HybridSDF: Combining Deep Implicit Shapes and Geometric Primitives for 3D Shape Representation and Manipulation

Subeesh Vasu[*,1] Nicolas Talabot[*,1] Artem Lukoianov[1,2] Pierre Baqué[2]
Jonathan Donier[2] Pascal Fua[1]

[1]CVLab, EPFL, {firstname.lastname}@epfl.ch

[2]Neural Concept, {firstname.lastname}@neuralconcept.com

Abstract

Deep implicit surfaces excel at modeling generic shapes but do not always capture the regularities present in manufactured objects, which is something simple geometric primitives are particularly good at. In this paper, we propose a representation combining latent and explicit parameters that can be decoded into a set of deep implicit and geometric shapes that are consistent with each other. As a result, we can effectively model both complex and highly regular shapes that coexist in manufactured objects. This enables our approach to manipulate 3D shapes in an efficient and precise manner.

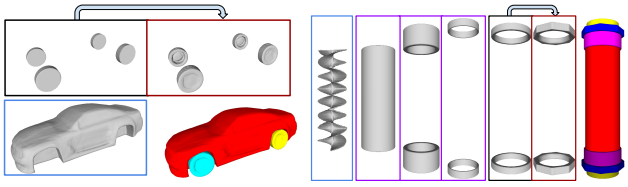
1. Introduction

Implicit surface representations have a long history [34] and have recently re-emerged in the form of signed distance functions [26] or occupancy fields [23, 4] that can be implemented by deep networks mapping 3D points to implicit function values. These representations are lightweight, high-fidelity, unlimited in resolution, naturally able to handle topology changes and, consequently, extremely popular in both computer vision and computer graphics, for example to explore a shape space for design purposes. However, because they represent objects as arbitrary, or generic, surfaces, they can easily fail to capture the regularities that are prevalent in manufactured objects and that are best represented by simple geometric shapes. As a result, the geometry and topology of complex objects such as those of Fig. 1 are not always faithfully preserved. On the other hand, approaches that rely purely on a collection of simple primitives often lack accuracy [27].

In this work, we propose a novel implicit representation combining geometric primitives and generic surfaces to enforce local regularities, consistency, and which enables parametric manipulation. To this end, we use three types of primitives defined as follow:

1. Generic primitives that may have arbitrarily complex shapes.
2. Simple geometric primitives, such as spheres and cylinders.
3. Primitives that bear a close resemblance to geometric ones but can deviate from them, such as car wheels that are almost but not quite cylinders.

These three kinds of primitives are represented implicitly through their Signed Distance Functions (SDF) that we refer to as *Generic-SDF*, *Geometric-SDF*, and *Geometry-assisted-SDF* respectively. Fig. 1 depicts their respective



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 1. *Modeling complex objects as a collection of primitives.* (a) We depict cars as the combination of a main body and the wheels. The car body is represented by a *Generic-SDF* (red) whose shape can change significantly. By contrast, the wheels are modeled by *Geometry-assisted-SDFs* (cyan and yellow) whose shapes are constrained not to differ excessively from that of the cylindrical *Geometric-SDFs* shown in the upper left corner. (b-h) This depicts a static mixer. It comprises a spiral helix (b) whose shape can also change significantly and is represented by a *Generic-SDF*. It is encased in a tube (c, red in h) with screws at both ends. The center tube along with the tube-like parts associated with the screw (d, e, yellow and violet in h) are represented by *Geometric-SDFs* of hollow cylinders. The screws also contains deformable rings modeled by *Geometry-assisted-SDFs* (g, blue in h) constrained not to deviate much from hollow cylinders (f).

[*]Equal contributions

Subeesh Vasu: subeeshvasu@gmail.com

Project page: https://ntalabot.github.io/hybridsdf/

roles in our data. They are parameterized in terms of implicit latent vectors and explicit geometric parameters, such as the radius and width of wheels, which are decoded together into 3D shapes.

This makes it possible to represent object parts that are true geometric primitives as such, parts that are similar but not identical by assisted primitives, and the remaining ones as generic surfaces that can assume any shape. Hence, our 3D models are interpretable in terms of parts that have a semantic meaning, unlike those of [13, 28], at minimal loss of accuracy. In short, our contribution is a novel SDF-based model that effectively combines the different kinds of primitives that are used for Computer Aided Design (CAD) objects to produce interpretable shapes that retain their internal consistency when edited. This opens up new avenues for user-friendly 3D shape manipulation and design with stronger surface regularities, unlike with purely deep implicit shapes, and automatic parameter detection from images and sketches, as we explore below.

2. Related work

Manufactured objects are often represented as CAD models made of geometric primitives. In recent years, the emergence of deep-learning has fostered a strong interest in developing more sophisticated primitives along new ways to represent complex surfaces and to combine these two kinds of approaches. We review them briefly below.

2.1. Primitive-based representations

Existing primitive-based methods for 3D shape representation aim to derive interesting abstractions by decomposing the 3D shapes into semantically meaningful simpler parts. In addition to simple primitives, more sophisticated ones such as cuboids [37, 39, 25, 36, 18, 35], superquadrics [29, 27], anisotropic Gaussians [9], or convexes [7] have been proposed. All these works rely on a single primitive type to represent the entire shape. Hence, for complex shapes, reconstruction accuracy directly depends on the number of primitives used. To improve on this, the approach of [28] defines a bijective function between traditional spherical primitives and a corresponding set of deformed shapes that can be arbitrarily complex.

As far as shape representation and shape manipulation are concerned, these deep learning-based methods have many limitations. They are typically trained in an unsupervised way to yield self-discovered arrangements of parts that do not necessarily correspond to semantically meaningful ones. Typically, the more primitives are used the more severe the problem becomes. For example, in [28], the latent space is parameterized by sphere parameters that have no direct relationship to the geometric parameters of the corresponding parts. Thus, there is no correspondence between the latent dimensions and true geometric parameters, such as location, orientation, or size, which is something we provide and can modify explicitly. This is partially addressed by works such as [2] that use a program composed of geometric operations to represent complex objects. However, such models are extremely hard to fit to data. Other works such as [19] aim to fit geometric surface patches to high resolution point clouds, but eventually answer a different problem as they do not learn a general shape representation that can be explored.

2.2. Explicit and implicit surfaces

Among existing 3D surface representations, meshes made of vertices and faces are one of the most popular and versatile types. There are excellent meshing algorithms [31] that can refine a mesh so that it accurately represents a known 3D shape, but there are far fewer that can produce a detailed 3D mesh of an initially unknown 3D shape of arbitrary topology. There have been many early attempts using only 3D meshes [22], but they require *ad hoc* heuristics that do not generalize well.

An alternative to surface meshes is to use an implicit descriptor where the surface is defined by the zero-crossing of a volumetric function $\Psi : \mathbb{R}^3 \rightarrow \mathbb{R}$ that may, for example, evolve over time [34]. The strength of this implicit representation is that the zero-crossing surface can change topology without explicit re-parameterization. Until recently, its main drawback was thought to be that working with volumes, instead of surfaces, massively increased the computational burden. This changed dramatically with the introduction of continuous deep implicit-fields. They represent 3D shapes as level sets of deep networks that map 3D coordinates to a signed distance function [26] or an occupancy field [23, 4]. This mapping yields a continuous shape representation that is lightweight but not limited in resolution. This representation has been successfully used for single-view reconstruction [23, 4, 38] and 3D shape-completion [5]. It has also been shown [33, 11] that 3D meshes could be extracted from these implicit representations without compromising differentiability. This makes it possible to back-propagate through the meshing of the SDF, such as when optimizing a 3D shape so that its projection match a specific contour [12] or to maximize its aerodynamic performance [1]. Recent works [30, 16, 20] have further extended the scope of implicit representations by developing network models that can yield fine-grained shape reconstructions.

2.3. Hybrid representations

Hybrid methods attempt to combine different representations to add functionalities. A common trend is to use a shared latent space across different representations such as voxel-based, image-based, or point-based representations to enhance performance in discriminative tasks as classifica-

tion and segmentation [15, 4, 24]. However, the *DualSDF* approach of [13] is the only one we know of that relies on an hybrid representation for shape generation and manipulation. It builds a shared latent space for two distinct shape representations: a sphere-based approximation and an implicit function-based high-fidelity reconstruction. The shared latent space couples the two representations and, as a result, the fine-grained model can be manipulated by changing the parameters of the spheres. However as the method of [28], *DualSDF* suffers from the limited interpretability of the spherical primitives that do not correspond to semantically meaningful parts and their geometric attributes, which is a key difference with our approach.

3. Approach

We first introduce our hybrid implicit representation, then our proposed model that decodes latent vectors and geometric parameters into complex 3D shapes, as illustrated in Fig. 2, and eventually the training losses.

3.1. HybridSDF representation

Signed Distance Functions (SDF) are mappings that associate to 3D points $\mathbf{p} \in \mathbb{R}^3$ a value in $\mathbb{R}$ that specifies their distance to a closed surface. By convention, the value is negative if $\mathbf{p}$ is inside the surface and positive otherwise. Hence, the surface is described by the zero-crossing of the SDF. This enables surfaces to change topology easily and differentiably, which can be exploited for shape optimization purposes. Furthermore, boolean operations such as union and intersection that are traditionally used in constructive solid geometry (CSG) can be implemented using simple $\min$ and $\max$ operators on SDFs, which makes it easy to combine them into a single final shape [14]. For these reasons, we represent our three kinds of primitives as their signed distance functions, which can be computed as

$$\mathbf{SDF}_{\text{prim}}(\mathbf{p}) = \Psi_{\text{prim}}(\mathbf{S}_{\text{prim}}, \mathbf{p}), \tag{1}$$

where Ψ_{prim} is a differentiable volumetric function, augmented to also take as input the parameters $\mathbf{S}_{\text{prim}}$ describing the shape. They can be explicit—such as radius and height—, implicit, such as latent vectors, or a combination of both. Additionally, we have rotation and translation parameters $\mathbf{R}$ and $\mathbf{T}$ that define the primitive's 6D pose within the overall shape space. In practice, they are implemented by transforming the query points $\mathbf{p}$ as

$$\mathbf{p}' = \mathbf{R}^\top(\mathbf{p} - \mathbf{T}), \tag{2}$$

which send them to the primitive's canonical space where the SDF is computed with Ψ_{prim}.

The SDF of our three primitive types are the following:

- **Generic-SDF** ($\mathbf{SDF}_{\text{generic}}$): arbitrarily complex primitive whose SDF is $\mathbf{SDF}_{\text{generic}}(\mathbf{p}) = \Psi_{\text{g}}(\mathbf{S}_{\text{g}}, \mathbf{p})$, where Ψ_{g} is an arbitrary volumetric function and $\mathbf{S}_{\text{g}}$ its shape parameters, usually a latent vector $\mathbf{LV}_{\text{generic}}$ or the combination of one with explicit parameters. Rotation and translation are not used for this primitive.

- **Geometric-SDF** ($\mathbf{SDF}_{\text{geom}}$): simple geometric primitive parameterized by a small number of shape parameters $\mathbf{S}$. It can be computed analytically through a known function Ψ_{geom}. For example, the SDF of a sphere is given by $\Psi_{\text{sphere}}(\mathbf{S}, \mathbf{p}) = \|\mathbf{p}\| - r$, where $\mathbf{S} = (r)$ contains the sphere radius. Similar functions can be easily written for cylinders and other CAD primitives [14].

- **Geometry-assisted-SDF** ($\mathbf{SDF}_{\text{assist}}$): primitive with a close resemblance to a simple geometric one. As with generic primitives, its SDF is given by $\mathbf{SDF}_{\text{assist}}(\mathbf{p}) = \Psi_{\text{a}}(\mathbf{S}_{\text{a}}, \mathbf{p})$, where Ψ_{a} is an arbitrary volumetric function and $\mathbf{S}_{\text{a}} = (\mathbf{LV}_{\text{assist}}, \mathbf{S}_{\text{assist}})$ is the concatenation of a latent vector and geometric shape parameters $\mathbf{S}_{\text{assist}}$. Additionally, we want to constrain its SDF to be close to its assisting geometry's. Therefore, we define another geometric-SDF $\mathbf{SDF}_{\text{geom}}$ with the same shape parameters $\mathbf{S}_{\text{assist}}$. It will only be used during training where we enforce $\mathbf{SDF}_{\text{assist}}(\mathbf{p}) \approx \mathbf{SDF}_{\text{geom}}(\mathbf{p})$ through the geometry-assistance loss $\mathcal{L}_{ga}$, as described in Section 3.3. Both share the same rotation and translation parameters. For instance, the car wheels in Fig. 1 are given by $\mathbf{SDF}_{\text{assist}}$, which are constrained to be similar to $\mathbf{SDF}_{\text{geom}}$ of the assisting cylinders. Note that only $\mathbf{SDF}_{\text{assist}}$ is used to build the final shape.

These three different primitive types are depicted by Fig. 1 and on the right side of Fig. 2. We use neural networks to implement the arbitrary volumetric functions Ψ_{g} and Ψ_{a}, which we refer to as deep implicit functions. Eventually, the full shape is taken as the union of all primitives, thus its SDF can be directly computed by applying the $\min$ operator over all SDFs:

$$\mathbf{SDF}_{\text{full}} = \min(\mathbf{SDF}_{\text{generic}}, \mathbf{SDF}_{\text{geom}}, \mathbf{SDF}_{\text{assist}}), \tag{3}$$

with

$$\mathbf{SDF}_{\text{geom}} = \min(\mathbf{SDF}^1_{\text{geom}}, \ldots, \mathbf{SDF}^{n_g}_{\text{geom}}), \tag{4}$$

$$\mathbf{SDF}_{\text{assist}} = \min(\mathbf{SDF}^1_{\text{assist}}, \ldots, \mathbf{SDF}^{n_a}_{\text{assist}}), \tag{5}$$

where n_g and n_a are the number of geometric and assisted primitives respectively. The shape is then defined as the zero-crossing of $\mathbf{SDF}_{\text{full}}$.

The number of primitives depends on the part decomposition of the shapes and, in practice, is known to the engineers performing the design. Generally, parts with strong regularities should be represented by either geometric or geometry-assisted primitives, while more complex ones should be generic primitives.

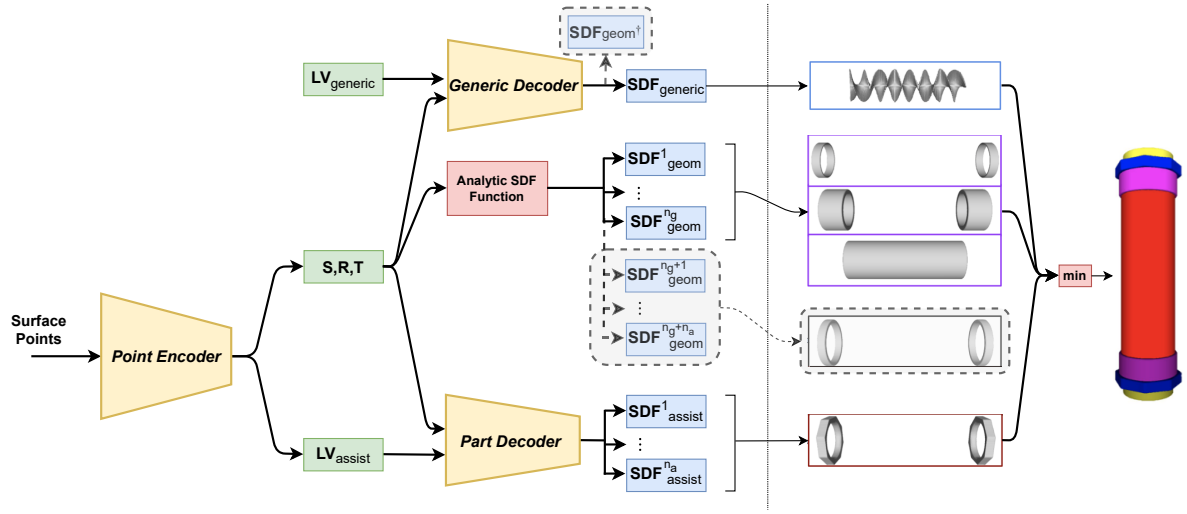


Figure 2. *HybridSDF architecture.* (*Left side*) The decoding part of the model takes as input a latent vector $\mathbf{LV}_{\text{generic}}$, geometric parameters $\mathbf{S}$, $\mathbf{R}$, and $\mathbf{T}$, along with an auxiliary latent vector $\mathbf{LV}_{\text{assist}}$. They are respectively decoded into the three types of primitives we have defined. A *Point Encoder* predicts $\mathbf{S}$, $\mathbf{R}$, $\mathbf{T}$, and $\mathbf{LV}_{\text{assist}}$ using 3D surface point clouds of the full shape. Grey boxes denote components used only for training purposes. The query point $\mathbf{p}$ is concatenated to the inputs of the decoders and analytic SDFs, but is not shown for clarity. (*Right side*) Example reconstruction for the mixer of Fig. 1. The blue, purple, and brown boxes contain the resulting surfaces: the generic primitive in blue, the geometric primitives in purple, and the geometry-assisted primitives in brown. Those in the black box denote the geometric shapes used to constrain the geometry-assisted primitives during training and are not used to reconstruct the final shape.

3.2. Network

We use a mix between an auto-decoder and an auto-encoder framework. Our shapes are decoded from a vector of parameters, as in [26, 13]. However, unlike in these works, we handle a combination of implicit and explicit, or latent and geometric, parameters and we rely on a disentangled representation where the different parameters account for separate dimensions. This makes it possible to support disentangled manipulation within that parameter space. As described below, we use a point cloud encoder to predict the parameters of the geometric and geometry-assisted primitives, while the latent vector of the generic one is trained in the auto-decoder fashion, as in [26]. In Appendix D, we also explore a purely auto-decoder variant of our approach where all parameters are instead predicted from a unique latent vector, as in [26, 13].

For the decoding, our network is made of multiple branches that decode the various primitive types for a given input, as illustrated in Fig. 2. The first branch decodes the concatenation of a latent vector $\mathbf{LV}_{\text{generic}}$ and the geometric parameters $\mathbf{S}$, $\mathbf{R}$, and $\mathbf{T}$ into the $\mathbf{SDF}_{\text{generic}}$ through the *Generic Decoder* network, whose architecture is taken from [26]. We add an auxiliary output $\mathbf{SDF}_{\text{geom}^\dagger}$ during training that must predict all geometric primitives in order to ensure the consistency between them and the generic primitive, as explained in Section 3.3. Then, we use analytical functions to compute the $\mathbf{SDF}_{\text{geom}}$ with the explicit parameters of the different geometric primitives. During training, this also computes the SDF of the assisting geometries. Finally, the last branch decodes $\mathbf{SDF}_{\text{assist}}$ of the geometry-assisted primitives through the *Part Decoder* network, similar in architecture to [26] but with only three layers as the shapes it produces are simpler. Its input is the concatenation of a latent vector $\mathbf{LV}_{\text{assist}}$ with the shape parameters $\mathbf{S}$ of the assisting geometry, while $\mathbf{R}$ and $\mathbf{T}$ are used to transform the position of the query point, as described above.

Point encoder. The explicit parameters $\mathbf{S}$, $\mathbf{R}$, and $\mathbf{T}$, as well as $\mathbf{LV}_{\text{assist}}$, are predicted by a *Point Encoder* that processes point clouds obtained from the training surfaces, as shown in Fig. 2. Its architecture is similar to that of PointNet [32] with fully connected layers and ReLU nonlinearities, and additional skip connections. We have empirically found this encoder to be necessary because directly training for these parameters in an auto-decoding framework would update them only once per epoch, which lead to early convergence in bad local minima, as shown in Appendix C.4. During training, the *Point Encoder* provides the values for $\mathbf{S}$, $\mathbf{R}$, $\mathbf{T}$, and $\mathbf{LV}_{\text{assist}}$, while in inference reconstruction it generates initial estimates that are optimized. It is however not needed for manipulation as users can instead manually edit the parameters. For different reconstruction tasks, such as single-image reconstruction which we explore below, it may be replaced by an encoder that accepts a different modality.

3.3. Training losses

HybridSDF learns to represent a set of 3D meshes from which SDF samples are extracted. It additionally leverages part labels, also in the form of meshes, to ensure that the primitives model the appropriate object parts. The model is trained on a weighted sum of losses with respect to its networks weights and latent vector $\mathbf{LV}_{\text{generic}}$, jointly with the *Point Encoder*. Mind that the formulas presented below are given for a single shape for clarity but are really summed over the training set, or rather a mini-batch per iteration. For each shape, we generate K sample points in the 3D space on which all losses are computed to speedup training. In the following text, it is assumed that $\mathbf{SDF}_\circ$ refers to a vector of the K points' signed distances, using the δ-clamped distances: $\text{clamp}_\delta(x) = \min(\delta, \max(-\delta, x))$, as in [26].

Reconstruction losses. In essence, the reconstruction is a regression problem aiming to make the predicted SDF as close as possible to the ground-truth (GT). We write

$$\mathcal{L}_r^f = \frac{1}{K} \left\| \mathbf{SDF}_{\text{full}} - \mathbf{SDF}_{\text{full}}^{\text{GT}} \right\|_1 , \tag{6}$$

that is, the $L1$-loss computed over the K samples for the full shape, where $\mathbf{SDF}_{\text{full}}^{\text{GT}}$ are the GT signed distance values.

For the parts, the labels can be noisy in practice, *e.g.* if they are automatically generated [17]. Therefore, we introduce the robust part reconstruction loss

$$\mathcal{L}_r^p = \sum_{\text{part}} \frac{1}{K'} \sum_{l=1}^{K'} Sort \left\{ \left| \mathbf{SDF}_{\text{part}} - \mathbf{SDF}_{\text{part}}^{\text{GT}} \right| \right\}_l , \tag{7}$$

where $\mathbf{SDF}_{\text{part}}$ is the geometric- or geometry-assisted-SDF corresponding to the part, $\mathbf{SDF}_{\text{part}}^{\text{GT}}$ its ground truth, $Sort$ rearranges the samples in increasing order of values, and $K' = \lfloor \gamma K \rfloor$, with $\gamma \in [0, 1]$. In other words, we only consider the K' samples with the lowest error. The hyper-parameter γ trades off robustness for reconstruction fidelity: the higher the noise, the lower γ should be set. For shape without part labels, we simply set $\mathcal{L}_r^p = 0$.

Geometry assistance. The geometry-assisted primitives are constrained to resemble their corresponding geometries by

$$\mathcal{L}_{ga} = \sum_{n=1}^{n_a} \frac{1}{K} \left\| \mathbf{SDF}_{\text{assist}}^n - \mathbf{SDF}_{\text{geom}}^{n_g+n} \right\|_1 , \tag{8}$$

where $\mathbf{SDF}_{\text{geom}}^{n_g+n}$ is the SDF of the geometric primitive used to constrain $\mathbf{SDF}_{\text{assist}}^n$.

Intersection loss. To prevent primitives from overlapping, we minimize an intersection loss between all pairs:

$$\mathcal{L}_{ic} = \sum_{x,y} \frac{1}{K} \left\| \Theta\big(\mathbf{SDF}_x, \mathbf{SDF}_y\big) \right\|_1 , \tag{9}$$

where Θ computes the overlap between two SDFs as

$$\Theta(\mathbf{SDF}_x, \mathbf{SDF}_y) = \max\Big(-\max\big(\mathbf{SDF}_x, \mathbf{SDF}_y\big), \mathbf{0}\Big) . \tag{10}$$

Consistency loss. In order to enforce the *Generic Decoder* to account for the explicit geometric parameters **S**, **R**, and **T**, a secondary output is added that must predict the SDF of all geometric primitives. This output is trained with the following loss:

$$\mathcal{L}_{cs} = \frac{1}{K} \left\| \mathbf{SDF}_{\text{geom}^\dagger} - \min_{n=1}^{n_g+n_a} \mathbf{SDF}_{\text{geom}}^n \right\|_1 , \tag{11}$$

where $\mathbf{SDF}_{\text{geom}^\dagger}$ denotes the auxiliary output as depicted in Fig. 2. As we show in Appendix C.3, we found this necessary in order to train the decoder to model the strong dependency of the generic primitive on the geometric parameters.

Regularization loss. Finally, we apply $L2$-regularization to all the latent vectors, $\mathbf{LV}_{\text{generic}}$ and $\mathbf{LV}_{\text{assist}}$, as in *DeepSDF*, and refer to the sum as $\mathcal{L}_{reg}$.

4. Experiments

Our approach is designed to provide stronger regularity for the parts corresponding to the geometric and geometry-assisted primitives, alongside precise and easy 3D shape manipulation via geometric parameter edition, by using explicit parameters describing shape attributes (**S**) and 6D pose (**R**, **T**). Additionally, it enables to automatically extract these parameters from single images and sketches. These facets of our work are explored below, and we provide additional results and an ablation study in Appendices E and C respectively.

Datasets. We report experimental results on two categories of objects: *Cars* from ShapeNet [3] to evaluate our method against existing work on frequently used shapes, and a new dataset of *Static Mixers* that are representative of complex manufactured objects.

Cars. We represent the cars in terms of a geometry-assisted-SDF for each wheel (using cylinders) and a generic-SDF for the car body, as shown in Fig. 1(a). We obtained the wheel labels from the noisy mesh labels of [17]. This gives us 2283 training shapes, 247 of which have part labels, and 500 test shapes.

Static Mixers. Used to mix fluids or powders that have different chemical properties, they comprise a static internal helix that is encased in a central tube with screws at both ends, as shown in Fig. 1(h). It is natural to represent the central tube and the tube-like parts associated with the screw by geometric-SDFs (hollow cylinders), the screw rings by geometry-assisted-SDFs (hollow cylinders), and the helix

	Mixers			Cars		
	CD↓	EMD↓	sIoU↑	CD↓	EMD↓	sIoU↑
DeepSDF	*2.27*	*2.76*	*86.3*	**11.9**	**1.44**	**54.9**
DualSDF	4.56	3.17	77.8	17.4	1.88	49.1
HierSQ	19.6	7.03	28.3	27.8	2.63	36.8
NeuralParts	6.43	13.8	44.1	25.2	2.17	40.0
HybridSDF	**1.57**	**2.51**	**90.9**	*12.5*	*1.48*	*52.9*

Table 1. *Reconstruction results on the test shapes.* We report average CD (30,000 points) multiplied by 10^4, EMD (10,000 points) multiplied by 10^2, and sIoU in [%]. Bold and italic numbers are best and second best performances respectively.

whose shape can vary most by a generic-SDF. Examples are provided in Appendix A. We use 1500 shapes for training and 450 for testing. During training, we use part labels for 100 of the exemplars.

Baselines. We compare against *DeepSDF* [26], an auto-decoding network predicting purely generic SDFs, similar to our *Generic Decoder*, *DualSDF* [13], an hybrid approach for shape manipulation, *HierSQ* [27], a primitive-based representation using a hierarchy of superquadrics, and *NeuralParts* [28], another hybrid method that uses network-predicted primitives to improve fidelity. For all of the above baselines, we used the code provided by the authors and we trained their networks on our data.

4.1. Shape reconstruction

We train our approach and the baselines on the training shapes, then we reconstruct the cars and mixers from the test sets and report quantitative results in Table 1. We use L_2-Chamfer Distance (CD), Earth Mover's Distance (EMD), and shell-IoU (sIoU) as in [33, 10]. Shell-IoU is the intersection over union computed on voxelized surfaces of reconstructed and ground truth shapes and should be maximized, in contrast to CD and EMD.

For the the mixers, our approach yields the best accuracy. This highlights that *HybridSDF* is well suited to manufactured objects partially defined by geometric primitives. On the cars, *DeepSDF* does marginally better, which may be counterintuitive given the qualitative mistakes *DeepSDF* sometimes makes, as can be seen in Fig. 3. We ascribe this to the fact that our learning task is harder. The *Generic Decoder* of *HybridSDF* must learn to represent the body of the car without wheels, even though only about 10% of the training shapes have part labels, which are noisy. Moreover, it has to adapt the car body to various wheel parameter values for shape edition, which *DeepSDF* cannot do.

Therefore, this slight accuracy decrease we observe for cars represent the trade-off between reconstruction accuracy and the ability to manipulate shapes explicitly, as in many related works [13, 35, 8]. Nonetheless, our representation yields an increase in quality and realism of the geometric and geometry-assisted parts. For cars, this translates to a better separation between body and wheels and to more cylindrical wheels thanks to geometry-assistance, as shown in Fig. 4, which also highlights that our method can exploit noisy labels generated by external segmentation pipelines. For the mixers, this manifests itself as the excellent surface regularity that the the geometrically assisted primitives provide. Furthermore, our approach allows parametric shape manipulation—*e.g.*, changing wheel sizes or mixer radii—while incurring only minimal loss in accuracy in comparison to *DeepSDF*, unlike related work such as *DualSDF* that we discuss next.

4.2. Parametric shape manipulation

A proxy representation was introduced in *DualSDF* [13] to offer editing abilities on top of *DeepSDF*. However, it cannot easily be used to target specific parameter values, while our approach allows more accurate shape manipulation given inputs in the form of intepretable geometric parameters such as translation, rotation, and size. To demonstrate this, we manipulate the central tube of static mixers and compare against *DualSDF*. For each shape in the training set, we randomly sample a target outer radius and thickness by perturbing the ground truth parameters. Then, the shape is automatically optimized to fit the new parameters. For *DualSDF*, we optimize a sphere from its coarse representation according to Equation (15) in [13]. The sphere is automatically selected as the farthest one to the vertical axis, amongst those that are close to the middle horizontal plane. Editing using *HybridSDF* is performed by directly changing the values of the geometric parameters **S**, **R**, and **T** and then decoding a new shape. It requires no optimization and is therefore both faster and exact.

The radius and thickness of the optimized tube are detected through a circle Hough transform on a horizontal cross-section of the SDF. We report the relative error to target parameters for the whole dataset in Table 2. *DualSDF* fails to manipulate the shapes to target specific parameter values, unlike our approach. This demonstrates that *HybridSDF* is better suited for applications requiring more precise and explicit control, for example in engineering contexts. Nonetheless, we note that *DualSDF* and our approach can also be complementary. Indeed, they can be integrated to leverage their respective shape edition abilities, as we show in the Appendix F.

In practice, the disentanglement of our approach enables a direct control of the geometric parameters, while independently maintaining or editing features from the implicit latent dimensions. This is illustrated in Fig. 5 where the type of helices and car body is maintained when manipulating the explicit parameters using *HybridSDF*.

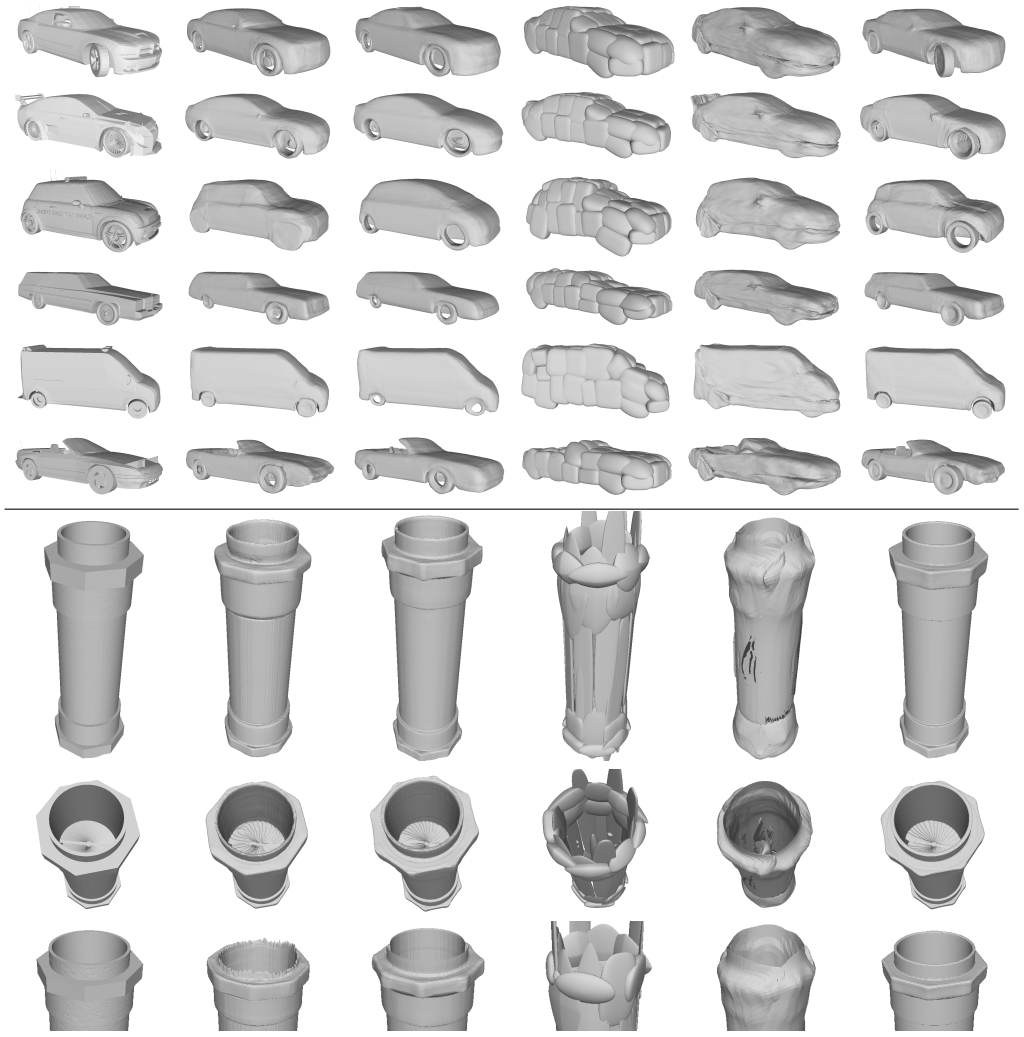


Figure 3. *Qualitative comparison of reconstruction results.* (*Top rows*) Ground-truth for different cars and corresponding reconstructions. *DeepSDF*, *DualSDF*, and *HybridSDF* deliver the most visually appealing results. However, *HybridSDF*, unlike *DeepSDF* and *DualSDF*, correctly captures the wheel orientation on the first example, and the wheel wells on the third. (*Bottom rows*) Ground-truth for different mixers and reconstructions. *DeepSDF*, *DualSDF*, and *HybridSDF* again yield the best looking results but both *DeepSDF* and *DualSDF* produce surface irregularities near the screw, while *HybridSDF* does not. Interestingly, *HierSQ* and *NeuralParts* completely fails to represent the helix, underscoring that iso-primitives are not the optimal way to represent shapes.

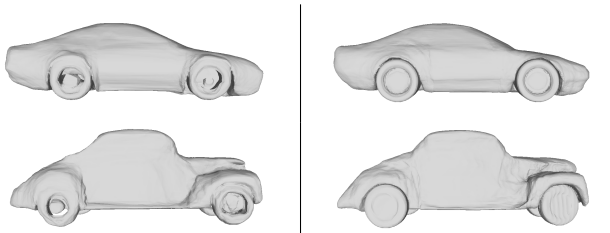
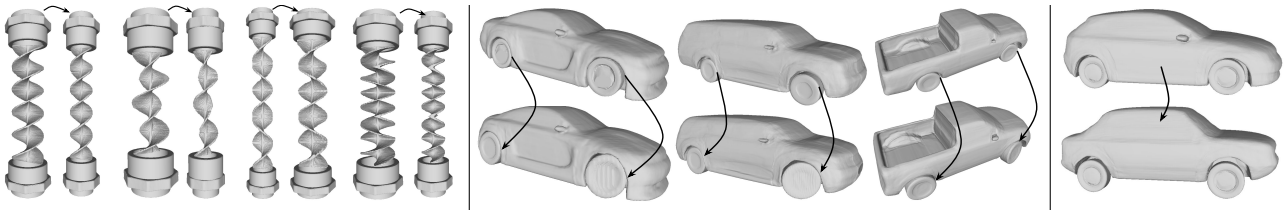


Figure 4. *Wheels in reconstructed test shapes.* Unlike some *DeepSDF* reconstructions, the wheels produced by *HybridSDF* are well defined, cylindrical, and separated from the car body.

	Error [%]↓	
	Radius	Thickness
DualSDF	12.7	37.8
HybridSDF	**0.4**	**4.5**

Table 2. *Manipulation performance on the Mixers.* Numbers are the relative error to the target parameter values, averaged through the dataset. Note that *HybridSDF* errors should be 0% as the target parameters are directly used for reconstruction. Instead, they reflect the imprecision in our radius and thickness detection pipeline.



(a) (b) (c)

Figure 5. *Parametric shape manipulation.* In each example, we show arrows going from the source object to the one obtained after manipulation using *HybridSDF*. (a) The geometric parameters **S**, **R**, and **T** of the static mixers are modified. The ring and helix adapt while retaining their original key features such as their type and number of turns. (b) The parameters of the wheels are edited and the car body changes accordingly: the wheel wells and the general size adapt (*e.g.*, the third car widens) while the type of the car persists. (c) The latent of the car body $\mathbf{LV}_{\text{generic}}$ is changed, but the parameters of the wheels are kept constant. A new car that fit the same wheels is obtained. For an equivalent experiment with the helices of the mixers, see Fig. 7.

$r_{\text{ours}} = 0.114$ $r_{\text{true}} = 0.113$ | $r_{\text{ours}} = 0.143$ $r_{\text{true}} = 0.146$ | $r_{\text{ours}} = 0.172$ $r_{\text{true}} = 0.176$

Figure 6. *Single-view reconstruction.* (*Top row*) Input images. (*Middle row*) Reconstructions using *HybridSDF*. (*Bottom row*) The estimated wheel radius r_{ours} and the ground truth r_{true}, as measured on the ground truth meshes.

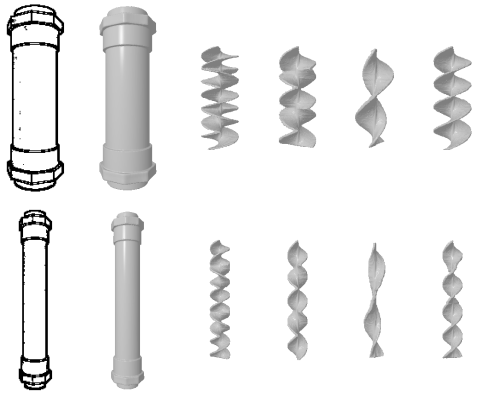


Figure 7. *Mixer reconstruction from sketches.* (*Left column*) Input sketches. (*Second column*) Reconstructions using *HybridSDF*. (*Remaining columns*) Helices reconstructed by combining the predicted tube parameters with four latent vectors $\mathbf{LV}_{\text{generic}}$. The helices adapt to the tube size while preserving their features.

4.3. Single-image reconstruction

We use the learned parameterized representation of the cars and mixers for single-view reconstruction. Similarly to [4], we train *encoders* to map an image space to *HybridSDF*'s parameter space. This eventually permits our method to reconstruct the 3D shapes while automatically recovering some part geometric parameters. For cars, we train an *encoder* to predict all latent vectors and geometric parameters from renderings obtained from [6]. We show test reconstruction results in Fig. 6. Not only does this yield realistic shapes but it also estimates the geometric parameters that characterize them, such as the radius of the wheels. For mixers, only the outer parts are visible in sketches such as those of Fig. 7. We therefore train another *encoder* to predict the parameters **S**, **R**, and **T**, and latent $\mathbf{LV}_{\text{assit}}$ from the sketches. One can then manually choose the central helix by supplying a latent vector $\mathbf{LV}_{\text{generic}}$, and it will directly fit to the tube length and width. This could lead to automatic retrieval of CAD objects from engineering drawings.

5. Conclusion

We have proposed a novel approach to combine deep implicit surfaces and geometric primitives for parametric manipulation that preserves representation accuracy, interpretability, and enforces consistency for complex objects made of multiple parts. Our disentangled representation makes it easy to define the geometric parameters of several parts and have the rest of the object adapt to these specifications, or to extract these parameters and reconstruct shapes from images and sketches.

In the current version of our architecture, using parameters significantly outside the training range can lead to failure, while consistency between parts is enforced by minimizing loss functions. In future work, we will look into improved generalization and imposing such consistency constraints as hard constraints [21] so that our loss have fewer terms. A further extension will include additional operations between parts, such as difference, similar to what is found in constructive solid geometry.

Acknowledgments This work was supported in part by the Swiss Innovation Agency.

References

[1] P. Baqué, E. Remelli, F. Fleuret, and P. Fua. Geodesic Convolutional Shape Optimization. In *International Conference on Machine Learning*, 2018. 2

[2] Dan Cascaval, Mira Shalah, Phillip Quinn, Rastislav Bodik, Maneesh Agrawala, and Adriana Schulz. Differentiable 3D CAD Programs for Bidirectional Editing. *arXiv Preprint*, 2021. 2

[3] A. Chang, T. Funkhouser, L. G., P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An Information-Rich 3D Model Repository. In *arXiv Preprint*, 2015. 5

[4] Z. Chen and H. Zhang. Learning Implicit Fields for Generative Shape Modeling. In *Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2, 3, 8

[5] J. Chibane, T. Alldieck, and G. Pons-Moll. Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion. In *Conference on Computer Vision and Pattern Recognition*, 2020. 2

[6] C. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3DR2N2: A Unified Approach for Single and Multi-View 3D Object Reconstruction. In *European Conference on Computer Vision*, 2016. 8

[7] B. Deng, K. Genova, S. Yazdani, S. Bouaziz, G. Hinton, and A. Tagliasacchi. Cvxnet: Learnable Convex Decomposition. In *Conference on Computer Vision and Pattern Recognition*, pages 31–44, 2020. 2

[8] M. Gadelha, G. Gori, D. Ceylan, R. Mech, N. Carr, T. Boubekeur, R. Wang, and S. Maji. Learning Generative Models of Shape Handles. In *Conference on Computer Vision and Pattern Recognition*, pages 402–411, 2020. 6

[9] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. Funkhouser. Learning Shape Templates with Structured Implicit Functions. In *International Conference on Computer Vision*, pages 7154–7164, 2019. 2

[10] B. Guillard, E. Remelli, and P. Fua. UCLID-Net: Single View Reconstruction in Object Space. In *Advances in Neural Information Processing Systems*, 2020. 6

[11] B. Guillard, E. Remelli, A. Lukoianov, S. Richter, T. Bagautdinov, P. Baque, and P. Fua. Deepmesh: Differentiable Iso-Surface Extraction. In *arXiv Preprint*, 2021. 2

[12] B. Guillard, E. Remelli, P. Yvernay, and P. Fua. Sketch2mesh: Reconstructing and Editing 3D Shapes from Sketches. In *International Conference on Computer Vision*, 2021. 2

[13] Z. Hao, H. Averbuch-Elor, N. Snavely, and S. Belongie. Dualsdf: Semantic Shape Manipulation Using a Two-Level Representation. In *Conference on Computer Vision and Pattern Recognition*, pages 7631–7641, 2020. 2, 3, 4, 6

[14] S. Haugo, A. Stahl, and E. Brekke. Continuous Signed Distance Functions for 3D Vision. In *International Conference on 3D Vision*, pages 116–125, 2017. 3

[15] V. Hegde and R. Zadeh. Fusionnet: 3D Object Classification Using Multiple Data Representations. *CoRR*, abs/1607.05695, 2016. 3

[16] M. Ibing, I. Lim, and L. Kobbelt. 3D Shape Generation with Grid-Based Implicit Functions. In *Conference on Computer Vision and Pattern Recognition*, pages 13559–13568, 2021. 2

[17] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri. 3D Shape Segmentation with Projective Convolutional Networks. In *Conference on Computer Vision and Pattern Recognition*, pages 3779–3788, 2017. 5

[18] F. Kluger, H. Ackermann, E. Brachmann, M. Yang, and B. Rosenhahn. Cuboids Revisited: Learning Robust 3D Shape Fitting to Single RGB Images. In *Conference on Computer Vision and Pattern Recognition*, pages 13070–13079, 2021. 2

[19] Eric-Tuan Lê, Minhyuk Sung, Duygu Ceylan, Radomir Mech, Tamy Boubekeur, and Niloy J Mitra. CPFN: Cascaded Primitive Fitting Networks for High-Resolution Point Clouds. In *Conference on Computer Vision and Pattern Recognition*, pages 7457–7466, 2021. 2

[20] S. Liu, H. Guo, H. Pan, P. Wang, X. Tong, and Y. Liu. Deep Implicit Moving Least-Squares Functions for 3D Reconstruction. In *Conference on Computer Vision and Pattern Recognition*, pages 1788–1797, 2021. 2

[21] P. Marquez-Neila, M. Salzmann, and P. Fua. Imposing Hard Constraints on Deep Networks: Promises and Limitations. In *CVPR Workshop on Negative Results in Computer Vision*, 2017. 8

[22] T. Mcinerney and D. Terzopoulos. Topology Adaptive Deformable Surfaces for Medical Image Volume Segmentation. *IEEE Transactions on Medical Imaging*, 18(10):840–850, 1999. 2

[23] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 1, 2

[24] S. Muralikrishnan, V. G. Kim, M. Fisher, and S. Chaudhuri. Shape Unicode: A Unified Shape Representation. In *Conference on Computer Vision and Pattern Recognition*, pages 3790–3799, 2019. 3

[25] C. Niu, J. Li, and K. Xu. Im2struct: Recovering 3D Shape Structure from a Single Rgb Image. In *Conference on Computer Vision and Pattern Recognition*, pages 4521–4529, 2018. 2

[26] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2, 4, 5, 6

[27] D. Paschalidou, L. V. Gool, and A. Geiger. Learning Unsupervised Hierarchical Part Decomposition of 3D Objects from a Single Rgb Image. In *Conference on Computer Vision and Pattern Recognition*, pages 1060–1070, 2020. 1, 2, 6

[28] D. Paschalidou, A. Katharopoulos, A. Geiger, and S. Fidler. Neural Parts: Learning Expressive 3D Shape Abstractions with Invertible Neural Networks. In *Conference on Computer Vision and Pattern Recognition*, pages 3204–3215, 2021. 2, 3, 6

[29] D. Paschalidou, A. O. Ulusoy, and A. Geiger. Superquadrics Revisited: Learning 3D Shape Parsing Beyond Cuboids. In *Conference on Computer Vision and Pattern Recognition*, pages 10344–10353, 2019. 2

[30] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. Convolutional Occupancy Networks. In *European Conference on Computer Vision*, pages 523–540, 2020. 2

[31] J. Peters and U. Reif. *Subdivision Surfaces*. Springer, 2008. 2

[32] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2017. 4

[33] E. Remelli, A. Lukoianov, S. Richter, B. Guillard, T. Bagautdinov, P. Baque, and P. Fua. Meshsdf: Differentiable Iso-Surface Extraction. In *Advances in Neural Information Processing Systems*, 2020. 2, 6

[34] J. A. Sethian. *Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999. 1, 2

[35] D. Smirnov, M. Fisher, V. G. Kim, R. Zhang, and J. Solomon. Deep Parametric Shape Predictions Using Distance Fields. In *Conference on Computer Vision and Pattern Recognition*, 2020. 2, 6

[36] C. Sun, Q. Zou, X. Tong, and Y. Liu. Learning Adaptive Hierarchical Cuboid Abstractions of 3D Shape Collections. *ACM Transactions on Graphics*, 38(6):1–13, 2019. 2

[37] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik. Learning Shape Abstractions by Assembling Volumetric Primitives. In *Conference on Computer Vision and Pattern Recognition*, pages 2635–2643, 2017. 2

[38] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann. DISN: Deep Implicit Surface Network for High-Quality Single-View 3D Reconstruction. In *Advances in Neural Information Processing Systems*, 2019. 2

[39] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem. 3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks. In *International Conference on Computer Vision*, 2017. 2