3 \to \{0, 1\} \tag{2}$$

such that $f^0_{\mathcal{S}}$ takes value 0 outside the shape and 1 inside the shape. The surface of the shape is defined as the location of discontinuity between values 0 and 1.

Given an input point cloud $P_{\mathcal{S}}$ sampled on the shape, the objective is to estimate a function

$$f_{\mathcal{S}} : \mathbb{R}^3 \to [0, 1] \tag{3}$$

such that $f_{\mathcal{S}} \approx f^0_{\mathcal{S}}$. We consider that $f_{\mathcal{S}}$ belongs to the family of function that can be predicted using a neural network. Thus, $f_{\mathcal{S}}$ is continuous and we consider that the surface is the 0.5-level set of the function.

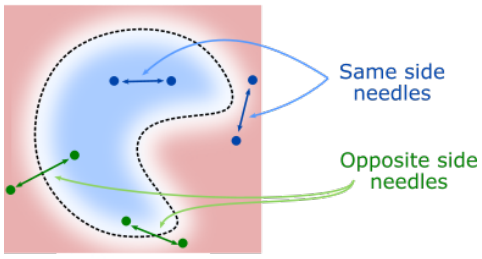


Figure 2. Needles as pairs of points.

3.2. Self-supervised learning from pairs of points

In other approaches, $f_{\mathcal{S}}^0$ is known [12, 46, 48], or partially known [1, 13] (known distance with great precision, unknown sign) at each point $\mathbf{x} \in \mathbb{R}^3$. The neural network can thus be trained directly to approximate $f_{\mathcal{S}}^0$ under supervision. On the contrary, in our self-supervised setting, $f_{\mathcal{S}}^0$ is not directly accessible and $P_{\mathcal{S}}$ is the only available information about $\mathcal{S}$. To reconstruct $f_{\mathcal{S}}^0$ given only $P_{\mathcal{S}}$, we take inspiration from Buffon's needle problem.

Buffon's needle problem. In [41], Buffon estimates the probability that the end points of a needle, dropped on a floor made of parallel strips of wood, lie on different strips.

Needle droping for surface reconstruction. Inspired by this probability problem, we base our reconstruction strategy using needles (pairs of points), instead of points as in previous works. Let $\mathbf{x}$ and $\mathbf{y}$ be the end points of a needle "dropped" in 3D space. We will construct our self-supervised training loss based on the facts that if $\mathbf{x}$ and $\mathbf{y}$ lie on the same side of $\mathcal{S}$, then $f_{\mathcal{S}}(\mathbf{x})$ and $f_{\mathcal{S}}(\mathbf{y})$ shall be equal, while if $\mathbf{x}$ and $\mathbf{y}$ lie on opposite sides of $\mathcal{S}$, then $f_{\mathcal{S}}(\mathbf{x})$ and $f_{\mathcal{S}}(\mathbf{y})$ shall be different. It is illustrated on Figure 2.

3.2.1 Needle-based formulation

Our goal is to estimate the ideal, but unknown, occupancy function $f_{\mathcal{S}}^0$. As this function takes values in $\{0, 1\}$, we can define a Bernoulli distribution $\mathcal{B}_{\mathbf{x}}^0$ at every position $\mathbf{x} \in \mathbb{R}^3$. Similarly, we define the Bernoulli distribution $\mathcal{B}_{\mathbf{x}}$ with parameter $b_{\mathbf{x}} = f_{\mathcal{S}}(\mathbf{x})$ at the same location $\mathbf{x} \in \mathbb{R}^3$.

To reconstruct the surface, our goal is now to minimise the distance between $\mathcal{B}_{\mathbf{x}}$ and $\mathcal{B}_{\mathbf{x}}^0$ at every location $\mathbf{x} \in \mathbb{R}^3$. But this problem is ll-posed because $\mathcal{B}_{\mathbf{x}}^0$ is unknown in our case. To overcome this issue, we propose to construct new random variables out of $\mathcal{B}_{\mathbf{x}}$ and $\mathcal{B}_{\mathbf{x}}^0$ for multiple pairs of different locations $\mathbf{x}$ and $\mathbf{y}$. Let X and Y be two independent Bernoulli variables at location $\mathbf{x}$ and $\mathbf{y}$ drawn according to $\mathcal{B}_{\mathbf{x}}$ and $\mathcal{B}_{\mathbf{y}}$. The probability that $X = Y$ satisfies

$$b_{\mathbf{x},\mathbf{y}} = b_{\mathbf{x}}\, b_{\mathbf{y}} + (1 - b_{\mathbf{x}})(1 - b_{\mathbf{y}}). \tag{4}$$

We can thus define a new Bernoulli distribution $\mathcal{B}_{\mathbf{x},\mathbf{y}}$ of parameter $b_{\mathbf{x},\mathbf{y}}$ for needle $(\mathbf{x}, \mathbf{y})$. We similarly define the target Bernoulli distribution $\mathcal{B}_{\mathbf{x},\mathbf{y}}^0$ of parameter $b_{\mathbf{x},\mathbf{y}}^0$ using $\mathcal{B}_{\mathbf{x}}^0$.

Then to estimate $b_{\mathbf{x}}^0$, i.e., reconstruct the surface, we propose to minimise the binary cross-entropy BCE, i.e., the log-loss, between $b_{\mathbf{x},\mathbf{y}}$ and $b_{\mathbf{x},\mathbf{y}}^0$

$$\mathcal{L}_{\text{recons}}(\mathbf{x}, \mathbf{y}) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in Q} \text{BCE}(b_{\mathbf{x}_i,\mathbf{y}_i}, b_{\mathbf{x}_i,\mathbf{y}_i}^0), \tag{5}$$

for a finite set of well-chosen needles

$$Q = \{(\mathbf{x}_i, \mathbf{y}_i) \,|\, i \in \{1, \dots, |Q|\}\}. \tag{6}$$

In practice, $\mathcal{B}_{\mathbf{x},\mathbf{y}}^0$ is unknown but it is enough to know if the needles $(\mathbf{x}_i, \mathbf{y}_i)$ traverse the surface or not to be able to compute $\mathcal{L}_{\text{recons}}$ in Eq. (5). Inferring if a needle crosses the surface or not, when the needle is well chosen, is much easier than guessing if a point $\mathbf{x}$ is inside or outside the object.

3.2.2 Needle picking

We now describe our strategy to pick needles with end points on opposite sides or on the same side of the surface.

Surface-crossing needles (opposite-side end points). As the only information we have about the shape is $P_{\mathcal{S}}$, and considering that the surface is continuous and locally planar in the neighborhood of $\mathbf{p} \in P_{\mathcal{S}}$, which is true at arbitrarily small scales, we drop likely surface-crossing needles as line segments centered at points of $P_{\mathcal{S}}$, thus with likely opposite-side end points. The corresponding set Q_{opp} is

$$Q_{\text{opp}} = \{(\mathbf{p} + \mathbf{h}, \mathbf{p} - \mathbf{h}) \mid \mathbf{p} \in P_{\mathcal{S}}, \mathbf{h} \sim \mathcal{N}(0, \sigma_h)\}, \tag{7}$$

where $\mathbf{h}$ is randomly sampled from the multivariate normal distribution $\mathcal{N}(0, \sigma_h) \in \mathbb{R}^3$ with standard deviation σ_h, possibly depending on $\mathbf{p}$. The high probability of crossing the surface actually depends on curvature (see supp. mat.). Empirically, exceptions are rare enough not to confuse learning, as with datasets containing some erroneous labels.

Non-surface-crossing needles (same-side end points). We also have to sample needles not crossing the surface. We may note that if a 3D point $\mathbf{p}$ is not too close to $P_{\mathcal{S}}$ and if $P_{\mathcal{S}}$ is dense enough (which is the case in practice for ordinary objects sampled with as few as 300 points), then it is likely that the line segment between $\mathbf{p}$ and its closest point $\mathbf{p}'$ in $P_{\mathcal{S}}$ does not cross the surface $\mathcal{S}$, unless possibly near to $\mathbf{p}'$, including due to noise in input point cloud sampling. To simply take that into account, we consider closest points to P_{opp}, where P_{opp} is the set of end points in Q_{opp}, i.e., points of $P_{\mathcal{S}}$ with a slight offset in two opposite directions. To create short-enough needles that are unlikely to cross the surface and add variability in needle orientation w.r.t. the surface, we actually consider the following set of needles:

$$Q_{\text{same}} = \{(\mathbf{p}, \mathbf{p}') \mid \mathbf{p} \in P_{\text{same}}, \mathbf{p}' = \text{nn}(\mathbf{p}, P_{\text{opp}} \cup P_{\text{same}})\}. \tag{8}$$

where P_{same} are points sampled in 3D space and $\text{nn}(\mathbf{p}, P)$ is the nearest neighbor of $\mathbf{p}$ in P (excluding $\mathbf{p}$ itself).

Reconstruction loss. It would be natural to define the loss on $Q = Q_{\text{opp}} \cup Q_{\text{same}}$ in Eq. (5). However, to account for possibly different sizes of Q_{opp} and Q_{same}, we apply Eq. (5) independently on both sets. The loss is thus composed of two terms: a "data" term $\mathcal{L}_{\text{opp}}$ defined on Q_{opp} that enforces the surface to be located near $P_{\mathcal{S}}$ and a "regularization" term $\mathcal{L}_{\text{same}}$ ensuring side label consistency inside/outside the shape. The reconstruction loss satisfies

$$\mathcal{L}_{\text{recons}} = \mathcal{L}_{\text{opp}} + \mathcal{L}_{\text{same}} \tag{9}$$

where

$$\mathcal{L}_{\text{opp}} = \frac{1}{|Q_{\text{opp}}|} \sum_{(\mathbf{x},\mathbf{y}) \in Q_{\text{opp}}} \text{BCE}(b_{\mathbf{x},\mathbf{y}}, b^0_{\mathbf{x},\mathbf{y}}), \tag{10}$$

$$\mathcal{L}_{\text{same}} = \frac{1}{|Q_{\text{same}}|} \sum_{(\mathbf{x},\mathbf{y}) \in Q_{\text{same}}} \text{BCE}(b_{\mathbf{x},\mathbf{y}}, b^0_{\mathbf{x},\mathbf{y}}). \tag{11}$$

Our loss, as [1, 2, 8, 29], does not ensure an occupied shape interior and void remaining space; the reverse is a solution too, yet yielding the same surface. Due to non determinism, retraining on the same data may lead to either solution. Still, emptyness can be imposed at bounding box boundary.

3.3. Learning shape representations

From a collection of shapes provided only as point clouds, we learn a neural network Φ. Given a new point cloud $P_{\mathcal{S}}$ as input, Φ predicts an occupancy function $f_{\mathcal{S}}$:

$$f_{\mathcal{S}}(\mathbf{x}) = S \circ \Phi(P_{\mathcal{S}}, \mathbf{x}) \tag{12}$$

where S is the sigmoid function to ensure $f_{\mathcal{S}}(\mathbf{x}) \in [0, 1]$.

Training Loss. Substituting Eq. (12) in the expression of the log-loss in Eq. (5) yields:

$$\begin{aligned} \text{BCE}(b_{\mathbf{x},\mathbf{y}}, b^0_{\mathbf{x},\mathbf{y}}) = & \log(e^{\Phi(P_{\mathcal{S}},\mathbf{x})} + 1) + \log(e^{\Phi(P_{\mathcal{S}},\mathbf{y})} + 1) \\ & - b^0_{\mathbf{x},\mathbf{y}} \log(e^{\Phi(P_{\mathcal{S}},\mathbf{x}) + \Phi(P_{\mathcal{S}},\mathbf{y})} + 1) \\ & - (1 - b^0_{\mathbf{x},\mathbf{y}}) \log(e^{\Phi(P_{\mathcal{S}},\mathbf{x})} + e^{\Phi(P_{\mathcal{S}},\mathbf{y})}) \end{aligned} \tag{13}$$

which can be further simplified by exploiting the fact that $b^0_{\mathbf{x},\mathbf{y}} = 1$ when $(\mathbf{x}, \mathbf{y}) \in Q_{\text{same}}$, and $b^0_{\mathbf{x},\mathbf{y}} = 0$ when $(\mathbf{x}, \mathbf{y}) \in Q_{\text{opp}}$. In addition, the gradient expression (see supp. mat.) is simple, benefiting from simplifications similarly to the usual BCE loss for singletons.

4. Experiments

We conduct experiments on different datasets to validate the loss we propose and its use for surface reconstruction. We first describe the network used in our experiments and the experimental setup. Then, we evaluate our ability to reconstruct shapes from point clouds sampled on the entire surface (section 4.1). Finally, we show that our method can be applied to real point clouds from automotive datasets, either by direct transfer of previously learned models or by training from scratch, justifying our ability to work on highly sparse point clouds (section 4.2).

Network. We use the encoder/decoder network from [46]. The encoder is a PointNet [51] with 5 residual blocks. The decoder is a fully connected network conditioned via batch normalization on the latent code [19, 22].

Parameters. Our method works both with sparse and dense point clouds; yet it is in sparse regime that it makes a difference. For all experiments, unless otherwise stated, we use a point cloud size $|P_{\mathcal{S}}| = |Q_{\text{opp}}| = 300$ and we set $|Q_{\text{same}}| = 2048$. The parameter σ_h used to draw the vectors $\mathbf{h}$ in the construction of Q_{opp} is critical. A large σ_h distributes well the end points of both sides of the surface, increasing the chances that no needle in Q_{same} actually crosses the surface. However, too large a σ_h may lead to needles crossing the surface twice, which we assume does not occurs in our construction of the training loss. On the contrary, too small a σ_h reduces the influence of $\mathcal{L}_{\text{opp}}$ to a small neighborhood around the points in $P_{\mathcal{S}}$, preventing a good surface coverage and also increasing the chance of having needles in Q_{same} that accidentally cross the surface. As a rule of thumb, we set $\sigma_h(\mathbf{p}) = d_{\mathbf{p}}/3$, where $d_{\mathbf{p}}$ is the distance between a point $\mathbf{p}$ to its nearest neighbor in $P_{\mathcal{S}}$.

Training procedure. We train our model in an end-to-end fashion, with Adam [38] and a learning rate of $5 * 10^{-4}$. During training, the points are either randomly sampled on the shape if an input mesh is available or picked in the original point cloud, e.g., for lidar scenes from KITTI. At test time, we predict the occupancy values on a grid and use a Marching Cubes algorithm [44] for shape extraction.

Metrics and dataset pre-processing. For each experiment, and in order to compare with previous works, we use the same dataset, same pre-processing and same metrics as used in the compared papers, as mentioned in each table caption (IoU, ℓ_1 and ℓ_2 Chamfer distance, worst Chamfer distance of the best $x\%$ of points).

4.1. Synthetic point clouds

To compare with state-of-the art methods, we evaluate our self-supervised formulation, named NeeDrop, on point clouds sampled uniformly on closed shapes. Tab. 1 and Fig. 3 show our results on ShapeNet [10] and DFaust [5].

Evaluation on ShapeNet. We train our model on two configurations of ShapeNet. We first present results on the car subset (Tab. 1(a)) as in [12, 13], where the raw meshes are closed using the pre-processing of [67].

In Tab. 1(b), we extend the experiment to all ShapeNet categories of [46]; qualitative examples are presented in Fig. 3(a). For each compared method, we highlight its level of supervision, on distance and/or sign/occupancy. Unlike NeeDrop, all other methods (except ShapeGF) have a certain level of supervision. Despite the absence of supervision

		Supervision		Chamfer $\ell_2 \downarrow$	
		Dist.	Sign	Mean	Median
Input: 300 points		-	-	6.649	6.441
PSGN [23]	Points	✓	✗	1.986	1.649
DMC [43]	Explicit	✓	✓	2.417	0.973
OccNet [46]	Implicit	✗	✓	1.009	0.590
IF-Net [12]	Implicit	✗	✓	1.147	0.482
NDF [13]	Implicit	✓	✗	0.626	0.371
NeeDrop	Implicit	✗	✗	1.703	1.109
NeeDrop*	Implicit	✗	✗	1.575	0.952
NeeDrop-FP	Implicit	✗	✗	2.461	1.897

(a) ShapeNet cars, closed meshes [67], results $\times 10^{-4}$, cf. [13].

		Supervision		Eval. metrics	
Input: 300 pts+noise		Dist.	Sign	IoU $\uparrow$	Cha. $\ell_1 \downarrow$
3D-R2N2 [15]	Voxels	✗	✓	0.565	0.169
PSGN [23]	Points	✓	✗	-	0.144
ShapeGF§ [8]	Points	✗	✗	-	0.083
DMC [43]	Explicit	✓	✓	0.674	0.117
OccNet [46]	Implicit	✗	✓	0.778	0.079
NeeDrop	Implicit	✗	✗	0.666	0.112
NeeDrop*	Implicit	✗	✗	0.675	0.111
NeeDrop-FP	Implicit	✗	✗	0.669	0.106

(b) ShapeNet subset of [15], all classes, results $\times 10^{-1}$, cf. [46].

	No. points		Superv.	Chamfer $\ell_2 \downarrow$		
Method	Train	Test	Dist.	5%	50%	95%
DeepSDF† [48]	16k	16k	✓	3.45	45.03	294.15
DeepSDF‡ [48]	16k	16k	✓	1.88	31.05	489.35
AtlasNet [31]	16k	16k	✓	0.10	0.17	0.37
SAL [1]	16k	16k	✓	0.07	0.12	0.35
IGR [29]	8k	16k	✓	4.79	12.04	104.08
SAL§ [1]	300	300	✓	0.086	0.134	0.421
SAL§ [1]	300	-	✗	*Failed to converge*		
IGR§ [29]	300	-	✗	*Failed to converge*		
NeeDrop	300	300	✗	0.269	0.433	1.149
NeeDrop*	300	300	✗	0.107	0.175	0.811
NeeDrop-FP	300	300	✗	0.202	0.526	1.322

Here, ✓denotes true distance to shape or distance computation from dense input

(c) DFaust, results $\times 10^{-3}$, cf. [1].

†‡: Implemented in [1], where the sign of the distance is computed from the oriented normals provided with the scans (†) or locally with Jets [9] followed by a consistent orientation based on minimal spanning trees (‡).
§: Trained by ourselves using original code made available.
*: Model finetuned with $\sigma_{\mathbf{h}}/2$ from the initial model.
-FP: Model trained on the same (not re-sampled) points at each epoch.

Table 1. Reconstruction from complete point clouds.

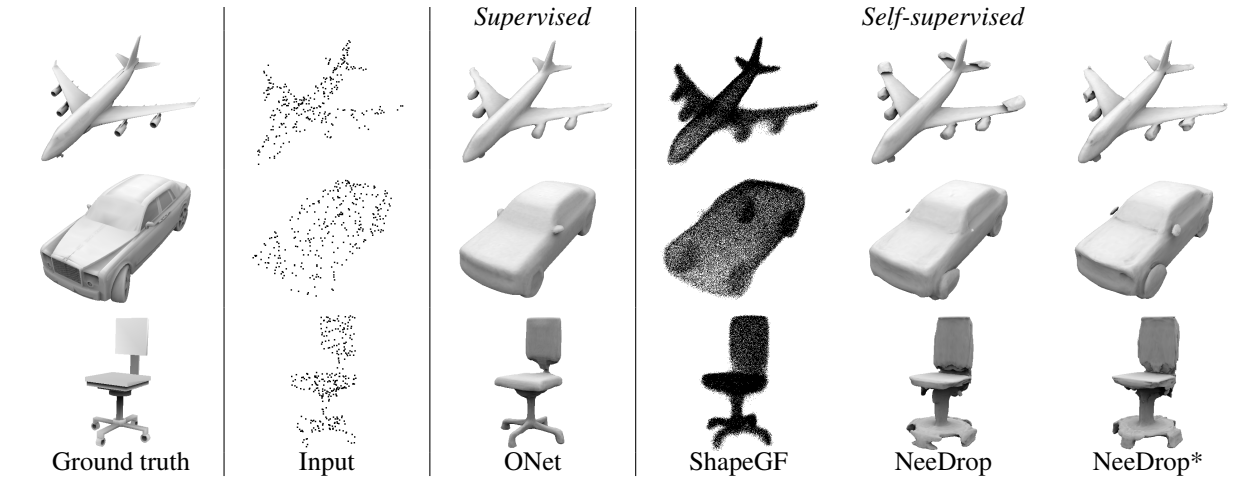
in NeeDrop, we obtain competitive results on both benchmarks. We are on par with DMC [43], and only outperformed by OccNet [46] among those methods. ShapeGF has a lower Chamfer distance but outputs noisy point clouds (Fig. 3(a)) from which constructing a mesh in not straightforward. In contrast, the occupancy predicted by NeeDrop allows for direct mesh extraction. (Trying to compare with IGR [29] on ShapeNet failed because we could not find parameters to reconstruct acceptable surfaces.)

Evaluation on DFaust. When comparing on DFaust (Tab. 1(c)), we include results for two variants of DeepSDF [48] reported in SAL [1]. The problem is that DeepSDF learns from a signed distance to a closed shape, whereas DFaust meshes are open. In the first variant DeepSDF†, the sign of the distance is computed based on the oriented normals provided with the scans; in the second variant DeepSDF‡, the normals are estimated locally with Jets [9] and oriented consistently using minimal spanning trees [1].

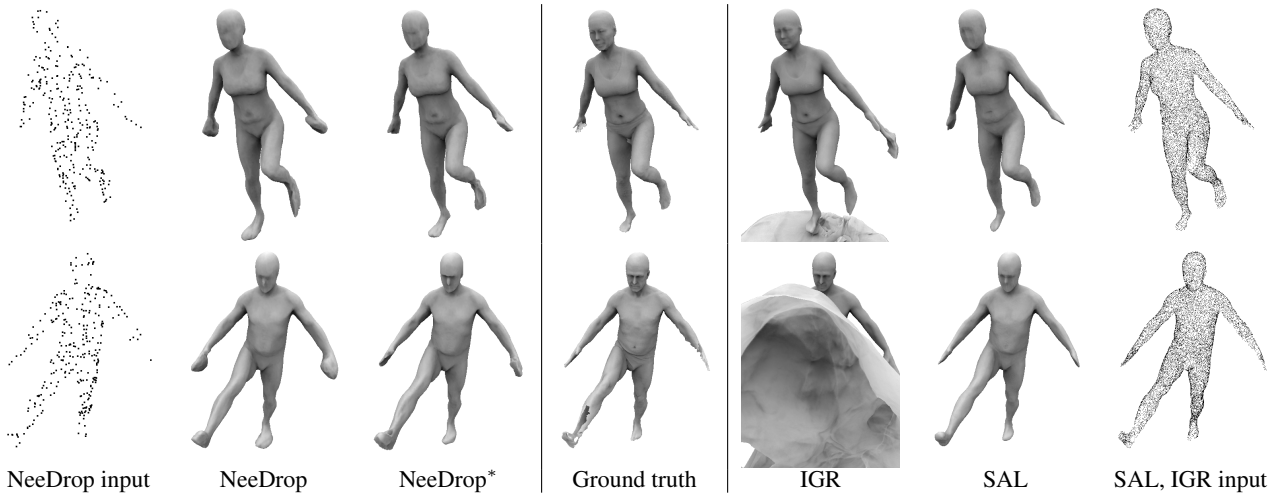
All methods reported in [1] were trained on very dense point cloud (16k points), while we target the much more difficult case of learning from low resolution point clouds. We thus also trained SAL [1] with the same amount of point as NeeDrop (300 pts), with supervision on the distance to the shape as well as without supervision, i.e., replacing the distance to the surface by the distance to the point cloud. Results in Tab. 1(c) with identical training settings (batch size 32, 2048 query pts) show that, as long as distances are measured w.r.t. the real surface, the performance of SAL degrades gracefully when the number of training points goes down from 16k to 300. However, when the distance to the surface has to be approximated from the training points, SAL fails to produce meshes when training only on 300 pts (and even on 1k pts). In fact, SAL may yield *unsigned* distance functions, which are also solutions to the loss; Marching Cubes then produce no surface. The specific network initialization of SAL tries to make it less likely, but unsigned distances still seems to be preferred solutions for sparse point clouds without exact distance supervision. The likely reason is that SAL solves an ill-posed problem: both the signed and the unsigned distance to the shape are solutions to the optimization problem. Obtaining a signed function is only favored by a particular initialization of the network, but not explicitly enforced during optimisation. In the self-supervised scenario with sparse point clouds, no regularisation prevents the implicit function to change sign between two points, where the actual surface should be, as the actual distance to the surface is not available. In contrast, NeeDrop enforces the surface to be supported by $P_{\mathcal{S}}$ and does not require any well-designed initialization.

SALD [2] does not offer code. Yet we expect sensitivity to the regularization weight for sparse data as a higher value is required to connect distant points with an iso-surface.

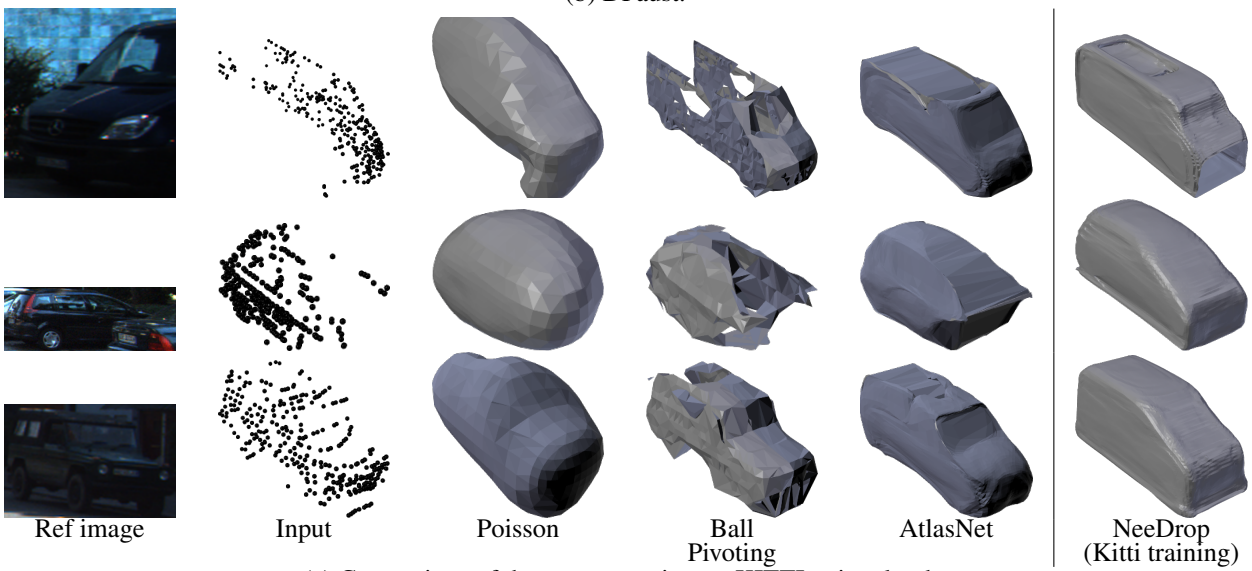
For IGR [29], we ran an experiment with 300 points as input for training (default is 8k pts). After a good training start, we observed divergence around epoch 200. We also used the the pre-trained model (trained with 8k points,



(a) ShapeNet



(b) DFaust.



(c) Comparison of the reconstruction on KITTI point clouds.

Figure 3. Qualitative comparison of NeeDrop to state-of-the-art methods, in ShapeNet, DFaust and KITTI.

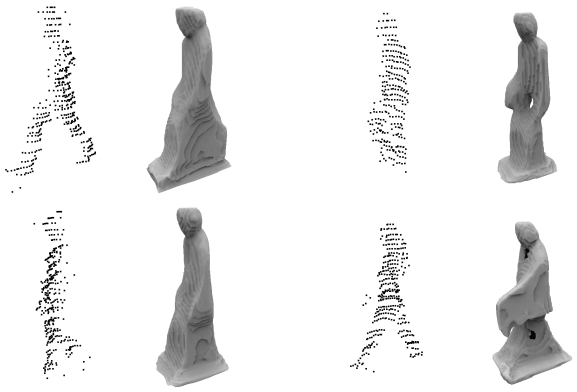


Figure 4. KITTI pedestrians: input point cloud and reconstruction.

tested with 16k) provided by the authors. IGR produces artifacts around the shape (see Fig. 3), which explains its worse quantitative performance in terms of Chamfer distance. However, we observe that it is the model which qualitatively recovers the best the details of the shapes.

Finetuning with a smaller $\sigma_{\mathbf{h}}$. This parameter is critical in our method. Its default value ensures a good convergence for all datasets we experimented with. Yet, small details may be lost with a too large $\sigma_{\mathbf{h}}$. In the spirit of curriculum learning, we finetune on some epochs the model previously learned with $\sigma_{\mathbf{h}}/2$, (named NeeDrop* in Tab. 1(c)) and observe a significant improvement. Fig. 3 visually illustrates the improvement on a sample from the DFaust dataset. With the initial model, hands and feet, i.e., thin surfaces, are estimated with surface blobs, which are reduced after finetuning. We also observe the appearance of the belly button and the eyebrows. Besides, our reconstruction with 300 pts is almost on par with SAL with 16,384 pts. See the supp. mat. for a study of needle crossing validity for varying $\sigma_{\mathbf{h}}$.

Fixed-point training. For comparison purposes, we sample new points on each shape at each epoch as in previous work. Yet, we also test our method with the same fixed 300 points sampled on each shape (NeeDrop-FP in Tab. 1). We observed that fixed-point training is stable. Besides, training with fixed points reaches almost the same performance as training with point re-sampling, except when the dataset is small (ShapeNet car subset vs the 13 ShapeNet classes); but even in this case, we still outperform a few supervised methods while not using exact distance or sign.

4.2. Real point clouds

Cars of KITTI. We perform a qualitative evaluation of the proposed method on the KITTI dataset. The input data are partial point clouds captured with lidars that either represent a car, a truck or a van, as extracted from the boxes of the detection dataset. In Figure 3(c), we present a comparison of our method against two direct (non-learned) methods, Poisson meshing [37] and Ball Pivoting [4], and a learning-based method, AtlasNet [31]. An image acquired by a RGB camera along with the point cloud is showed on Figure 3(c) for illustration purposes, but it is not used as input. In this experiment, we take advantage of the car's plan of symmetry for all methods: as KITTI cars are given in 3D boxes, with frame origin on the box center and axis y along the car length, we use as input a symmetrized point cloud w.r.t. the plane 0yz. For Poisson meshing and Ball Pivoting, reasonable hyper parameters were estimated through a grid search. AtlasNet was trained using a sphere primitive.

As expected, the direct methods fail to provide a proper extrapolation of the shape far from the symmetrized input point cloud. While Poisson meshing provides a closed mesh, it barely contains any detail. Ball Pivoting however produces a detailed mesh around the input point cloud but it fails to reconstruct the hidden parts. AtlasNet is able to take advantage of the symmetry but fails when the front or back of the car is missing (see first and last row in Figure 3(c)). In constrast to these methods, NeeDrop is able to take advantage of the existing symmetry, to produce a realistic car shape, and to reconstruct hidden parts of the cars.

KITTI pedestrians. As a last experiment on KITTI, we tried a very challenging target for shape reconstruction: pedestrians. Unlike for cars, we cannot symmetrize the input point cloud and must learn directly on the raw point cloud. In addition, pedestrian shapes are much more diverse than cars, which are globally convex. We see on Fig. 4 that, despite the task difficulty, our method is able to recover the coarse shape of a pedestrian. Our failure to capture human limbs on KITTI is not due to $\sigma_{\mathbf{h}}$, as our method works well on DFaust. It is mostly due to partial views, that cannot be symmetrized, unlike cars; it seems there are then many solutions, centered around merged limbs. Other factors are morphology variety, clothing and hand-carried items, while DFaust only features 10 different people in underwear.

5. Conclusion

In this study, we investigate self-supervised shape reconstruction and representation. To this end, we consider "needles" dropped in 3D space around the shape, for which we are able to estimate if they cross the surface or not. We also define a loss on these line segments suitable for neural network training. The resulting approach, NeeDrop, is the first totally self-supervised approach for learning an implicit occupancy function from a collection of shape available only as sparse point clouds. We show that our method is competitive with state-of-the-art supervised or partially supervised reconstruction methods. We conduct qualitative comparisons on datasets with ground-truth meshes and qualitative experiments on the challenging KITTI dataset. We successfully reconstruct plausible shapes from partial point cloud of cars and show promising results on the very challenging pedestrian category.

References

[1] Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020.

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

[3] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. A survey of surface reconstruction from point clouds. In *Computer Graphics Forum*, volume 36, pages 301–329. Wiley Online Library, 2017.

[4] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[5] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[6] Alexandre Boulch, Gilles Puy, and Renaud Marlet. Fkaconv: Feature-kernel alignment for point cloud convolution. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[7] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[8] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 364–381. Springer, 2020.

[9] Frédéric Cazals and Marc Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005.

[10] 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.

[11] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[12] 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.

[13] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, December 2020.

[14] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[15] 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 *European conference on computer vision*, pages 628–644. Springer, 2016.

[16] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[17] Angela Dai and Matthias Nießner. Scan2mesh: From unstructured range scans to 3d meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5574–5583, 2019.

[18] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5868–5877, 2017.

[19] Harm de Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron Courville. Modulating early visual processing by language. In *31st International Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pages 6597–6607, 2017.

[20] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3844–3852, 2016.

[21] Boyang Deng, JP Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. Nasa: Neural articulated shape approximation. In *16th European Conference on Computer Vision*, pages 612–628. Springer, 2020.

[22] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. In *International Conference on Learning Representations (ICLR)*, 2017.

[23] 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.

[24] Yutong Feng, Yifan Feng, Haoxuan You, Xibin Zhao, and Yue Gao. Meshnet: Mesh neural network for 3d shape representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8279–8286, 2019.

[25] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020.

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

[27] Benjamin Graham. Spatially-sparse convolutional neural networks. *arXiv preprint arXiv:1409.6070*, 2014.

[28] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9224–9232, 2018.

[29] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning*, pages 3789–3799. PMLR, 2020.

[30] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. 3D-CODED : 3D correspondences by deep deformation. In *European Conference on Computer Vision (ECCV)*, 2018.

[31] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 216–224, 2018.

[32] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *2017 International Conference on 3D Vision (3DV)*, pages 412–420. IEEE, 2017.

[33] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. *arXiv preprint arXiv:1810.09381*, 2018.

[34] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2307–2315, 2017.

[35] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020.

[36] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 364–375, 2017.

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

[38] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[39] Florian Kluger, Hanno Ackermann, Eric Brachmann, Michael Ying Yang, and Bodo Rosenhahn. Cuboids revisited: Learning robust 3d shape fitting to single rgb images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13070–13079, 2021.

[40] Lubor Ladicky, Olivier Saurer, SoHyeon Jeong, Fabio Maninchedda, and Marc Pollefeys. From point clouds to mesh using regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3893–3902, 2017.

[41] Georges-Louis Leclerc, comte de Buffon. *Supplément à l'Histoire Naturelle*, volume 4. Imprimerie royale, 1777.

[42] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on X-transformed points. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 820–830, 2018.

[43] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2916–2925, 2018.

[44] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

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

[46] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4460–4470, 2019.

[47] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4743–4752, 2019.

[48] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[49] Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J Black. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics (TOG)*, 34(4):1–14, 2015.

[50] Albert Pumarola, Stefan Popov, Francesc Moreno-Noguer, and Vittorio Ferrari. C-flow: Conditional generative flow models for images and 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7949–7958, 2020.

[51] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[52] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view CNNs for object classification on 3D data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5648–5656, 2016.

[53] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5105–5114, 2017.

[54] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 704–720, 2018.

[55] Danilo Jimenez Rezende, SM Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *NIPS*, 2016.

[56] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning deep 3D representations at high resolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3577–3586, 2017.

[57] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. Octnetfusion: Learning depth fusion from data. In *2017 International Conference on 3D Vision (3DV)*, pages 57–66. IEEE, 2017.

[58] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1746–1754, 2017.

[59] 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.

[60] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017.

[61] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[62] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2643, 2017.

[63] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017.

[64] Hao Wang, Nadav Schor, Ruizhen Hu, Haibin Huang, Daniel Cohen-Or, and Hui Huang. Global-to-local generative model for 3d shapes. *ACM Transactions on Graphics (TOG)*, 37(6):1–10, 2018.

[65] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018.

[66] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[67] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. DISN: Deep implicit surface network for high-quality single-view 3D reconstruction. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[68] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4541–4550, 2019.

[69] Li Yi, Boqing Gong, and Thomas Funkhouser. Complete & label: A domain adaptation approach to semantic segmentation of lidar point clouds. *arXiv preprint arXiv:2007.08488*, 2020.