SPAGHETTI: Editing Implicit Shapes Through Part Aware Generation

AMIR HERTZ, Tel Aviv University, Israel
OR PEREL, Tel Aviv University and NVIDIA, Israel
RAJA GIRYES, Tel Aviv University, Israel
OLGA SORKINE-HORNUNG, ETH Zurich, Switzerland
DANIEL COHEN-OR, Tel Aviv University, Israel

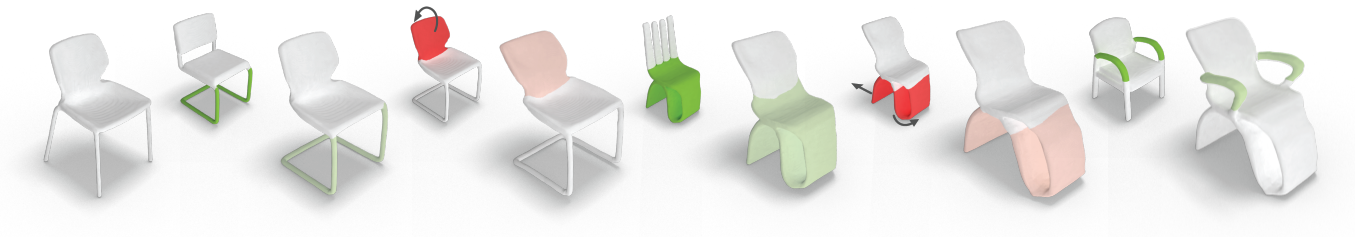


Fig. 1. Design exploration with SPAGHETTI. Using our method, the user can easily compose new shapes (in grey) out of parts (in green) taken from reference shapes or make further local adjustments of selected parts (in red).

Neural implicit fields are quickly emerging as an attractive representation for learning based techniques. However, adopting them for 3D shape modeling and editing is challenging. We introduce a method for **E**diting **I**mplicit **S**hapes **T**hrough **P**art **A**ware **G**enera**T**ion, permuted in short as SPAGHETTI. Our architecture allows for manipulation of implicit shapes by means of transforming, interpolating and combining shape segments together, without requiring explicit part supervision. SPAGHETTI disentangles shape part representation into extrinsic and intrinsic geometric information. This characteristic enables a generative framework with part-level control. The modeling capabilities of SPAGHETTI are demonstrated using an interactive graphical interface, where users can directly edit neural implicit shapes. Our code, editing user interface demo and pre-trained models are available at github.com/amirhertz/spaghetti.

CCS Concepts: • **Computing methodologies** → **Shape modeling**; **Neural networks**; **Learning latent representations**; *Graphics systems and interfaces*.

Additional Key Words and Phrases: neural networks, shape synthesis, shape modeling

ACM Reference Format:
Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. 2022. SPAGHETTI: Editing Implicit Shapes Through Part Aware Generation. *ACM Trans. Graph.* 41, 4, Article 106 (July 2022), 20 pages. https://doi.org/10.1145/3528223.3530084

Authors' addresses: Amir Hertz, Tel Aviv University, Israel, amirhertz@mail.tau.ac.il; Or Perel, Tel Aviv University and NVIDIA, Israel, orr.perel@gmail.com; Raja Giryes, Tel Aviv University, Israel, raja@tauex.tau.ac.il; Olga Sorkine-Hornung, ETH Zurich, Switzerland, sorkine@inf.ethz.ch; Daniel Cohen-Or, Tel Aviv University, Israel, cohenor@gmail.com.

0730-0301/2022/7-ART106
https://doi.org/10.1145/3528223.3530084

1 INTRODUCTION

In recent years, there is a surge of interest in applying neural implicit fields to represent 3D shapes and scenes. By design, such parameterizations do not limit the shape resolution, thereby allowing to faithfully recover the underlying continuous surface or volume.

The learned nature of neural implicit fields also promotes them as naturally compressed representations, capable of capturing high resolution details with low memory cost. Altogether, these attributes make them an intriguing medium for developing novel generative techniques.

Most of the recent research efforts have been focused on refining the quality of represented signals [Sitzmann et al. 2020; Takikawa et al. 2021; Martel et al. 2021] or leveraging implicit representations for shape reconstruction [Erler et al. 2020; Genova et al. 2020; Chabra et al. 2020a]. While implicit surface representations are well established in classical shape modeling literature (e.g., [Cani et al. 2008; Schmidt and Wyvill 2011]), so far only little attention has been given to editing of neural implicit shapes [Hao et al. 2020]. In particular, conditioning the form of a neural implicit shape on specified user controls is not straightforward, further hindering the adoption in 3D creative applications.

In this paper, we introduce SPAGHETTI, a novel generative model that supports direct editing of neural implicit shapes. Our framework allows for part level of control by (i) applying transformations on local areas of the generated object; and (ii) mixing and interpolating segments of different shapes.

The editing power of our model comes from its dual-level disentanglement. First, our network learns to separate local part representations from each other. This is essential, as modifications to a single part should have little effect on the rest. Our network is trained to achieve this separation without explicit part supervision. Second, each part representation is factored into intrinsic and extrinsic components, respectively controlling its detailed surface geometry and embedding in 3D space. Doing so allows our learning process to

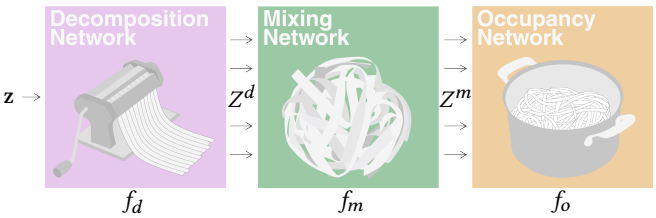


Fig. 2. Method overview. Our implicit shape generative model learns to decompose a global shape embedding $\mathbf{z}$ into a set of embeddings Z^d that correspond to distinct 3D parts. Then, a mixing network outputs a set of contextual embeddings Z^m. Finally the implicit shape is given by an occupancy network f_o that is conditioned on the contextual part embeddings.

introduce local affine transformations on shape parts while keeping them within the data distribution (see Figure 1).

As illustrated in Figure 2, our architecture can be roughly divided to three steps. At the beginning of our pipeline, the Decomposition Network receives a latent shape embedding $\mathbf{z}$ and projects it onto a set of latent codes Z^d. Each $\mathbf{z}^d \in Z^d$ corresponds a distinct part of the 3D shape, with its latent representation of surface information and global transformation. Then, the Mixing Network, based on a transformer encoder architecture, processes Z^d and outputs contextual codes Z^m. Finally, the implicit shape is generated by an Occupancy Network, in the form of a transformer decoder, where a query coordinate is weighted according to Z^m to output its occupancy value.

During inference, the user can control the 3D shape by modifying its components in their raw latent state. The user can modify the local extrinsic properties of each part, as well as add, remove or replace components taken from other shapes. After each editing step, the modified parts are re-composed into a new implicit shape. See examples in Figure 1.

To enable the editing of new shapes that were not seen during training, we introduce a *shape inversion* optimization, which finds the matching part codes for a given unseen shape. Our mid latent representation of disentangled part embeddings facilitates the extrapolation outside the training data and enables high quality inversions.

We demonstrate the effectiveness of our method using a graphical user interface, where SPAGHETTI is accelerated by an Octree, to allow for an interactive editing experience of 3D implicit shapes. Code, pre-trained models and demos will be made publicly available upon publication.

2 RELATED WORK

3D generative models. The introduction of deep generative models within the computer vision community [Kingma and Welling 2014; Goodfellow et al. 2014; Dinh et al. 2014; Radford et al. 2016; Oord et al. 2016] quickly spawned a rich line of works capable of producing visually appealing images. Despite their success on images, adapting these models to generate 3D shapes proved to be non-trivial. Wu et al. [2016] were the first to extend the unconditional generative adversarial network (GAN) setting to 3D, using volumetric convolutions to create a voxel-based generative model. A followup by Liu et al. [2017] allows users a finer granularity of control by interactively painting a voxel grid and converting it to a high-resolution shape using a GAN. Girdhar et al. [2016] leverage a shared latent space of 2D views and voxels to generate shapes conditioned on images. These ideas were later expanded beyond voxels to the non-Euclidean domains of point clouds [Li et al. 2018; Yang et al. 2019] and meshes [Ranjan et al. 2018; Tan et al. 2018; Nash et al. 2020]. See Chaudhuri et al. [2019] for a contemporary introduction to the field.

3D part-level representation. Decomposing 3D shapes into parts was traditionally proposed as means of improving shape representation, usually to facilitate downstream tasks involving recognition, retrieval or manipulation [Hoffman and Richards 1984; Mitra et al. 2014; Huang et al. 2014]. With the rising popularity of data-driven approaches [Kalogerakis et al. 2010; Kim et al. 2013], neural architectures have also been augmented with part level segmentation as a means of enriching shape representations [Qi et al. 2016; Wang et al. 2019b].

An emerging trend promotes the encoding of shape parts in a joint latent space to facilitate better generalization of the representation to novel, unseen shapes [Nash and Williams 2017]. Other works attempt to achieve this goal by addressing the geometry and structural composition of parts separately [Gao et al. 2019]. Although these approaches are able to capture fine geometric details, they require part supervision. To avoid the need for labels, shape parts can be composed as deep hierarchies using binary-space partitions [Chen et al. 2020], recursive neural networks [Li et al. 2017, 2019; Paschalidou et al. 2020b], and Gaussian mixture models (GMM) [Achlioptas et al. 2018; Hertz et al. 2020]. Our work uses a part decomposition network that is close in spirit to [Hertz et al. 2020], but we opt for a simplified variant of a flat GMM, rather than a hierarchy.

Neural implicit shapes. Non-neural implicit representations have been developed and used in a variety of 3D applications, such as reconstruction, modeling and morphing [Cohen-Or et al. 1998; Carr et al. 2001; Turk and O'Brien 2005; Schmidt and Wyvill 2011]. In these classical works, 3D shapes are implicitly approximated through a pre-defined family of functions, or interpolated distance fields. With the advances of learning based techniques, coordinate-based neural networks gain attention as powerful parameterizations able to fit arbitrary signals, and in particular, implicit shapes. Neural implicit functions are used to capture the geometry of 3D shapes as occupancy indicator functions [Chen and Zhang 2019; Mescheder et al. 2019; Peng et al. 2020], level sets of distance fields [Park et al. 2019; Atzmon and Lipman 2020], or indirectly as volumetric radiance fields [Mildenhall et al. 2020; Zhang et al. 2020]. Pioneering works by Chen and Zhang [2019], Mescheder et al. [2019] and Park et al. [2019] show how multiple shapes can be decoded with a single network by encoding shapes as latent codes. They are able to achieve high-quality shape reconstruction with a continuous representation that learns priors from a 3D dataset.

Neural implicit representations have been expanded to hybrid representations based on spatial structures of latent codes. Peng et al. [2020], Jiang et al. [2020b] and Chabra et al. [2020b] promote the usage of local dense grids as a means of introducing an inductive bias of spatial repetitions. Martel et al. [2021] and Takikawa et al. [2021] use hierarchical octree representations to achieve faster rendering

and higher reconstruction quality. Despite their promising results, directly applying these methods to shape generation and editing is non-trivial.

Various works advocate sparse representations where part templates are decoded with neural implicit functions. Yang et al. [2018] and Groueix et al. [2018] learn 3D shape representations as surface elements parameterized by 2D to 3D coordinate-based mapping networks. Closer to our work, SIF [Genova et al. 2019b] and LDIF [Genova et al. 2020] study representations that decompose shapes into coarse Gaussian template parameterizations, further localized with implicit surface functions to obtain a full shape reconstruction. A key observation is that the usage of template parts allows for smooth interpolation in latent space between shapes, suggesting that implicit part templates are promising for geometric editing. Unlike these works, we avoid the usage of encoders in favor of a more direct method to achieve decomposition into Gaussian parts. In SPAGHETTI, we also augment part representations with contextual information to achieve global coherency, suitable for shape editing. See Xie et al. [2021] for summary of latest achievements in this field.

Interactive editing. In this paper, we refer to editing as manipulation of non-atomic 3D shape elements by means of mixing parts [Funkhouser et al. 2004] or through guided transformations, which leverage prior knowledge about the shape structure [Gal et al. 2009; Fish* et al. 2014; Mitra et al. 2014]. Prominent examples of neural modifiers include editing of complex shapes through learned approximations of coarse primitives [Tulsiani et al. 2017; Hao et al. 2020], deformation networks [Wang et al. 2019a; Yifan et al. 2020; Jiang et al. 2020a; Uy et al. 2021] or segmenting and manipulating semantic parts [Wei et al. 2020]. To extend these methods to new shapes, unseen during training, some degree of shape encoding or inversion is required.

Neural solutions for mixing parts have been investigated in MR-GAN [Gal et al. 2021] and SP-GAN [Li et al. 2021] for point clouds, which also allows for part-generation and part-mixing. Pertaining coordinate-based networks, COALESCE [Yin et al. 2020] studies stitching of parts through the synthesis of joint connections between them. Nonetheless, COALESCE requires segmentation labels.

Closest approaches. Most of the aforementioned methods do not allow part-level mixing or interpolation between neural implicit shapes. Our method is the first to allow for both types of editing operations seamlessly on neural implicit surfaces. In Table 1, we summarize the properties and applications of notable coordinate based methods. Among these methods, DualSDF [Hao et al. 2020] is the closest to ours. DualSDF learns mirrored coarse and fine representations per shape, using primitives and an implicit signed distance function, respectively. Their interaction goal is different from ours: they perform shape manipulation by minimizing the objective function over transformed primitive attributes. Since some of these attributes are unconstrained, primitives that are not directly manipulated by the user may still be affected by the optimization. Thus, the method cannot guarantee that global shape attributes are maintained during editing (Figure 6). In contrast, our method uses sparse, learned representations that allow for direct editing of local shape parts while remaining faithful to the global shape structure. Since our method is part-aware, it also naturally supports mixing of parts from other shapes, as well as unconditional generation of novel shapes.

Table 1. Neural implicit shapes methods and applications.

Method	Inversion	Generation	Editing	Mixing
DeepSDF [2019]	✓	✓	✗	✗
IM-NET[2019]	✓	✓	✗	✗
OccNet [2019]	✓	✓	✗	✗
LDIF [2020]	✓	✗	✗	✗
COALESCE [2020]	✗	✗	✗	✓
DualSDF [2020]	✓	✓	✓	✗
SPAGHETTI	✓	✓	✓	✓

Fig. 3. Decomposition level. SPAGHETTI represents each learned shape as a GMM (colored blobs) and composes them into an implicit shape (in grey).

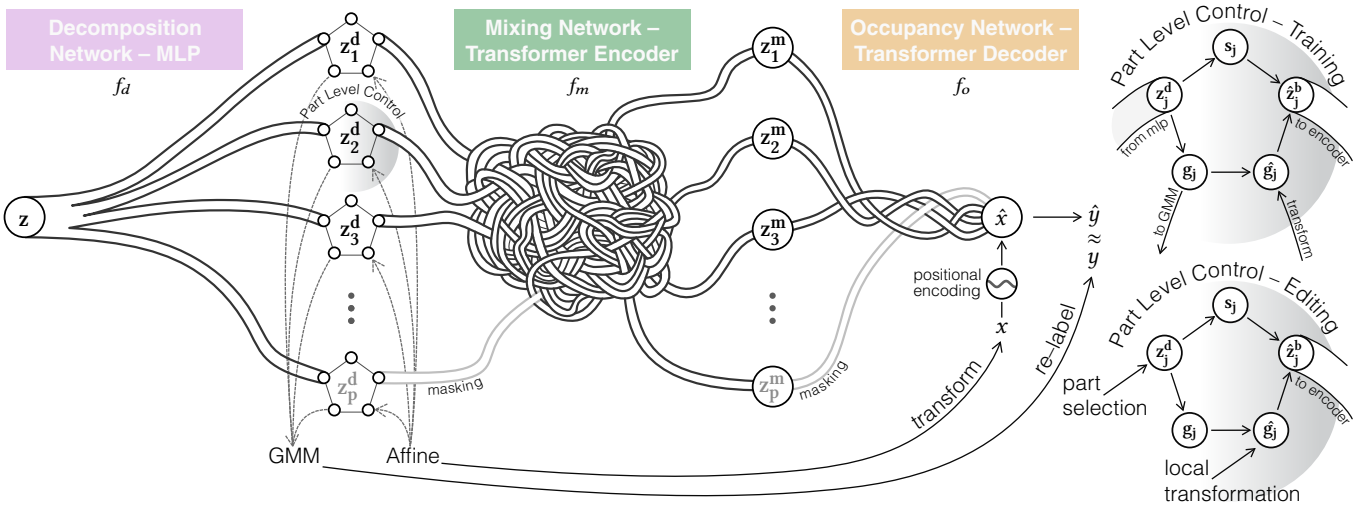
3 METHOD

Our goal is to establish a framework for editing 3D shape parts represented as learned neural implicit fields.

First, as part-level labels are expensive to obtain, we would like to learn the actual part decomposition during training. Specifically, we require the parts we form to be compact but sufficiently descriptive, as over-fragmentation may complicate the interactive editing process, and under-fragmentation may limit the degrees of freedom it allows.

Second, the latent part representations need to be disentangled from each other. This is essential to allow individual part editing and shape mixing. At the same time, the aggregation of parts should form a globally coherent 3D shape.

Third, the mapping between the latent representation and their corresponding occupancy indicators should ideally be equivariant with respect to affine transformations. Such a design allows us to directly manipulate the latent shape representation of parts during



(a) SPAGHETTI's pipeline. We train a *decoder only* network that (i) decomposes a shape embedding $\mathbf{z}$ into distinct parts embeddings $\mathbf{z}_j^d$ that correspond to a Gaussian mixture model (GMM); (ii) processes the (masked) set of $\mathbf{z}_j^d$ into contextual embeddings $\mathbf{z}_j^m$ and (iii) outputs an implicit shape, where a query coordinate $\mathbf{x}$ is projected into a high-dimensional space using positional encoding and is then weighted according to Z^m in order to determine the occupancy indicator $\hat{y}$. The part disentanglement is achieved using the self-supervision, provided by the GMM for re-labeling the ground truth labels y. The local transformation control is achieved by applying rigid transformations on the both the Gaussians and the query coordinates.

(b) Part level controller. Our network is trained (top) to have a disentangled representation of the surface geometry ($\mathbf{s}_j$) and its rigid attributes ($\hat{\mathbf{g}}_j$). During inference (bottom), the user can compose and manipulate the shape parts.

Fig. 4. Method overview. Left: the network architecture. Right: a zoom-in into the part level control component.

editing, while visually mirroring these transformation results in 3D space ad-hoc.

These requirements form the foundation of our architecture. In our formulation, shapes are represented and stored by a single learned global code. Our framework comprises of three parts, as summarized in the high-level overview of Figure 2. Note, in the following sections, $\mathbf{z}, \mathbf{z}^d$ or $\mathbf{z}^m$ denote a high dimensional vector; Z, Z^d or Z^m denote a collection of such vectors.

The first module, the Decomposition Network f_d, maps the shape code $\mathbf{z}$ to a sparse representation of m parts denoted as Z^d. We fix p to roughly agree with the cardinality of distinct shape parts. Each of these representations is projected to the parameters of a 3D Gaussian. There are numerous advantages to this choice. Predominately, it allows us to train shape representation $\mathbf{z}$ as a Gaussian Mixture Model, which naturally forces our architecture to maintain global shape coherency. Likewise, it allows our network to leverage priors about shape categories, as Gaussians are implicitly encouraged to fit parts which are common across all shapes. Most importantly, this decomposition allows for direct control over the shape parts, by applying 3D transformations on the Gaussians associated with them.

The second component, Mixing Network f_m, augments the part representations with contextual information, which further ensures that the local part embeddings remain aware of the global shape structure.

The final building block, Occupancy Network f_o, decodes the contextual embeddings to binary occupancy values, essentially forming our neural implicit representation.

Note that we treat our end-to-end framework as an auto-decoder which jointly trains an implicit shape generative model and embedding vectors $Z = \{\mathbf{z}_i\}_{i=1}^n$ corresponding to n training examples. To simplify the notations, we omit the shape index i for the rest of this section. In practice, we train over datasets of shapes from the same class; see further training details in A.

The full architecture of SPAGHETTI is detailed in Figure 4, and accompanies the in-depth discussions for the rest of this chapter.

3.1 Decomposition Network

Our first objective is to obtain a part-level decomposition of a given shape, which forms the basis of our part based editing. Previous works [Genova et al. 2019a; Chen et al. 2019; Hertz et al. 2020] have shown the ability of 3D generative neural networks to learn a consistent part-level decomposition across generated shapes, without using explicit part-level supervision. Similarly, SPAGHETTI utilizes this ability by conditioning the shape generation on the parts partitioning, and their manipulation.

We formulate the Decomposition Network f_d as a decoder, trained to map input shape code $\mathbf{z}$ into a GMM representation. Given a shape embedding $\mathbf{z}$, we first split it into p distinct vectors $f_d(\mathbf{z}) = Z^d$, where $Z^d \in \mathbb{R}^{p \times d_{\text{model}}}$ is a set of high dimensional parts embeddings $\{\mathbf{z^d}_j\}_{j=1}^p$. The encoding of each part $\mathbf{z^d}_j \in d_{\text{model}}$ is further

projected to two sets of parameters: intrinsic surface geometry information $\mathbf{s}_j \in d_{\text{surf}}$, and extrinsic parameters represented by the Gaussian $\mathbf{g}_j \in \mathbb{R}^{16}$ (see Figure 4b, top):

$$\begin{aligned} \mathbf{s}_j &= W_s \mathbf{z}^d{}_j + \mathbf{b}_s \\ \mathbf{g}_j &= W_d \mathbf{z}^d{}_j + \mathbf{b}_d \end{aligned} \tag{1}$$

Intuitively, $\mathbf{g}_j$ marks the area of influence of each part j, whose detailed structural information is captured by $\mathbf{s}_j$. One of the advantages of this representation is that across the entire dataset, similar intra-category parts are represented using the same Gaussians in a consistent way.

The Decomposition Network f_d, is a multi-layer perceptron (MLP) where after the first fully connected layer, we split the embedding to p vectors and the rest of the layers are shared between the m embeddings. It is followed by the projection of $\mathbf{z}_j^d$ onto the pair $\mathbf{s}_j$ and $\mathbf{g}_j$ (Eq (1)). See Appendix A.2 for further implementation details.

The low dimensional Gaussian $\mathbf{g}_j$ is a stacked representation of the parameters: mixing weight $\pi_j \in \mathbb{R}^1$, center $\boldsymbol{\mu}_j \in \mathbb{R}^3$ and factorized covariance matrix values $U_j \in \mathbb{R}^{3\times3}$, $\boldsymbol{\lambda}_j \in \mathbb{R}^3$. The covariance matrix can be calculated using the eigendecomposition $\Sigma_j = U_j^{-1} D_j U_j$, where D_j is a diagonal matrix with the vector $\boldsymbol{\lambda}_j$ as its diagonal and U_j is a unitary matrix.

Given a batch of points $X_{\text{vol}} \in \mathbb{R}^{B\times3}$ randomly sampled *inside* the full shape that corresponds to the global embedding $\mathbf{z}$, the network f_d is trained by the GMM negative log-likelihood loss:

$$\mathcal{L}_{GMM}(X_{\text{vol}}, GMM) = -\log p\left(X_{\text{vol}}|\text{GMM}\right), \tag{2}$$

where:

$$p\left(X_{\text{vol}}|\text{GMM}\right) = \prod_{\mathbf{x}\in X_{\text{vol}}} \sum_{j=1}^{p} \pi_j \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j\right).$$

The GMM loss encourages the Decomposition Network f_d to dissipate the Gaussians over the entire shape volume, such that every randomly sampled point can be explained by at least one of the Gaussians in the mixture. Figure 3 shows examples for the GMM decomposition learned for various shapes.

3.2 Implicit shape composer

The rest of our network is trained to compose together the set of part embeddings Z^d, to a high resolution implicit shape. Each part representation $\mathbf{z}^d \in Z^d$ from the Decomposition Network is disentangled to extrinsic and intrinsic components, which are then reconstructed back together to form representations $\hat{\mathbf{z}}^d \in \hat{Z}^d$. For brevity, we defer the discussion about attribute disentanglement to Section 3.3, and resume directly from where $\hat{\mathbf{z}}^d$ are fed to the rest of the pipeline (Figure 4b).

Our composition begins by using the Mixing Network f_m to reinforce the representation of each part $\hat{\mathbf{z}}^d$ with contextual information from other parts in $\hat{Z}^d$. The Mixing Network outputs part representations $Z^m = f_m(\hat{Z}^d)$, that are global aware. Then, we use Occupancy Network f_o to obtain the composed implicit function $\hat{y}$. Given a query coordinate $\mathbf{x} \in \mathbb{R}^3$, we calculate $\hat{y} = f_o(\mathbf{x}|Z^m)$, e.g: attend coordinate x on contextual representations Z^m. This yields the part-aware coordinate embedding $\hat{\mathbf{x}}$, which we then proceed to decode for its occupancy value $\hat{y}$.

Both networks are realized through the full Transformer architecture of Vaswani et al. [2017] where Mixing Network f_m is realized through the Transformer encoder, and f_o is a customized variant of the decoder. The Transformer is suitable for our task due to its powerful capabilities of learning contextual representations from sequences or unordered sets in various domains [Radford et al. 2018; Devlin et al. 2018; Lee et al. 2019; Zhao et al. 2021].

The Transfromer encoder of the Mixing Network f_m does not use positional encoding, since we're interested in embeddings of an unordered set. Global aware representations Z^m are obtained through a series of multi head attention layers:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{3}$$

Under the formulation of Vaswani et al. [2017], the queries, keys and values matrices of *each* head h in layer t are given by projecting respectively:

$$\begin{gathered} Q_e = \hat{Z}^d W^{Q_e}; \qquad K_e = \hat{Z}^d W^{K_e}; \qquad V_e = \hat{Z}^d W^{V_e} \\ W^{Q_e} \in \mathbb{R}^{d_{\text{model}}\times d_k}; \quad W^{K_e} \in \mathbb{R}^{d_{\text{model}}\times d_k}; \quad W^{V_e} \in \mathbb{R}^{d_{\text{model}}\times d_v} \end{gathered}$$

with dimensions $d_k = d_v = d_{\text{model}}/h$. The final embedding of each shape part, $\mathbf{z}^m \in d_{\text{model}}$, is obtained by concatenating the per-head outputs and projecting them together.

The Transformer decoder of Occupancy Network f_o uses the output from f_m and coordinates $\mathbf{x} \in \mathbb{R}^3$. During training, coordinates $\mathbf{x}$ are sampled along with their shape occupancy label y around the surface and within a bounding volume $[-1, 1]^3$. We denote each batch of sampled pairs as $X_{\text{surf}} = \left\{(\mathbf{x}_i, y_i)\right\}_{i=1}^{B}$. Appendix A.1 for further elaborates about the sampling scheme and data preparation.

Before feeding coordinates $\mathbf{x}$ to the decoder attention blocks, we first project them onto a high dimensional space $PE(\mathbf{x}) \in \mathbb{R}^{d_{\text{pe}}}$ using a *learned* positional encoding layer. To avoid potential ambiguity, we clarify that positional encodings were previously mentioned in the context of Transformers as means of preserving order in sequences. Similar formulations have been discussed in the literature of Neural Implicit Fields [Tancik et al. 2020] as means of increasing the network sensitivity to coordinate based input and overcoming *Spectral Bias* [Rahaman et al. 2019]. Our formulation refers to the latter definition, pertaining coordinate based networks, and is closer in definition to a single SIREN layer [Sitzmann et al. 2020]:

$$\text{PE}(\mathbf{x}) = \sin\left(a(W_{pe}\mathbf{x} + B_{pe})\right), \tag{4}$$

where $W_{pe} \in \mathbb{R}^{d_{\text{pe}}\times3}$ and $B_{pe} \in \mathbb{R}^{d_{\text{pe}}}$ are learned parameters and a is a fixed scalar. Using a learned variant allows us for an easier initialization which avoids careful tuning due to scale sensitivity issues common in deterministic PE parameterizations. [Hertz et al. 2021].

Network f_o proceeds to calculate the part-aware coordinate embedding $\hat{\mathbf{x}}$ with a sequence of T cross attention layers (Eq. (3)), enumerated as $0 \le t < T$. From a *single coordinate* point of view, layer t in f_o outputs the embedding $\hat{\mathbf{x}}_{t+1}$, calculated by the cross

attention of $\hat{\mathbf{x}}_t$ with Z^m:

$$\begin{gathered}\mathbf{q}_d = \hat{\mathbf{x}}_t W^{Q_d}; \qquad K_d = Z^m W^{K_d}; \qquad V_d = Z^m W^{V_d} \\ W^{Q_d} \in \mathbb{R}^{d_{\text{pe}} \times d_k}; \quad W^{K_d} \in \mathbb{R}^{d_{\text{model}} \times d_k}; \quad W^{V_d} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{pe}}}\end{gathered} \tag{5}$$

We define $\hat{\mathbf{x}}_0 = \text{PE}(\mathbf{x})$, and designate the final output as $\hat{\mathbf{x}} = \hat{\mathbf{x}}_T$. Each attention layer consists of multi-head attention followed by a position-wise feed-forward network.

Unlike the *classic* transformer decoder, we omit the self-attention layers from the decoder. This is essential for a couple of reasons: (i) The occupancy indicator of a coordinate $\mathbf{x} \in X_{\text{surf}}$ should be agnostic to other coordinates we feed in the same set X_{surf} with $\mathbf{x}$, and (ii) We assume the amount of sampled points is considerably larger than the number of parts, e.g: $B >> m$. While it is acceptable to allow quadratic dependency on m, it is desirable to keep the network run-time complexity linear in B.

The last part of f_o is a MLP which decodes $\hat{\mathbf{x}}$ to yield an occupancy indicator $\hat{y}$. The occupancy loss is given by the binary cross entropy loss (*BCE*):

$$\mathcal{L}_{occ}\left(X_{\text{surf}}, \hat{Z}^d\right) = \frac{1}{|X_{\text{surf}}|} \sum_{(x,y) \in X_{\text{surf}}} BCE\left(\hat{y}, y\right), \tag{6}$$

Notice that the contextual-part Z^m is indifferent to the query coordinates. Therefore, in order to reconstruct a $3D$ shape from $\mathbf{z}$, we feed forward the mapping network and the transformer encoder once. Then, the decoder operates in parallel for multiple coordinates, while Z^m remains fixed.

3.3 Disentanglement of extrinsic attributes

We turn to introduce the extrinsic-geometry disentanglement component, which enables local transformation control over the generated shapes (Figure 4b). Recall, we have already retrieved the geometric properties for each part representation, $\mathbf{z}^d{}_j$, by projecting it to the stacked representation of Gaussian $\mathbf{g}_j$ (see Section 3.1). In addition, we obtained the detailed surface information representation $\mathbf{s}_j$.

In the following, we attempt to make embedding $\mathbf{s}_j$, *invariant* to affine transformations applied over the shape part. At the same time, we want the mapping of $\mathbf{g}_j$ to the shape part geometry to be *equivariant* with respect to affine transformations. In other words, transformations applied on $\mathbf{g}_j$ should be directly mapped to the decoded shape part j.

To that end, we apply a random affine transformation T on $\mathbf{g}_j$ to obtain the transformed Gaussian $\hat{\mathbf{g}_j}$. We then up-project and inject it back to $\mathbf{z}^d{}_j$ by:

$$\hat{\mathbf{z}}^d{}_j = \underbrace{\mathbf{s}_j}_{\text{intrinsic}} + \underbrace{W_u \hat{\mathbf{g}_j} + \mathbf{b}_u}_{\text{extrinsic}}. \tag{7}$$

The modified set $\hat{Z^d} = \{\hat{\mathbf{z}}^d{}_j\}_{j=1}^p$ is then routed to the composition networks as described in Section 3.2. Finally, we apply the same transformation T on X_{surf} such that output implicit function $\hat{y}$ learns to mimic the transformed shape.

On one hand, any extrinsic attributes, i.e., part location and orientation, that might be concealed in $\mathbf{s}_j$ are now irrelevant for the reconstruction of $\hat{y}$. This is true since transformation T is applied only on $\mathbf{g}_j$. On the other hand, $\hat{\mathbf{g}_j}$ does not contain any intrinsic surface geometry information by construction. It is extracted from the low dimensional embedding $\mathbf{g}_j$, which holds the parameters of a single Gaussian. Thus, disentanglement of extrinsic and intrinsic geometric attributes is achieved.

3.4 Part-level disentanglement

Ideally, we would like part representations Z^d to contain only local part information, and contextual representations Z^m to be global-aware. The former is required to allow an intuitive part-editing mechanism, where the latter is crucial to have a high-quality reconstruction of the shape $\hat{y}$.

Even though, each part embedding $\mathbf{z}^d{}_j$ corresponds to a single shape Gaussian, there is no guarantee that additional global information will not *leak* between different part embeddings of the same shape. Indeed, such "leaks" may harm the quality of local editing and our ability to mix parts between shapes (see Section 4.4).

To overcome this problem, we augment each training iteration with an additional forward pass, which promotes Z^d to contain local information and better separate the Gaussians area of effect. Different to before, we carefully select a subset of part embeddings, denoted as $Z^{b-} \subset Z^d$. Specifically, we're interested in choosing part representations that may contain mutual knowledge about each other, which is more common with Gaussians that are proximate or potentially overlap. We therefore randomize a direction vector $\mathbf{u} \in \mathbb{R}^3$, and sort all Gaussian centers in that direction. Then, we choose L sequential Gaussians whose part representations constitute Z^{b-}.

From here, we proceed to reconstruct $\hat{y}$ as usual. We expect that the output implicit shape of this forward pass will contain *only* shape parts that are governed by representations Z^{b-}.

Since we do not have direct supervision to guide this optimization, we utilize the clustering induced by the *GMM* of the sampled points X_{surf}, to generate self-supervised labels. We assign each coordinate $\mathbf{x} \in X_{\text{surf}}$ to the Gaussian that maximize its expectation:

$$Cluster\left(\mathbf{x}|GMM\right) = \arg\max_j \pi_j \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j\right),$$

then, we re-label occupancy y as:

$$y^- = \begin{cases} y, & \text{if } Cluster\left(\mathbf{x}|GMM\right) \in \mathbb{I}(Z^{b-}) \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

where $\mathbb{I}(Z^{b-})$ are the indices of the Gaussians in Z^{b-}. Intuitively, we relabel the *outside-coordinates*, e.g: the coordinates that are clustered outside the Gaussians of Z^{b-}, as "non-occupied". The part-level disentanglement loss is then given by an occupancy loss term calculated over the modified set:

$$\mathcal{L}_{dis}(X_{\text{surf}}, Z^{b-}, GMM) = \mathcal{L}_{occ}(X^-_{\text{surf}}, Z^{b-}), \tag{9}$$

where X^-_{surf} is the set of re-labeled pairs (x, y^-).

In practice, for parallel batch training, the selection of subset Z^{b-} is achieved by using attention masking. The attention weights of part embeddings not in Z^{b-} are forced to be zero. This is illustrated by the light gray straws in Figure 4a.

3.5 Training loss function

The complete loss term for our network is given by the terms discussed so far:

$$\mathcal{L}_{SPAGHETTI} = \mathcal{L}_{GMM} + \mathcal{L}_{occ} + \mathcal{L}_{dis} + \gamma\|\mathbf{z}\|_2, \tag{10}$$

where γ is a hyperparameter controlling the loss weight. We also apply regularization $\|\mathbf{z}\|_2$ per global shape embedding, as advocated by previous auto-decoder works [Bojanowski et al. 2018; Park et al. 2019]. The latent regularization promotes the shape codes to be normally distributed. That, in turn, makes the space of global shape codes easier to sample from.

3.6 Shape inversion

Our framework learns to represent shapes through an auto-decoder [Park et al. 2019]. To allow editing of a new shape that is not part of the training data, we first have to match it with a shape embedding $\mathbf{z}$. Our goal is to acquire part embeddings Z^d such that the generated implicit shape is *as close as possible* to the new given shape. We suggest a simple two-steps optimization process to find these embeddings.

In the first step, we begin with a randomly initialized code:

$$\mathbf{z} \sim \mathcal{N}\left(\boldsymbol{\mu}_{tr}, \Sigma_{tr}\right),$$

using the mean and covariance of shape embeddings seen during training. We sample points X_{vol} inside the new given shape, and use f_d with frozen weights to obtain Z^d. Specifically, $\mathbf{z}$ is optimized using the GMM negative log-likelihood (Eq. (2)) between the generated GMM of $\mathbf{z}$ and sampled points, X_{vol}, inside the new given shape.

In the second optimization step, we ensure the obtained Z^d reproduce the shape accurately. Therefore, we freeze the layers of f_m, f_o and use occupancy loss (Eq. (6)) to further optimize the part embeddings Z^d:

$$\arg\max_{Z^d} \mathcal{L}_{occ}(X_{\text{cube}}, Z^m),$$

where Z^m are given by the contextual encoder $Z^m = f_m(Z^d)$.

Upon convergence, we can use Z^d in our interactive interface as described in the following section.

3.7 User interface for interactive shape editing

We demonstrate how a pre-trained SPAGHETTI model can be applied as an interactive editing framework. In this setting, the user can manipulate newly generated shapes, or reference shapes taken from an existing dataset. Examples of some editing operations can be seen in Figures 1, 5, 6, as well as in the supplementary video. Our coarse GMM representation is used for partitioning the generated shapes. The partition provided by the GMMs enables a simple interface for quick selection of shape parts that are highlighted together with the same color-coding.

Users can select parts from different shapes, mix them together, and assemble new shapes. Under the hood, each part selection corresponds to selection of latent vectors from Z^d. These selected latents can then be combined with latent vectors of selected parts from other shapes, to form a single set of latent codes. The combined latents are forwarded through our pretrained network (as shown in Figure 4a) to synthesize newly mixed shapes.

In addition, users can select shape parts and apply affine transformations to them, which results with local deformation of the selected parts. As shown in Figure 4a, the specified transformations are applied on the Gaussian parameters of the selected part. After transformations are applied, the Gaussian representation is injected back to the part latent representation, which goes to the transformer part of our network to synthesize the new manipulated shape.

Finally, we remark on the rasterization pipeline we used for rendering implicit shapes. After each editing step, we reconstruct a mesh with the marching cube algorithm [Lorensen and Cline 1987], using a grid resolution of 256^3. To reduce the number of queries through the network [Takikawa et al. 2021; Hedman et al. 2021], we backed the grid implementation with our in-house implementation of an Octree. The acceleration structure enables an interactive rate of about two seconds between each editing step. Future applications may opt to avoid the mesh conversion step and render the implicit shape directly using ray-marching techniques.

4 EXPERIMENTS

SPAGHETTI can be leveraged for various applications that include shape inversion, generation, editing and mixing. These tasks are summarized in Table 1. In this section we demonstrate the advantages of SPAGHETTI for each application and compare it to the other relevant learning based methods for implicit shapes synthesis. In addition, we conduct an ablation study to evaluate the different components used in our framework.

Our experiments are conducted on the ShapeNet dataset [Chang et al. 2015]. We use the train-test split of DeepSDF [Park et al. 2019], where the number of shapes in the train set, per category, varies from $\sim 1k$ (lamps) to $\sim 5k$ (tables). We train a separate network for each shape category. Further details are included in Appendix A.1.

4.1 Editing comparisons

Concerning neural implicit shapes, only few methods provide editing controls. Moreover, some of the existing methods have set different editing objectives than us. In the following, we discuss the main difference between SPAGHETTI and two contemporary methods which are closest to us: COALESCE [Yin et al. 2020] and DualSDF [Hao et al. 2020]. These works enable shape mixing and shape editing, respectively.

Shape mixing. The objective of COALESCE is to put together a given set of shape parts and output a unified shape. Their pipeline consists of three main steps: (i) A part alignment network, which outputs a transformation for each input part such that the transformed parts are aligned together. (ii) A joint synthesis network that synthesizes new implicit joint parts, which connect the separate parts. (iii) "Poisson mesh stitching" is applied over the parts and the newly formed joints. Each COALESCE network is trained over segmented parts from a single shape category. Figure 5 shows mixing examples of our method compared to the stitching operation of COALESCE, using pre-trained networks on the chairs and airplanes categories. Both methods, receive as an input a set of shape parts. Evidently, synthesizing joints as implicit functions and then stitching them *automatically* to existing meshes is prone to result in noisy artifacts at the joints regions. SPAGHETTI combines the latent parts

Table 2. Shape inversion comparisons. CD = chamfer distance; EMD = earth mover's distance; ACC: mesh accuracy [Seitz et al. 2006]. In all measurements, lower score is better. All measurements are multiplied by a scale of 10^3.

Method	Airplanes			Chairs			Lamps		
	$CD_{mean/med.}$	$EMD_{mean/med.}$	ACC	$CD_{mean/med.}$	$EMD_{mean/med.}$	ACC	$CD_{mean/med.}$	$EMD_{mean/med.}$	ACC
IM-NET [2019]	0.435 / 0.195	28.96 / 25.09	17.541	0.625 / 0.436	33.40 / 30.53	22.062	2.431 / 1.396	73.83 / 59.40	57.313
LDIF [2020]	0.634 / 0.178	31.50 / 20.23	17.237	0.800 / 0.425	29.69 / 25.65	16.787	3.076 / 0.684	56.18 / 42.36	45.478
DeepSDF [2019]	0.095 / 0.029	16.00 / 13.17	6.229	0.323 / 0.113	24.23 / 19.72	14.56	0.792 / 0.205	34.62 / 24.70	15.40
DualSDF [2020]	0.806 / 0.097	31.93 / 22.20	26.78	0.688 / 0.369	34.98 / 32.46	29.59	2.906 / 0.827	75.04 / 50.12	46.69
SPAGHETTI	**0.050** / **0.011**	**9.27** / **7.17**	**4.237**	**0.102** / **0.032**	**13.55** / **11.26**	**6.884**	**0.559** / **0.041**	**17.05** / **12.17**	**6.671**
SPAGHETTI no-enc	0.052 / 0.014	10.84 / 8.66	4.599	0.140 / 0.044	16.40 / 12.95	8.352	0.966 / 0.061	22.20 / 16.41	7.410
SPAGHETTI no-dis	0.068 / 0.015	11.09 / 8.43	5.310	0.135 / 0.039	15.81 / 12.44	7.952	0.854 / 0.083	23.81 / 16.52	11.30

Table 3. Shape generation comparisons. ↑ (↓): higher (lower) is better. MMD-CD scores are multiplied by 10^3; MMD-EMD scores are multiplied by 10^2; JSD scores are multiplied by 10^2.

Method	Airplanes				Chairs				Tables			
	JSD↓	MMD ↓	COV ↑	1-NNA ↓	JSD↓	MMD ↓	COV ↑	1-NNA ↓	JSD↓	MMD ↓	COV ↑	1-NNA ↓
		CD / EMD	CD / EMD	CD / EMD		CD / EMD	CD / EMD	CD / EMD		CD / EMD	CD / EMD	CD / EMD
DeepSDF	3.89	3.8 / 10.2	32.6 / 33.5	70 / 71	1.62	11.1 / 13.6	41.2 / 44.3	60 / 61	1.35	17.3 / 14.8	42.1 / 41.1	59 / 59
IM-NET	3.77	4.2 / 10.5	30.0 / 33.2	65 / 64	2.37	12.44 / 15.1	37.7 / 36.4	61 / 62	3.35	16.6 / 15.7	37.7 / 39.2	61 / 61
DualSDF	6.78	4.2 / 11.1	25.0 / 24.1	70 / 77	4.49	10.4 / 15.9	32.6 / 27.1	70 / 76	2.19	12.3 / 15.2	36.3 / 32.7	68 / 72
SPAGHETTI	**2.28**	**2.4** / **8.10**	**35.0** / **41.3**	**61** / **61**	**1.02**	**6.01** / **11.4**	**50.8** / **51.2**	**58** / **59**	**1.15**	**5.9** / **11.1**	**47.8** / **48.8**	**56** / **56**

Table 4. Disentanglement ablation. Correspondence (Cor.) measures the IoU (%) between distinct parts in our coarser GMM representation to the output implicit shape. Coverage (Cov.) measures the IoU of the whole output implicit shape and the union of its distinct parts. For both, higher score is better.

Method	Tables		Chairs		Lamps	
	Cor.	Cov.	Cor.	Cov.	Cor.	Cov.
SPAGHETTI full	0.779	**0.857**	0.670	**0.765**	0.747	**0.854**
SPAGHETTI no-enc	**0.810**	0.829	**0.673**	0.736	**0.795**	0.781
SPAGHETTI no-dis	0.423	0.527	0.259	0.411	0.455	0.661

Table 5. Mixing ablation. Segmentation (Seg.) measures the Jensen-Shannon divergence (multiplied by 10^2) between segmented parts of the generated mixed shapes to the ground truth input parts. Area measures the surface area error (%) between them. For both, lower score is better.

Method	Tables		Chairs		Lamps	
	Seg.	Area	Seg.	Area	Seg.	Area
SPAGHETTI full	**1.89**	**20.6**	**1.64**	**9.9**	**7.03**	**12.3**
SPAGHETTI no-enc	2.39	23.1	2.06	12.9	7.55	14.0
SPAGHETTI no-dis	3.33	29.2	2.66	17.9	9.92	25.1

to synthesize the entire implicit shape as a single unit, which results in generated shapes appearing more globally coherent.

Shape editing. Given a single latent vector, DualSDF simultaneously synthesizes a high-resolution implicit shape and a corresponding coarse representation, made out of primitive shapes. Then, the user can control the implicit shape by manipulating the primitives of the coarse representation. Each editing step is followed by an optimization of a global latent, which yields a new implicit shape, satisfying constraints set by the coarse form. Figure 6 illustrates the differences in the characteristics of our method and DualSDF [Hao et al. 2020]. The coarse representation of DualSDF does not hold the full geometric properties of the corresponding implicit shape. As a result, after each editing step, the geometry of the shape, or even its topology, might differ from its starting form. Edit operations are not guaranteed to preserve the identity of the original shape, but rather allow a *guided* traversal over the latent space. SPAGHETTI is resilient to such effects as it is trained over an embedding space of disentangled parts. Consequentially, the effect of local changes is bounded, e.g, each editing step affects only a specified portion of the latent representation, therefore, not risking a change of the shape identity.

4.2 Shape inversion

We evaluate the shape inversion quality of our method on 3 shape categories from the Shapenet dataset [Chang et al. 2015]: airplanes, chairs and lamps, using the train-test split of DeepSDF [Park et al. 2019]. We compare our results to other generative and reconstruction methods that output implicit shapes: IM-NET[Chen and Zhang 2019], LDIF [Genova et al. 2019a], DeepSDF [Park et al. 2019] and DualSDF [Hao et al. 2020]. We train a separate network for each

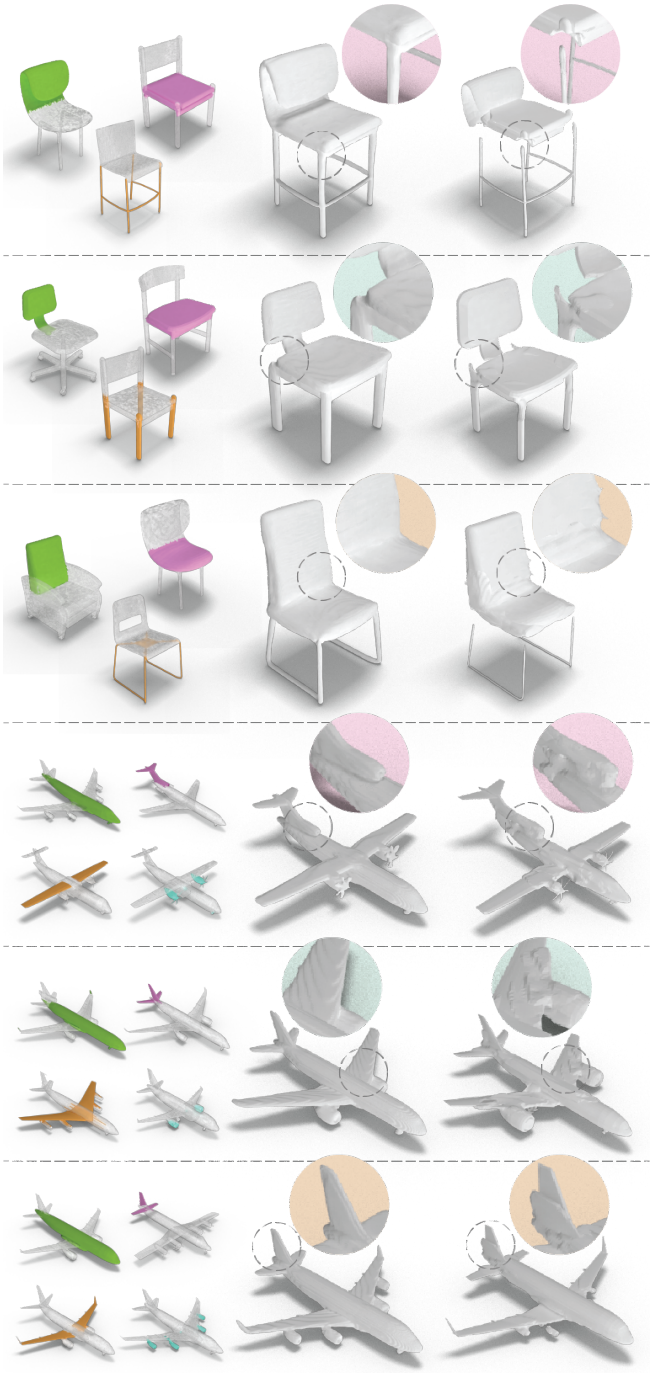


Fig. 5. Mixing comparison. On the left are the input parts supplied to our method and COALESCE [Yin et al. 2020] in order to create a unified novel *mixed* shape. While COALESCE synthesizes only the joints between parts, our method synthesize the whole *mixed* shape.

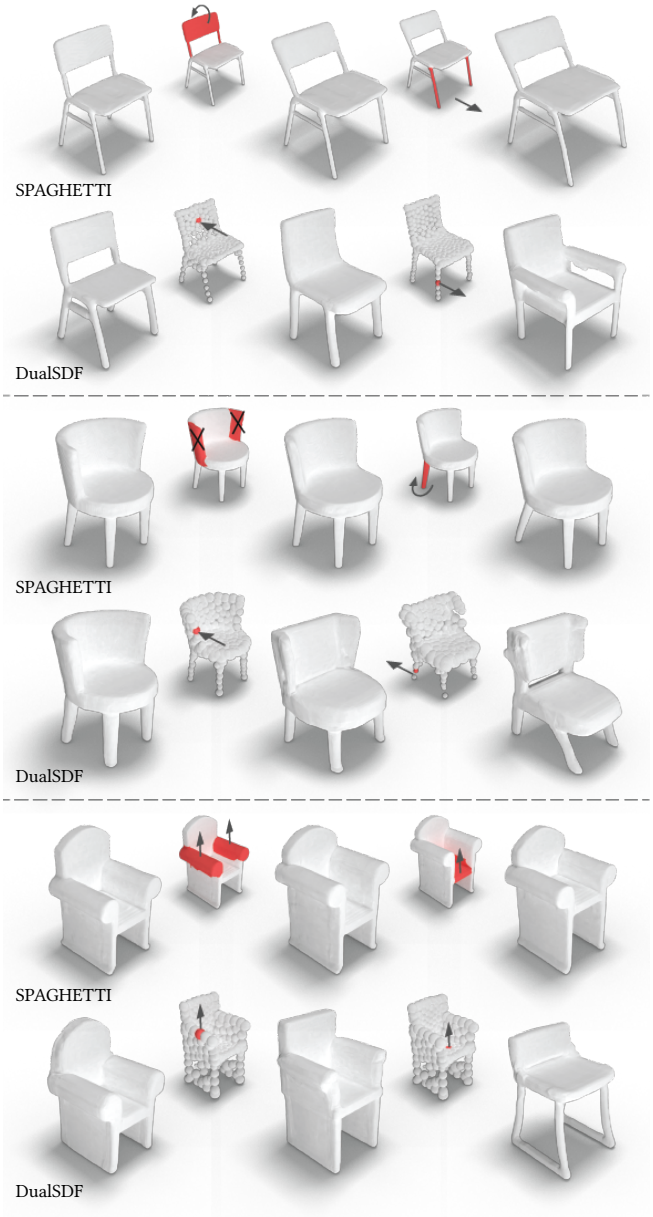


Fig. 6. Sequential editing of implicit shapes using our method (top rows) and DualSDF [Hao et al. 2020] (bottom rows). The input edit (marked in red) provided by the user is shown between the columns.

shape category for each method using their official code and training settings.

The shape inversion process for auto-decoder based methods, DeepSDF and DualSDF, is obtained through an optimization process. They optimize a latent shape vector z, that minimizes the reconstruction loss:

$$\arg\min_{\mathbf{z}} \mathcal{L}_{rec}\left(G(x, \mathbf{z}), y\right),$$

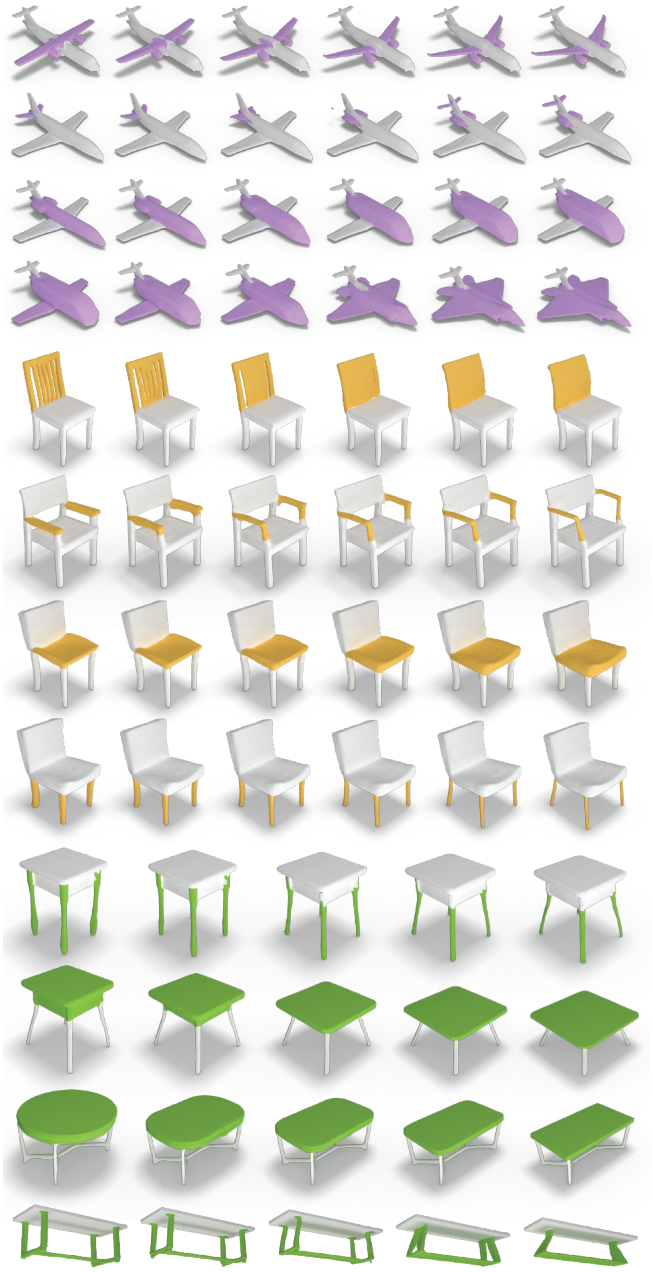


Fig. 7. Part level interpolation. By interpolating the attention weight of f_o, SPAGHETTI can make continuous interpolation between a selected specific part of a shape or any number of parts. In color are the modified parts.

where G is the method's generative model, x are the $3D$ coordinates and y are their corresponding labels, i.e., the signed distances of the SDF representation. L_{rec} is determined by the specific loss settings of each method.

IM-NET and LDIF use encoder-decoder architectures and achieve shape inversion by employing the encoder. IM-NET's Occupancy Network is conditioned on a latent vector that is encoded from a 3D occupancy grid of the input shape. LDIF encodes 24 depth images from different views of the input shape and recovers an implicit shape.

Evaluation metric. Following prior works, the measured distances between the *inverted* implicit shape and the ground truth meshes are chamfer distance (mean and median), earth mover's distance (mean and median) and mesh accuracy [Seitz et al. 2006]. The chamfer distance and earth mover's distance are measured between 30,000 and 1024 sampled points, respectively, on the generated shape and the ground truth mesh. The mesh accuracy value d is the minimal distance such that 90% of 30,000 sampled points on the generated shape are within an Euclidean distance d of the surface of the ground truth shape.

The quantitative results are summarized in Table 2 and qualitative results are shown in Figure 8.

4.3 Shape generation evaluation

In addition to the editing capabilities of our method, we can use a pre-trained SPAGHETTI network for random, unconditional shape generation. Similar to [Bojanowski et al. 2018], we represent the latent distribution as a multivariate Gaussian distribution that best fits the latent space Z of our training data. Then, for shape generation, we feed-forward a sampled vector $\mathbf{z}$ through our network to get its corresponding shape.

Ideally, for fair evaluation, we would like to measure the quality of our generated shapes set A, with respect to the underlying shape distribution of the training data. However, since this distribution is unknown, we can only measure the quality of the generated shapes in A with respect to some empirical distribution represented by an additional shapes set B. In our case, the set B is composed of shapes from the training and test datasets.

In our evaluations, we randomly sampled 2048 points on each shape in A and B. Then, we followed prior 3D generative works [Yang et al. 2019; Gal et al. 2021] and used the metrics introduced by [Achlioptas et al. 2018]:

The Jensen-Shannon Divergence (JSD) measured between the voxel occupancy probability induced by all shapes in A versus all shapes in B.

Coverage (Cov) measured by the percentage of shapes in B that are covered by a shape in A. For this evaluation we assign for each shape in A, its closest shape in B. Then, a shape in B is considered covered if it is assigned by at least one shape in A.

Minimum matching distance (MMD) measured by the average distance of each shape in B and its *closest* shape in A.

1-nearest neighbor accuracy (1-NNA) proposed by [Lopez-Paz and Oquab 2017]. This measurement penalizes each shape s either in A or B whose *closest* shape lays in the same group as s.

We show results with both chamfer distance (CD) and earth mover's distance (EMD) as the distance measures for the COV, MMD and 1-NNA metrics. We randomly generate 1000 shapes to compose the set A for each method and shape category. The set B is composed of randomly selected 500 shapes from the training set and 500 shapes from the test set, repeated per category.

The quantitative results are summarized in Table 3, where we compare our method to 3 other generative methods for implicit

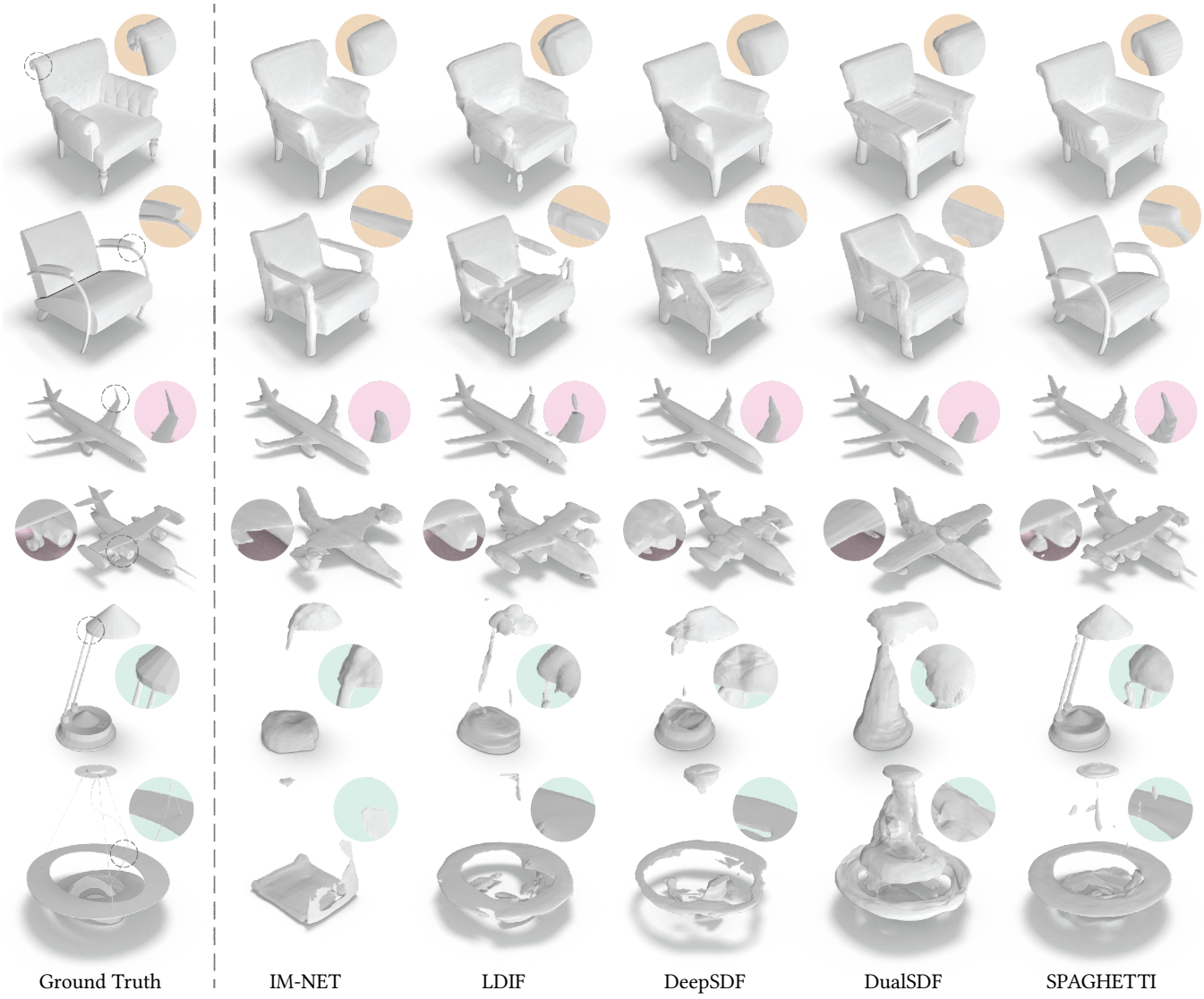


Fig. 8. Shape inversion comparison. Uncurated results of the first two test shapes from the Chairs, Airplanes and Lamps categories of ShapeNet [2015].

shapes: IM-NET[Chen and Zhang 2019], DeepSDF [Park et al. 2019] and DualSDF [Hao et al. 2020]. We repeated our evaluation over 3 shape categories from the Shapenet dataset [Chang et al. 2015]: airplanes, chairs and tables. Additional qualitative results of random generated samples are included in appendix B.

4.4 Ablation studies

To validate our architecture and training settings, we compare our final model to two reduced variants of our method. The first variant, "SPAGHETTI no-enc", omits the middle Mixing Network f_m. For the second variant, "SPAGHETTI no-dis", the network architecture remains the same, but we omit the disentanglement loss $\mathcal{L}_{dis}$ (Eq. (9)) from the training objective.

The results of the ablation study for shape inversion are included in Table 2. Notably, the two reduced SPAGHETTI variants achieve slightly worse inversion results. More importantly, part level control and the quality of editable shapes significantly degrades for these reduced variants. In the following we further discuss these phenomena.

Disentanglement ablation. One of the main objectives of our work is to enforce part-level disentanglement over our mid latent representation Z^d, such that shape manipulations achieved through modifications of latent vector $\mathbf{z}_i^d$, will result only in *local* changes to the output shape. Effectively, $\mathbf{z}_i^d$ should control the shape region that is most likely coming from the corresponding Gaussian g_i.

Fig. 9. Mixing ablation study. Each network configuration receives the selected parts (left) and outputs the mixed implicit shape (right).

We evaluate SPAGHETTI's ability to correspond the extrinsic GMM representation with disjoint, implicit shape parts the network outputs, by conducting two ablation tests.

For each training shape, representations Z^d are obtained by feeding the shape through the Decomposition Network. We then employ the coarse segmentation annotations of PartNet [Mo et al. 2019]. We sample $1e6$ coordinates within the volume of the shape, and assign each of them the segmentation label from PartNet.

We compare every part from PartNet, against the Gaussians obtained from Z^d. Each segmented point is attributed to the Gaussian that maximizes its expectation (Eq. (8)). Then, each Gaussian is mapped to a PartNet label, according to the segmented points attributed to it.

Our setup so far, automatically simulates a user's selection of semantic parts by manually marking their corresponding Gaussians. We partitioned the mid-latent part representations Z^d, and their corresponding GMM to disjoint groups $Z^d = \biguplus_{i=1}^{p} Z_i^{b_-}$, where the Gaussians that correspond to each subset $Z_i^{b_-}$ represent a distinct segmented part p_i among the p parts provided by PartNet.

Using this partitioned latent space, we measure two intersection over union (IoU) scores with respect to the $1e6$ points already sampled inside the shape, and $1e6$ more points uniformly sampled within the unit cube.

First, we measure the latent representations to part correspondence (Cor.) of each part *separately*. For each part p_i, we mask out the representations in Z^d of the Gaussians not associated with $Z_i^{b_-}$. Then, we feed the $2e6$ sampled coordinates through the Occupancy Network f_o, attended over masked part representations, processed by f_m. We specify a label of $y = 1$ for each coordinate $\mathbf{x} \in p_i$, and $y = 0$ otherwise. The IoU is measured by comparing these labels with predictions obtained from the occupancy indicator $f_o\left(\mathbf{x}|f_m\left(Z_i^{b_-}\right)\right)$. Intuitively, a higher IoU score indicates that the latent vectors Z^{b_-} are indeed responsible for the generation of the specific parts associated with their Gaussians.

In the second test we conduct, we measure the coverage IoU (Cov.) between the union of implicit shapes obtained from the distinct parts and the full implicit shape. Here, a higher score means that the union of the separate implicit functions faithfully represents the complete shape, i.e., that:

$$\bigcup_{i=1}^{p} f_o\left(\mathbf{x}|f_m\left(Z_i^{b_-}\right)\right) \approx f_o\left(\mathbf{x}|f_m\left(Z^d\right)\right).$$

We train and evaluate the three network configurations over three ShapeNet categories with part segmentation annotations: tables, chairs and lamps. The results are summarized in Table 4.

Compared to the full SPAGHETTI architecture, the "SPAGHETTI no-dis" variant, trained without the explicit disentanglement loss, yields poor correspondence between the latent part representations and the output shape.

"SPAGHETTI no-enc" variant, which omits the Mixing Network, achieves slightly better IoU scores for local part correspondences (Cor.) but performs worse in terms of global shape coverage (Cov.). We attribute these differences to interactions that occur within the Mixing Network, which augment the contextual representation of each part vector $\mathbf{z}^m$ with global information. Introducing global information requires additional capacity from the part embeddings, which may come at the expanse of local part information. Nevertheless, in practice, we aim to maintain a balance between global coherency and local parts separation. We therefore find that the inclusion of the Mixing Network is crucial for synthesizing distinct parts to a globally coherent shape (Figure 9). In particular, the Mixing Network is an important backbone for the "mixing" edit-operations, e.g: synthesizing together parts from different shapes.

Mixing ablation. We conduct an additional ablation test, where we start with the same segmented part partition as before. Each latent code $\mathbf{z}^d$ and its projected Gaussian are mapped to some part p_i from PartNet, but for each shape, we replace one of the part codes $\mathbf{z}^d$

with a code from another shape. The replaced code comes from the same part category, i.e: we may replace the latent code representing a leg of a chair with code representing a leg of another chair.

A qualitative comparison for this experiment is shown in Figure 9. We observe that the settings that remove the Mixing Network (SPAGHETTI no-enc) or train without $\mathcal{L}_{dis}$ (SPAGHETTI no-dis) are prone to noisy artifacts. At the same time, our full settings preserve the distinct parts better, while generating well-figured mixed shapes.

Since we do not have a ground truth mixed shape to compare to, we are compelled to conduct an indirect quantitative comparison instead. Our ablation aims to give an upper bound for the quality of mixed shapes, by comparing specific attributes of the "mixed" shapes with respect to their input parts.

The first approximation bound we use, measures the difference between the surface area of the mixed shape to the surface area of the input shape. We report the percentage of that difference with respect to the surface area of the input parts. We refer to this measurement as *area evaluation*. Since this evaluation only gives a rough indication for the quality of mixed shapes, we suggest an additional criterion.

The second bound we employ, measures the segmentation quality of the generated shape with respect to the input parts. For this evaluation, we trained a segmentation network [Qi et al. 2017] for each shape category, and used it to estimate the surface area of each segmentation class, per shape. Then, we divide those areas by the total area of the entire shape, to obtain a distribution over the segmentation-classes. We report the Jensen-Shannon divergence of this distribution, with respect to the ground truth segmentation distribution of the input parts. We refer to this measure as *segmentation evaluation*.

Qualitative segmentation examples are shown in Figure 9 and quantitative results are summarized in Table 5 (Seg. and area evaluation).

4.5 Part level interpolation

We now demonstrate the properties of the obtained manifold that contains latent part embeddings Z^d, and evaluate the quality of continuous interpolations between them. Our goal is to demonstrate that this manifold is smooth in the geometric sense.

Our evaluation does not concern global interpolation, where one can simply interpolate between different shape embeddings $\mathbf{z}$ (although our method can be used also for this purpose as well), but instead we focus on interpolations performed between sets of shape parts. Such interpolations are non-trivial, as given two part embeddings sets Z_1^d and Z_2^d, we do not posses correspondences between matching part. Moreover, Z_1^d and Z_2^d might consist of different number of part embeddings. Therefore, we suggest an interpolation scheme throughout the attention weights computed by the cross attention layers of f_o.

First, we compute all contextual part embeddings for two given shapes: $Z_1^m = f_m(Z_1^d)$ and $Z_2^m = f_m(Z_2^d)$. Then, we replace each multi head attention value with an interpolated attention weight.

Recall, the Occupancy Network uses the multi head attention formulation: $\text{Attention}(\mathbf{q}_d, K_d, V_d)$.

Fig. 10. Limitations. Since SPAGHETTI is trained without semantic parts supervision, the learned partitions may over-cluster semantic parts (chairs back) , or limit the resolution of editing by under-clustering parts (chairs leg).

This is the Transformer decoder depicted in Eq. (5), where $\mathbf{q}_d$ is the query vector of $\hat{x_t}$, and K_d, V_d are the key and value matrices for Z^m. For brevity, we omit the decoder notation d in the following description.

Let K_1, K_2 and V_1, V_2 represent the keys and values of Z_1^m and Z_2^m, respectively (Eq. (5)). For a linear interpolation weight $\alpha \in [0, 1]$, the interpolated attention is given by:

$$\text{I-Attention}(\alpha, \mathbf{q}, K_1, V_1, K_2, V_2) = (1-\alpha)\text{Attention}(\mathbf{q}, K_1, V_1) + \alpha\text{Attention}(\mathbf{q}, K_2, V_2).$$

From here, we continue to output the occupancy indication by attending coordinates $\mathbf{x}$ over the interpolated attention value.

Figure 7 illustrates some qualitative results of part level interpolation. During rendering, we highlight the interpolated parts by examining the attention weights of coordinates on the iso-surface, or vertices of a reconstructed mesh. We assign each surface coordinate $\mathbf{x}$ to the part that *receives* the most attention from $\mathbf{x}$:

$$\arg\max_j \sum_{t=0}^{T} \sum_{i=0}^{h} \text{softmax}\left(\frac{\mathbf{q}_{\mathbf{ti}} K_{ti}^T}{\sqrt{d_k}}\right)_j,$$

where aggregation is done over the transformer layers and heads. Finally, we color the vertices that are assigned to the symmetric difference indices $\mathbb{I}\left(Z_1^m \triangle Z_2^m\right)$, that is, coordinates that attend to interpolated codes.

5 CONCLUSIONS, LIMITATIONS AND FUTURE WORK

We presented a generative neural model for implicit 3D shapes, featuring local editing capabilities. Our model is part-aware, requires no part-supervision, and leverages the Transformer architecture to form globally coherent shapes. The network was designed to allow editing at an interactive rate, where, as demonstrated, the user can interact with the generated model using a simple interface.

Our method relies on dual-level disentanglements. First, our model creates disentangled latent codes for disjoint generated shape parts. This allows supporting selection and mixing between different

shape parts. Second, each part embedding is a factorized representation of intrinsic and extrinsic information, which are used to conduct local deformations over the shape.

In our work, we assume to have 3D training data that consists of shapes with similar structure, such as chairs and airplanes. Such structured data enables the unsupervised learning of consistent partitions of compact parts. However, in some shape categories, this assumption holds weakly. For example, the lamps dataset consists of many unique lamps (Figure 3). Moreover, since we do not employ part-level supervision, our model is agnostic to the semantics of part shapes. Our GMM partition may over-cluster together parts into the same Gaussian which may prevent the desired level of control. For example, see Figure 10, where the leg of the chair is represented by only two Gaussians, even though it consists of many sub parts. Similarly, the model may under-cluster large parts and represent them with multiple latent codes, for example, the back of a chair can be represented by more than a single Gaussian. We conjecture that these limitations can be addressed by introducing semi-supervised annotations, or through hierarchical part decomposition [Eckart et al. 2018; Paschalidou et al. 2020a; Hertz et al. 2020] which provides partitioning of the input shape at various scales.

Another direction left for future work is the design of an encoder which predicts the latent codes for a pre-trained SPAGHETTI model, conditioned on different input modalities. For example, when trained on complete 3D shapes, such an encoder may accelerate the inversion process by replacing the optimization steps with a single feed-forward pass. In addition, by training an encoder on 3D scans or 2D images, we may leverage SPAGHETTI's expressiveness for means of 3D reconstruction.

In this work, we explored how learning-based techniques can further assist future workflows of 3D modeling. In a supervised or semi-supervised settings, where the part decomposition is guided by instance-level labels, the performance can be further improved, which may also lead to co-segmentation as a byproduct. In the future, we would like to add more intuitive 3D editing tools, and other types of interactive interfaces. For example, guiding the 3D modeling process by 2D sketches or through textual descriptions.

ACKNOWLEDGMENTS

This work is supported in part by the European Research Council (ERC) Consolidator grant no. 101003104 and Starting grant no. 757497, by the Israel Science Foundation grant no. 2492/20 and by the Innovation Programme grant no. 101003104.

REFERENCES

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. 2018. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*. PMLR, 40–49.

Matan Atzmon and Yaron Lipman. 2020. SAL: Sign Agnostic Learning of Shapes From Raw Data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gavin Barill, Neil Dickson, Ryan Schmidt, David I.W. Levin, and Alec Jacobson. 2018. Fast Winding Numbers for Soups and Clouds. *ACM Transactions on Graphics* (2018).

Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. 2018. Optimizing the Latent Space of Generative Networks. In *International Conference on Machine Learning*. PMLR, 600–609.

Marie-Paule Cani, Takeo Igarashi, and Geoff Wyvill. 2008. *Interactive Shape Design*. Morgan & Claypool Publishers. 78 pages. https://doi.org/10.2200/S00122ED1V01Y200806CGR006

J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. 2001. Reconstruction and Representation of 3D Objects with Radial Basis Functions *(SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 67–76. https://doi.org/10.1145/383259.383266

Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. 2020a. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision*. Springer, 608–625.

Rohan Chabra, Jan E. Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. 2020b. Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 608–625.

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

Siddhartha Chaudhuri, Daniel Ritchie, Kai Xu, and Hao Zhang. 2019. Learning Generative Models of 3D Structures. In *Eurographics*.

Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. 2020. BSP-Net: Generating Compact Meshes via Binary Space Partitioning. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).

Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. 2019. BAE-NET: Branched Autoencoder for Shape Co-Segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

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

Daniel Cohen-Or, Amira Solomovic, and David Levin. 1998. Three-Dimensional Distance Field Metamorphosis. *ACM Trans. Graph.* 17, 2 (apr 1998), 116–141. https://doi.org/10.1145/274363.274366

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516* (2014).

Benjamin Eckart, Kihwan Kim, and Jan Kautz. 2018. Hgmr: Hierarchical gaussian mixtures for adaptive 3d registration. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 705–721.

Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J Mitra, and Michael Wimmer. 2020. Points2surf learning implicit surfaces from point clouds. In *European Conference on Computer Vision*. Springer, 108–124.

Noa Fish*, Melinos Averkiou*, Oliver van Kaick, Olga Sorkine-Hornung, Daniel Cohen-Or, and Niloy J. Mitra. 2014. Meta-representation of Shape Families. *Transactions on Graphics (Special issue of SIGGRAPH 2014)* (2014), 11 pages. * joint first authors.

Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. 2004. Modeling by example. *ACM transactions on graphics (TOG)* 23, 3 (2004), 652–663.

Rinon Gal, Amit Bermano, Hao Zhang, and Daniel Cohen-Or. 2021. MRGAN: Multi-Rooted 3D Shape Generation with Unsupervised Part Disentanglement. In *ICCV Workshop on Structural and Compositional Learning on 3D Data (StruCo3D)*.

Ran Gal, Olga Sorkine, Niloy J Mitra, and Daniel Cohen-Or. 2009. iWIRES: An analyze-and-edit approach to shape manipulation. In *ACM SIGGRAPH 2009 papers*. 1–10.

Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. 2019. SDM-NET: Deep Generative Network for Structured Deformable Mesh. *ACM Trans. Graph.* 38, 6, Article 243 (nov 2019), 15 pages. https://doi.org/10.1145/3355089.3356488

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

Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. 2019a. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7154–7164.

Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas A. Funkhouser. 2019b. Learning Shape Templates With Structured Implicit Functions. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), 7153–7163.

R. Girdhar, D.F. Fouhey, M. Rodriguez, and A. Gupta. 2016. Learning a Predictable and Generative Vector Representation for Objects. In *ECCV*.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (Montreal, Canada) *(NIPS'14)*. MIT Press, Cambridge, MA, USA, 2672–2680.

Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. 2018. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In

Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).

Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge Belongie. 2020. Dualsdf: Semantic shape manipulation using a two-level representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7631–7641.

Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. 2021. Baking Neural Radiance Fields for Real-Time View Synthesis. *ICCV* (2021).

Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. 2020. PointGMM: A Neural GMM Network for Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. 2021. SAPE: Spatially-Adaptive Progressive Encoding for Neural Optimization. In *Thirty-Fifth Conference on Neural Information Processing Systems.*

D.D. Hoffman and W.A. Richards. 1984. Parts of recognition. *Cognition* 18, 1 (1984), 65–96. https://doi.org/10.1016/0010-0277(84)90022-2

Jingwei Huang, Hao Su, and Leonidas Guibas. 2018. Robust Watertight Manifold Surface Generation Method for ShapeNet Models. *arXiv preprint arXiv:1802.01698* (2018).

Qixing Huang, F. Wang, and Leonidas J. Guibas. 2014. Functional map networks for analyzing and exploring large shape collections. *ACM Transactions on Graphics (TOG)* 33 (2014), 1 – 11.

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. https://libigl.github.io/.

Chiyu Jiang, Jingwei Huang, Andrea Tagliasacchi, and Leonidas Guibas. 2020a. ShapeFlow: Learnable Deformations Among 3D Shapes. In *Advances in Neural Information Processing Systems.*

Chiyu Max Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. 2020b. Local Implicit Grid Representations for 3D Scenes. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).*

Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. 2010. Learning 3D Mesh Segmentation and Labeling. *ACM Trans. Graph.* 29, 4, Article 102 (jul 2010), 12 pages. https://doi.org/10.1145/1778765.1778839

Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. 2013. Learning Part-Based Templates from Large Collections of 3D Shapes. 32, 4, Article 70 (jul 2013), 12 pages.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. arXiv:http://arxiv.org/abs/1312.6114v10 [stat.ML]

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*. 3744–3753.

Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. 2018. Point cloud gan. *arXiv preprint arXiv:1810.05795* (2018).

Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. 2017. GRASS: Generative Recursive Autoencoders for Shape Structures. *ACM Trans. Graph.* 36, 4, Article 52 (jul 2017), 14 pages. https://doi.org/10.1145/3072959.3073637

Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. 2019. GRAINS: Generative Recursive Autoencoders for INdoor Scenes. *ACM Trans. Graph.* 38, 2, Article 12 (feb 2019), 16 pages. https://doi.org/10.1145/3303766

Ruihui Li, Xianzhi Li, Ke-Hei Hui, and Chi-Wing Fu. 2021. SP-GAN:Sphere-Guided 3D Shape Generation and Manipulation. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 40, 4 (2021).

Jerry Liu, Fisher Yu, and Thomas Funkhouser. 2017. Interactive 3D Modeling with a Generative Adversarial Network. In *2017 International Conference on 3D Vision (3DV)*. 126–134. https://doi.org/10.1109/3DV.2017.00024

David Lopez-Paz and Maxime Oquab. 2017. Revisiting Classifier Two-Sample Tests. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJkXfE5xx

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

Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. 2021. ACORN: Adaptive Coordinate Networks for Neural Scene Representation. *arXiv preprint arXiv:2105.02788* (2021).

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).*

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Niloy J Mitra, Michael Wand, Hao Zhang, Daniel Cohen-Or, Vladimir Kim, and Qi-Xing Huang. 2014. Structure-aware shape processing. In *ACM SIGGRAPH 2014 Courses*. 1–21.

Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. 2019. PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. 2020. PolyGen: An Autoregressive Generative Model of 3D Meshes. *ICML* (2020).

C. Nash and C. K. I. Williams. 2017. The Shape Variational Autoencoder: A Deep Generative Model of Part-Segmented 3D Objects. *Comput. Graph. Forum* 36, 5 (aug 2017), 1–12. https://doi.org/10.1111/cgf.13240

Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. 2016. Conditional Image Generation with PixelCNN Decoders. In *Proceedings of the 30th International Conference on Neural Information Processing Systems* (Barcelona, Spain) *(NIPS'16)*. Curran Associates Inc., Red Hook, NY, USA, 4797–4805.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 165–174.

Despoina Paschalidou, Luc Van Gool, and Andreas Geiger. 2020a. Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1060–1070.

Despoina Paschalidou, Luc van Gool, and Andreas Geiger. 2020b. Learning Unsupervised Hierarchical Part Decomposition of 3D Objects from a Single RGB Image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).*

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. 2020. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer, 523–540.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2016. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *arXiv preprint arXiv:1612.00593* (2016).

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 5105–5114.

Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1511.06434

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. (2018).

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. 2019. On the Spectral Bias of Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 5301–5310. https://proceedings.mlr.press/v97/rahaman19a.html

Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J. Black. 2018. Generating 3D faces using Convolutional Mesh Autoencoders. In *European Conference on Computer Vision (ECCV)*. 725–741. http://coma.is.tue.mpg.de/

Ryan Schmidt and Brian Wyvill. 2011. ShapeShop: Free-Form 3D Design with Implicit Solid Modeling. In *Sketch-based Interfaces and Modeling*. Springer, 287–312.

Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, Vol. 1. IEEE, 519–528.

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. 2020. Implicit Neural Representations with Periodic Activation Functions. In *Proc. NeurIPS.*

Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11358–11367.

Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shi hong Xia. 2018. Variational Autoencoders for Deforming 3D Mesh Models. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), 5841–5850.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *NeurIPS* (2020).

Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. 2017. Learning Shape Abstractions by Assembling Volumetric Primitives. In *Computer Vision and Pattern Regognition (CVPR)*.

Greg Turk and James F. O'Brien. 2005. Shape Transformation Using Variational Implicit Functions. In *ACM SIGGRAPH 2005 Courses* (Los Angeles, California) *(SIGGRAPH '05)*. Association for Computing Machinery, New York, NY, USA, 13–es. https://doi.org/10.1145/1198555.1198639

Mikaela Angelina Uy, Vladimir G. Kim, Minhyuk Sung, Noam Aigerman, Siddhartha Chaudhuri, and Leonidas Guibas. 2021. Joint Learning of 3D Shape Retrieval and Deformation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. 2019a. 3DN: 3D Deformation Network. In *CVPR*.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019b. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.* 38, 5, Article 146 (oct 2019), 12 pages. https://doi.org/10.1145/3326362

Fangyin Wei, Elena Sizikova, Avneesh Sud, Szymon Rusinkiewicz, and Thomas Funkhouser. 2020. Learning to Infer Semantic Parameters for 3D Shape Editing. In *2020 International Conference on 3D Vision (3DV)*. 434–442. https://doi.org/10.1109/3DV50981.2020.00053

Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. 2016. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2016/file/44f683a84163b3523afe57c2e008bc8c-Paper.pdf

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. 2021. Neural Fields in Visual Computing and Beyond. *arXiv preprint arXiv:2111.11426* (2021).

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

Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. 2018. FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation. 206–215. https://doi.org/10.1109/CVPR.2018.00029

Wang Yifan, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. 2020. Neural Cages for Detail-Preserving 3D Deformations. In *CVPR*.

Kangxue Yin, Zhiqin Chen, Siddhartha Chaudhuri, Matthew Fisher, Vladimir Kim, and Hao Zhang. 2020. COALESCE: Component Assembly by Learning to Synthesize Connections. In *Proc. of 3DV*.

Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. 2020. NeRF++: Analyzing and Improving Neural Radiance Fields. *arXiv:2010.07492* (2020).

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. 2021. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 16259–16268.

A IMPLEMENTATION DETAILS

Our network architecture and training are implemented using Pytorch [Paszke et al. 2019]. Our user interface is implemented using the Visualization Toolkit (VTK).[1]

A.1 Data Preparation.

We use the ShapeNet dataset [Chang et al. 2015] for training and testing our network to produce the results in this paper. Specifically, we use the train-test split of DeepSDF [Park et al. 2019]. The number of training shapes in each category varies from $\sim 1k$ (lamps) to $\sim 5k$ (tables).

[1] http://www.vtk.org.

For each shape we sample points and occupancy labels by the following steps: i) We convert the shape to a watertight mesh using the implementation of Huang et al. [2018]. ii) We scale the shape to a unit sphere. iii) For the GMM loss (X_{vol} in Eq. (2)), we sample 500, 000 points uniformly inside each shape. iv) For the occupancy loss (X_{surf} in Eq. (6)), we randomly sample 500, 000 points inside the bounding cube $[-1, 1]^3$. Additional 500, 000 points are sampled on the surface of the mesh, and are perturbated with Gaussian noise of $\sigma = 0.01$. 500, 000 more surface points are perturbated with $\sigma = 0.02$. In total, we sample 1, 500, 000 points for each shape.

For the occupancy labels we use the *libigl* implementation [Jacobson et al. 2018] of fast winding number [Barill et al. 2018].

Notice that in all the other methods that we compare to [Park et al. 2019; Chen and Zhang 2019; Hao et al. 2020; Genova et al. 2019a], we use the same train/test shapes and use their official implementation[2] for both data preparation and training.

A.2 Network Architecture

Throughout all experiments, our networks were trained using the settings below.

Decomposition Network The dimension of the shape codes Z is 256. Shape codes are initialized randomly from $\mathcal{N}\left(0, 256^{-2}\right)$. Then, as illustrated in Figure 11, the decomposition is similar to PointGMM attentional split layers [Hertz et al. 2020], where, first, a fully connected layer of f_d splits $\mathbf{z}$ into $p = 16$ part vectors of dimension 512 each. Afterwards, we apply 4 multi-head attention layers with $h = 4$ number of heads to output the part embedding $Z^d \in \mathbb{R}^{p \times 512}$.

Implicit Shape Composer. The Mixing Network f_m and Occupancy Network f_o are utilized through the full Transformer architecture [Vaswani et al. 2017]. We removed the self attention layers from the decoder, and used learned positional encoding for the decoder as well. These changes are described in details in Section 3.2.

We use $d_{\text{model}} = 512$ as the dimension of the Mixing Network (Transformer Encoder). For the Occupancy Network (Transformer Decoder), we set the dimension of d_{pe} to 256. The Mixing Network, transformer encoder, has $T = 4$ multi-head attention layers with $h = 4$ number of heads. The Occupancy Network has $T = 6$ multi-head attention layers with $h = 8$ number of heads. To output the occupancy indicator $\hat{y}$, we process $\hat{\mathbf{x}}_T \in \mathbb{R}^{d_{\text{pe}}}$ through another MLP with five hidden layers of size 512.

A.3 Training

Each network was trained to minimize the loss term in Eq. (10) for 2000 epochs with batch size of 18 shapes. Each shape in the batch is represented by 2000 uniformly sampled points inside X_{vol}, and 6000 occupancy points of X_{surf}, where both are randomly selected from the pre-processed points, as described in A.1.

For the disentanglement loss Eq. (9), the subset Z^{b-} is composed of $4 <= L <= 12$ part embeddings as described in Section 3.3, where the value of L is randomly selected in each training iteration.

[2] IM-NET https://github.com/czq142857/IM-NET-pytorch
DeepSDF https://github.com/facebookresearch/DeepSDF
DualSDF https://github.com/zekunhao1995/DualSDF
LDIF https://github.com/google/ldif

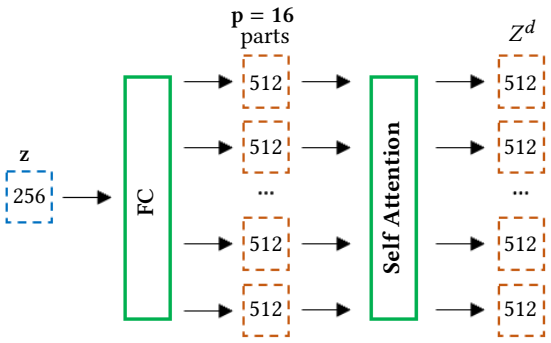


Fig. 11. Architecture of Decomposition Network f_d. Shape embedding $\mathbf{z}$ is projected through a fully connected layer to p intermediate part embeddings of dimension 512 each. Then, the embeddings are forwarded through self attention layers to produce output part embedding $\mathbf{z}^d \in Z^d$ of 512 dimensions.

We set the loss weight in Eq. (10) to $\gamma = 10^{-4}$. We use the Adam optimizer [Kingma and Ba 2014] with a learning rate of 10^{-4} and the default settings ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$). We use 2000 warm-up iterations and apply exponential learning decay of 0.9 in intervals of 500 epochs.

The affine transformations that are applied on X_{surf} and X^-_{surf} (see Section 3.3) are randomly selected from 100,000 pre-computed transformations. Each is composed from a random translation $t \in [-0.3, 0.3]^3$, random uniform scale $s \in [0.7, 1.3]$ and a random uniform rotation from $SO(3)$.

Each model was trained on a single RTX A6000 GPU for 1-5 days, depending on the size of the training data. We run our user interface on a laptop equipped with RTX 3700 GPU.

A.4 Shape Inversion Optimization

For shape inversion, we perform a 2-step optimization, as described in Section 3.6. Both the first and second steps of the optimization are run for a fixed amount of 250 iterations.

On a machine equipped with a RTX 3700 GPU, the processing time of 800 shapes takes 2 hours, where shapes are processed in batches of 20 in parallel.

B RANDOM GENERATION RESULTS

From next page we show qualitative comparisons for the random generation evaluation in Section 4.3. For each method and shape category we show the first 36 generated shapes.

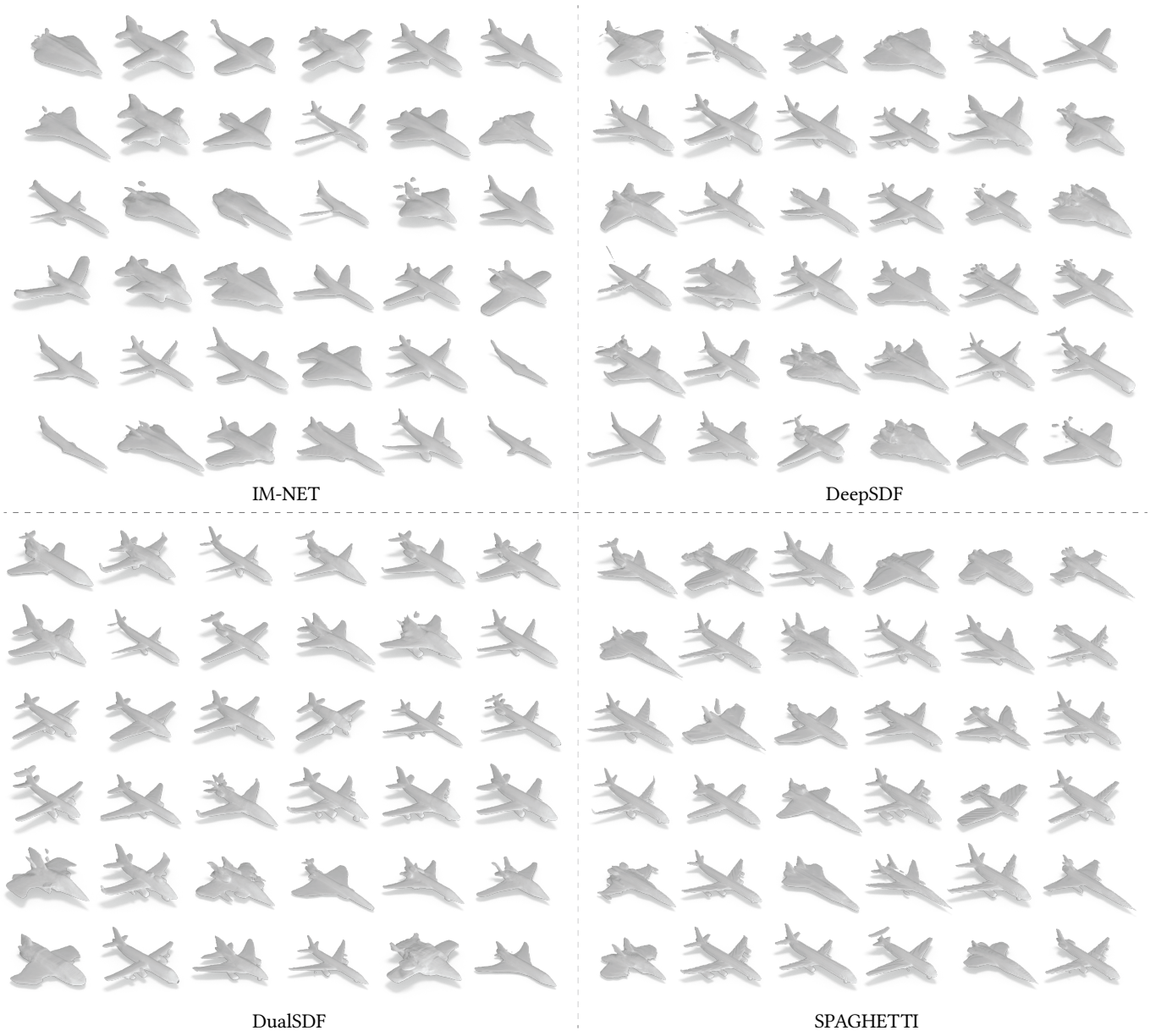


Fig. 12. Shape generation results. First 36 sampled airplanes used for the comparison in Table 3.

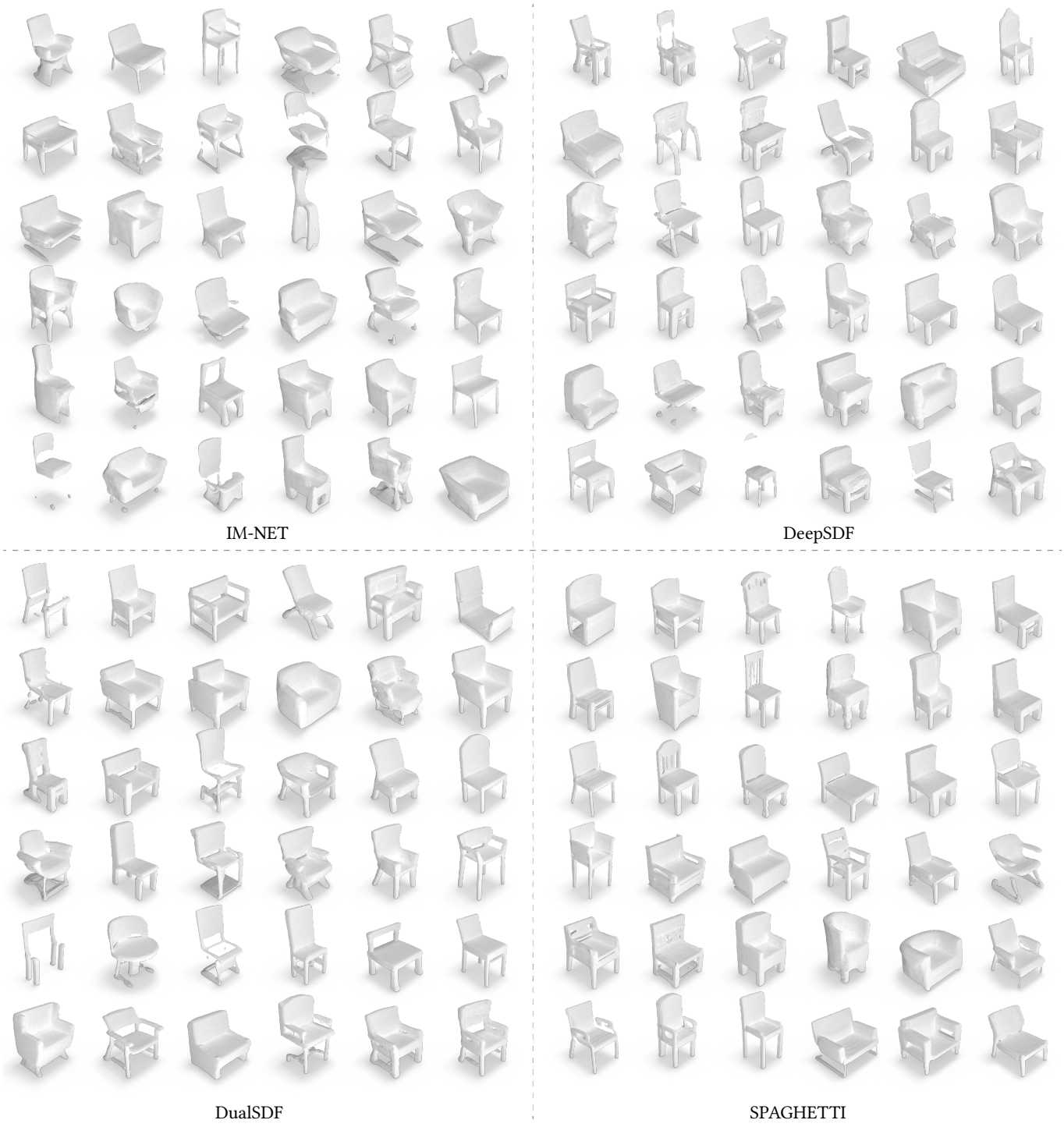


Fig. 13. Shape generation results. First 36 sampled chairs used for the comparison in Table 3.

Fig. 14. Shape generation results. First 36 sampled tables used for the comparison in Table 3.