3D Shape Generation with Grid-based Implicit Functions

Moritz Ibing Isaak Lim Leif Kobbelt
Visual Computing Institute, RWTH Aachen University

Abstract

Previous approaches to generate shapes in a 3D setting train a GAN on the latent space of an autoencoder (AE). Even though this produces convincing results, it has two major shortcomings. As the GAN is limited to reproduce the dataset the AE was trained on, we cannot reuse a trained AE for novel data. Furthermore, it is difficult to add spatial supervision into the generation process, as the AE only gives us a global representation. To remedy these issues, we propose to train the GAN on grids (i.e. each cell covers a part of a shape). In this representation each cell is equipped with a latent vector provided by an AE. This localized representation enables more expressiveness (since the cell-based latent vectors can be combined in novel ways) as well as spatial control of the generation process (e.g. via bounding boxes). Our method outperforms the current state of the art on all established evaluation measures, proposed for quantitatively evaluating the generative capabilities of GANs. We show limitations of these measures and propose the adaptation of a robust criterion from statistical analysis as an alternative.

1. Introduction

Training stable 3D GANs can be challenging and often better results are obtained by splitting the generation process into two parts. First an autoencoder (AE) is trained to obtain a compressed latent representation, then a GAN is trained to model the density of this global latent space. This simplifies the task, as the AE typically generates reasonable output for a wide range of latent vectors. Training the AE can be seen as imposing a bias on the generation process to produce shapes similar to those seen during its training.

This procedure has obtained convincing results both for the generation of point clouds [1] and implicit functions [5]. However, using the latent space of an AE as intermediate shape representation has some severe drawbacks. Small variations of shapes are usually of local nature. As the latent representation used in prior work is of inherently global nature, such changes cannot be modeled easily. This makes it difficult to add any localized modifications or constraints into the generation process. Tasks such as conditional generation based on semantic information (e.g. segmentation/part labels) become hard to train. Another problem is that the AE limits the space of shapes that can be generated. Indeed every AE, which does not represent the entire space of possible shapes in its latent space, limits the expressiveness of the GAN through the bias it imposes. E.g. if we train a GAN to create tables in such a manner, this is unlikely to work with the latent space of an AE trained for chairs.

A more natural choice of representation is to view a shape as a composition of many different local parts. This allows the network to choose parts that occur in different objects and arrange them to create new shapes. Recently, Jiang et al. [13] learned local latent representations by subdividing the object space into a grid to reconstruct scenes from point clouds with high fidelity. This can be seen as only imposing a localized bias on the shape generation process. Learning to capture the geometry within a single grid cell is much simpler than learning to represent the whole shape. Thus the reconstruction quality of the AE is much improved.

We argue that this representation has several further advantages. Training a generative model on this localized latent space enables more variety in the generated shapes, as latent vectors belonging to different shapes or even different object classes can be mixed, giving the generator more degrees of freedom. Its expressiveness is thus not limited by the AEs ability to generalize. In fact we do not even need to train an AE for each specific class. An AE trained on cells of e.g. tables can still be used to generate chairs, as on part level both classes share similarities. Furthermore, we can use convolutional architectures and thus build on existing research in image generation. Although this argument applies to voxel based models as well [34], those are limited by the grid resolution that can fit into memory. As we can represent complex surfaces per grid cell, we can model realistic shapes while still keeping the grid resolution low. Lastly, the spatial decomposition of the latent space allows us to perform conditional generation based on spatial information, such as bounding boxes or semantic labels.

When it comes to evaluating GANs there is no univer-

sally established measure to quantitatively rate, to what degree such networks are able to approximate a data distribution. Image GANs are usually evaluated with the inception score [29] or Fréchet inception distance [11]. The computation of both measures involves the application of the inception network [32]. Since many different representations are used to encode 3D shapes (point clouds, voxel grids, implicit functions, etc.), this is not straightforward to apply for our use case. While previous quality scores for 3D data have been proposed, we show that they have some limitations. Therefore, we propose to apply a statistic measure from a two sample test for multivariate sets [4] to compute the statistical difference between a set of generated objects and a test set of unseen shapes from the data distribution.

Our key contributions are as follows:

- We show that localized grid-based implicit functions are better suited for 3D shape generation with GANs than global implicit functions for three reasons. First, they offer higher quality results, since each cell only has to represent fairly simple geometry. Second, localized implicit functions can be combined together in novel ways and therefore offer more flexibility. Third, localized grid-based implicit functions allow us to control the generation process spatially, which is difficult to accomplish with global implicit functions.

- We show that common evaluation techniques for 3D shape generation have several drawbacks. To alleviate this we propose a new robust score inspired from statistical analysis.

2. Related Work

Many different representations (and network architectures) for processing 3D shapes have been proposed. In the context of this paper we will focus on previous work on the representation of shapes as functions via neural networks as well as prior work on generation of shapes with GANs.

2.1. Representing Shapes as Functions

There are two families of functions that can be used to represent shapes: parametric and implicit functions. The parametric approach to describe a surface (preferably an orientable 2-manifold in 3D) is to define a function $\mathbb{R}^2 \rightarrow \mathbb{R}^3$. Evaluating this function (e.g. on a 2D unit square) then gives positions on the surface of a shape in $\mathbb{R}^3$. Such maps can be learned and represented by neural networks [31]. Since a single map is often insufficient to represent complex shapes, this idea has been modified in AtlasNet [8] to instead learn several functions, where each describes part of a single shape. A different approach is taken by [17] who instead represent a shape with localized functions in a grid, where each function models the surface within a grid cell.

Implicit functions come in two types: signed distance functions $\mathbb{R}^3 \rightarrow \mathbb{R}$ map each point in 3D to a distance value, that is by convention negative inside of the shape and positive outside. The surface of the shape is then the zero level-set of this function. On the other hand there are binary functions $\mathbb{R}^3 \rightarrow \{0, 1\}$, that classify a point as being inside or outside of a given shape. Methods such as Marching Cubes [18] can be used in both cases to extract a surface. Recently several approaches have been published, that apply neural networks to represent such functions [25, 21, 5]. As a global function can have difficulties representing a shape with all its details, several methods have been proposed to mitigate this problem by using localized functions embedded in a grid structure. Chibane et al. [6] employ a multi-scale approach, where feature vectors at different scales and grid points are interpolated based on the current point position, while Jiang et al. [13] save a single feature vector per grid cell and classify a point (in local coordinates) together with its cell vector. We use a similar approach for our generative model.

2.2. GANs in 3D

To the best of our knowledge [34] were the first to develop GANs for the 3D setting, choosing voxels as a representation. This allowed them to use convolutional networks both as the generator and discriminator. However, the results generated by their method are limited by the low resolution of 64^3 grids (mainly due to memory constraints). Achlioptas et al. [1] developed GANs for point clouds in two settings. They trained a generator and discriminator directly on the point clouds but also introduced a two step approach. They first trained an autoencoder for reconstructing point clouds and then fitted a generative model to the latent space. For this they proposed both a GAN and a Gaussian Mixture Model. A similar approach is used by [16]. The idea of splitting GAN training into two steps proved to be applicable to implicit functions as well, as IM-GAN [5] demonstrated. Very recently [15] showed that it is also possible to train GANs directly on implicit functions, however they do not reach the same quality as latent approaches.

3. 3D Shape Generation with Localized Implicit Functions

We propose a two step procedure for the generation of 3D shapes (see Figure 1) similar to [5]. Firstly, we train an autoencoder that maps inputs to localized implicit functions. Secondly, we train a GAN that learns the distribution of such functions. We will discuss our autoencoder first and then describe our proposed unconditional and conditional generation methods. Details of the different architectures and the training procedure can be found in the supplementary material (Appendix B, C).

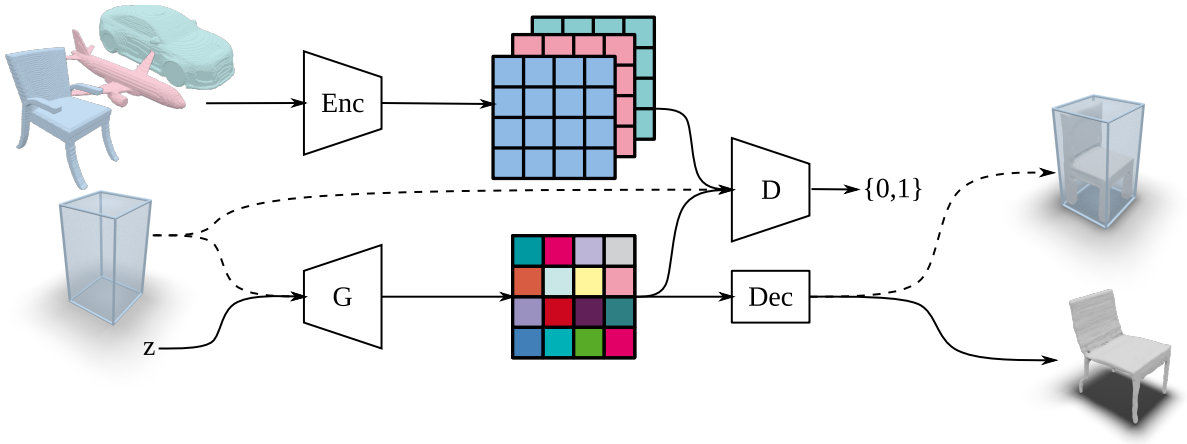

Figure 1: Overview of our method: With the help of an AE (Enc & Dec) we can represent 3D shapes as a grid of latent vectors. Our proposed GAN (G & D) generates new grids of latent vectors, from which shapes can be extracted via the decoder (Dec). Furthermore, we can optionally (dashed arrows) condition, e.g. on bounding boxes, for spatial control in the generation process.

3.1. Autoencoder

The architecture we employ for our autoencoder is related to [13]. Our encoder takes as input high-resolution binary voxel grids (n^3) and after several (strided) convolutions outputs (comparably) low-resolution grids (k^3), where each grid cell is equipped with an h-dimensional latent vector. Thus each cell of the output encodes the geometry of an overlapping cubical subset of the high-resolution voxel grid. In all of our experiments the resolution of the latent grid is chosen as $k = 32$ and the number of channels as $h = 8$.

The decoder evaluates the implicit function represented by this grid of latent cells at a set of sample points. To this end, we find the cell a point is contained in and concatenate its latent vector with the point's position (in local coordinates of the cell). Then we feed this vector into a MLP with a scalar output. The resulting score (0 for outside and 1 for inside of a shape) can be trained with a binary cross entropy loss.

As independent functions in neighboring cells can lead to discontinuities of the implicit function at the boundary between cells, we interpolate the results obtained from neighboring grid cells, leading to smoother results.

$$f(x, C, \theta) = \sum_{j \in \mathcal{N}} w_j \, \mathrm{Dec}_\theta(c_j \in C, k(x - x_j)), \qquad (1)$$

where C is the grid of latent vectors, c_j is the vector corresponding to cell j and x_j is its cell center, $\mathcal{N}$ is the neighborhood of point x and w_j the trilinear interpolation weight of x with regard to the center of cell j. We refer to the resolution of the latent grid as k. The decoder Dec has parameters θ. This interpolation is especially important in our GAN setting, as it simplifies the task of generating matching cells. To make sure that the necessary information for this is stored in the latent vectors, we choose the receptive field of the encoder so that each latent grid cell can encode the geometric information of neighboring cells as well.

3.2. Unconditional Generation

After we have trained an autoencoder, as described in the previous section, we can now train a GAN on its latent space. Chen et al. [5] learn a single latent vector for the entire shape and therefore train simple MLPs as generator and discriminator to generate new vectors and thus new shapes. We on the other hand generate a full grid of latent vectors, each describing a local part of the shape, that together constitute the object. Thus, it is not only important that the local geometry described by the implicit function for each grid cell itself is reasonable, it must match with its neighbors as well. Although these considerations put an additional burden on the GAN, this approach has a decisive advantage over previous latent approaches. As it is possible to arrange the grid cells in various ways, we are less constrained by any bias the autoencoder may introduce. This is because we only need the autoencoder to reconstruct building blocks instead of entire shapes.

Our task can be seen as being analogous to image generation, therefore similar considerations apply. Instead of generating a 2D grid with 3 channels, we generate a 3D grid with h channels. For this reason we use 3D convolutional networks as generator and discriminator. The generator is a simple 3D CNN with strided convolutions, batch normalization and LeakyReLU as non-linearity. We do not use any

residual layers or skip connections.

In order to provide feedback to the generator at various scales, we use three patch discriminators (adapted from pix2pix [12]) that have residual layers and apply spectral normalization [23]. All three discriminators have the same architecture, but do not share parameters. Each discriminator is run convolutionally over the latent grid, rating each local patch separately. The scores are then averaged for the discriminators' final output. The discriminators work at different input resolutions (32^3, 16^3, and 8^3) and thus inspect features of different scale. The lower resolutions are computed by trilinear interpolation of the latent vectors. This training scheme puts a strong focus on local details, emphasizing the higher complexity and expressiveness of single grid cells compared to images or binary voxel grids. The effectiveness of patch discriminators is shown in Appendix D.

For better convergence we use a gradient penalty on interpolations between true and fake data as introduced in [9]. We obtained the best results with zero-centered gradient penalty [28, 20] and maximum reduction [14]. As loss we use the standard non-saturating (NS) loss [7].

3.3. Conditional Generation

In many use cases we would like to control certain characteristics of the generated objects. To this end we can guide the generation process with conditional GANs [22]. These conditions can be of global nature, like class labels, or more localized. An example for this are architectures that create images based on information in the form of other images (e.g. photos from label maps) [12, 33, 26]. For this purpose the input image is used as a mask providing a pixel-wise conditioning.

The spatial organisation of our latent space easily allows us to adapt this approach to 3D data. For our generator we use the architecture of SPADE [26] adapted to 3D. The key idea is to compute a cell-wise scale and bias depending on a mask and apply it to the feature maps at different layers. We use the same discriminators as for unconditional generation. The only difference is that we concatenate the mask with the generated grid in the beginning. We experiment with different ways to provide these masks (e.g. bounding boxes, shape parts, silhouettes), depending on the respective application. The losses and training parameters are the same as for unconditional generation.

4. Evaluation

In this section we present the measure we use to compare the performance of different GANs as well as motivate its effectiveness compared to previous measures. Furthermore, we show quantitative and qualitative results of our unconditional and conditional generation scheme. All evaluations are performed on the ShapeNet Core dataset (v1) [2]. For comparability reasons we follow the training split and evaluation setup from [5] and generate distributions for the categories car, chair, plane, rifle and table. As ground truth we use the voxelized models from [10]. In each category we sorted the models by name and used the first 80% for training and the rest for testing.

4.1. Quality Measures

Several approaches have been suggested to evaluate the fidelity of generative models. Generally any evaluation method needs to answer two questions:

1. How to measure the similarity of individual data points (e.g. 3D shapes)?

2. How to compare two data distributions?

The answer to the first question typically depends on the domain, while the answer to the second question is domain agnostic for the most part. In the realm of 3D shapes, several similarity measures have been proposed. For point cloud based methods the Chamfer distance (CD) and Earth Mover distance (EMD) were introduced [1]. On the other hand [30] proposes to use features extracted from a pretrained PointNet network [27]. As we generate surfaces instead of point clouds, using these methods would require to sample all shapes, by which we lose fine details of the underlying surface. Chen et al. [5] argue that point based distances do not align well with visual similarity and instead propose to use the light field descriptor (LFD) [3] to measure similarity. We follow this reasoning and use the LFD for our evaluation. Note that the similarity between LFDs is measured in a non-Euclidean manner.

Having answered the first question, i.e. how to compare individual data points, we will now discuss how two distributions can be compared. We use the test set as a proxy of a different unseen sampling drawn from the data distribution that generated the training set. We refer to the generated data set as $\mathcal{A}$ and to the test set as $\mathcal{B}$.

Two options for comparing distributions $\mathcal{A}$ and $\mathcal{B}$ were introduced by Achlioptas et al. [1]. *Coverage* is meant to measure the diversity of $\mathcal{A}$ (with regard to $\mathcal{B}$). For each shape in $\mathcal{A}$ we mark the closest neighbor in $\mathcal{B}$ according to the distance defined above. We then count the percentage of shapes in $\mathcal{B}$, that have been marked. If our generative scheme has low diversity, for example due to mode collapse, the generated shapes would lie close together and cover only a small subset of the space spanned by $\mathcal{B}$. This measure however does not evaluate the fidelity of individual shapes within $\mathcal{A}$, as the actual distance between shapes does not matter. To take this into account the *Minimum Matching Distance* (MMD) is introduced. Here we compute the distance from each shape in $\mathcal{B}$ to its closest neighbor in $\mathcal{A}$ and take the mean over those.

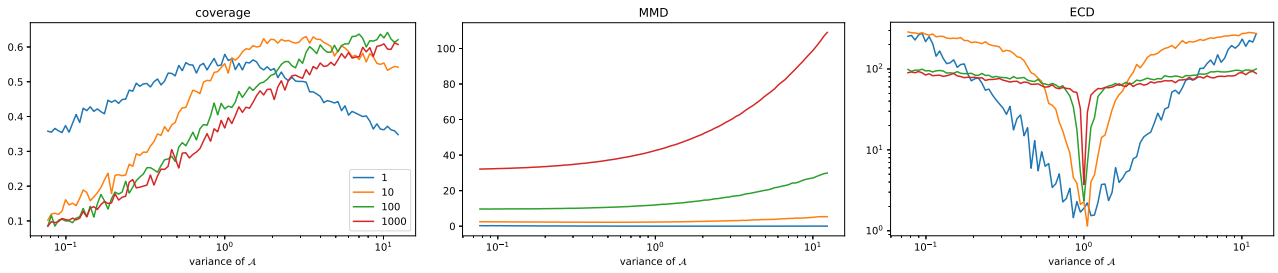

Figure 2: Comparing different distribution measures on two Gaussian distributions with different dimensions (1, 10, 100, 1000) and zero mean. $\mathcal{B}$ has a variance of 1, while the variance of $\mathcal{A}$ varies.

Another option to compare distributions that is popular for evaluating GANs on images, is the Fréchet inception distance (FID) [11]. This measure assumes the distributions to be Gaussian and thus computes mean and covariance of the descriptors within $\mathcal{A}$ and $\mathcal{B}$ and then computes the Fréchet distance. Note that this method assumes Euclidean distance between feature maps and does not work on general distance measures. Therefore, it cannot be applied to the LFD.

We argue that these metrics all have shortcomings in evaluating the quality of a generated distribution. For Coverage and MMD the outliers in $\mathcal{A}$ are not penalized in any way, whereas outliers in $\mathcal{B}$ can have a significant impact. Furthermore, considering only the nearest neighbor is not sufficient to compare the actual density of distributions, as differences in local densities do not have much of an impact. On the other hand FID runs into problems, when the underlying distributions are not Gaussian.

An easy example, where coverage and MMD already fail, is to compare (based on the Euclidean distance) samples from two isotropic Gaussian distributions with zero mean for various dimensions (Figure 2). $\mathcal{B}$ was sampled from a Gaussian with variance 1, while we mimic different generative models by sampling multiple sets of $\mathcal{A}$ from different Gaussians with varying variance. Ideally, the measures would give the best results for $\mathcal{A}$ sampled from a Gaussian with variance 1. However, the coverage never reaches 1 even when $\mathcal{A}$ and $\mathcal{B}$ are drawn from the same distribution. More critically, depending on the dimension, the coverage does not have its peak at 1, as would be expected. The MMD is either not much affected, or has its minimum, when the variance is minimal. Thus, even in combination, the two measure are not reliable in order to determine how likely it is that the two sets come from the same distribution.

To show the deficiencies of the Fréchet distance as an evaluation measure, we compare different types of distributions, whose values we choose so that they all have zero mean and unit variance (Table 1). By definition the Fréchet distance is not able to differentiate between any of these distributions, as the fitted Gaussians are identical.

	Gaussian	uniform	binary
Gaussian	5.31	5.27	5.24
	1.55	52.73	163.58
uniform		5.18	5.16
		1.58	135.21
binary			5.06
			1.95

Table 1: Comparing different measures on different distributions. Fréchet distance in blue, ECD in black. The samples are always 100 dimensional, where each value is chosen from a distribution with parameters chosen so that we have zero mean and unit variance. The binary distribution consists of the values -1 and 1.

We therefore are interested in a test, that remedies these issues. It should take local densities into account, consider the complete distribution of $\mathcal{A}$ and $\mathcal{B}$ without being too much affected by outliers and actually decrease when the two distributions are similar. Furthermore it should be able to distinguish all kinds of distributions and not be restricted to Gaussians. Testing whether two sample sets come from the same underlying distribution is a well known problem in statistical analysis and referred to as a *two sample test*. Since we want to measure the distance between $\mathcal{A}$ and $\mathcal{B}$ by considering the likelihood of them being sampled from the same distribution, this problem is closely related to ours. We propose to use a statistic introduced in [4] as a distance measure between our two sets, since it is fairly robust even for multivariate data. This approach builds a k-minimum spanning tree of the neighborhood graph of $\mathcal{A} \cup \mathcal{B}$. Edges in this tree are classified according to whether they connect shapes within $\mathcal{A}$, within $\mathcal{B}$ or between $\mathcal{A}$ and $\mathcal{B}$. The final score is computed as a weighted difference between the number of these edges and the edge count we would expect if $\mathcal{A}$ and $\mathcal{B}$ were from the same distribution. The exact formula for this measure can be found in Appendix A. Throughout the rest of this paper we will refer to this measure as *Edge Count Difference* (ECD).

		Plane	Car	Chair	Rifle	Table	Avg. w/o planes	Avg.
COV(%)	3DGAN		12.13	25.07	62.32	18.80	29.58	
	PC-GAN	73.55	61.40	70.06	61.47	77.50	67.61	68.80
	IM-GAN	70.33	69.33	75.44	65.26	**86.43**	74.12	73.36
	Our	**81.58**	**80.67**	**82.08**	**81.47**	86.19	**83.10**	**82.80**
	Train	85.04	85.67	84.73	84.00	87.13	85.38	85.13
MMD	3DGAN		1,993	4,365	4,476	5,208	4,010	
	PC-GAN	3,737	1,360	3,143	3,891	2,822	2,804	2,991
	IM-GAN	3,689	1,287	2,893	3,760	2,527	2,617	2,831
	Our	**3,226**	**1,225**	**2,768**	**3,366**	**2,396**	**2,453**	**2,607**
	Train	2,225	984	2,317	3,085	2,066	2,113	2,135

Table 2: Quantitative evaluation of generative models. As 3DGAN was not trained on plane models, this entry is missing. Results for the train set are reported to give reference values.

	Plane	Car	Chair	Rifle	Table
3DGAN		28,855	26,279	6,495	32,116
IM-GAN	6,543	20,606	2,553	3,288	1,018
Our	**355**	**1,062**	**144**	**94**	**188**
Train	1	11	1	2	5

Table 3: Quantitative evaluation of generative models with ECD. We do not report averages, as values for different dataset sizes are not comparable.

As shown in Figure 2 this measure performs reliably on the toy example with Gaussian distributions introduced above. It actually achieves its minimum, when $\mathcal{A}$ and $\mathcal{B}$ are sampled from the same distribution. Furthermore it has no problems distinguishing different distributions with the same variance (Table 1). As we only need to be able to obtain distances between samples to compute the ECD, we are not restricted to the Euclidean space and thus can apply this test together with the LFD.

4.2. Results

To compare our method to the state of the art, we evaluate the results w.r.t. both previous measures Coverage and MMD (Table 2). We compare to 3DGAN [34] (a voxel based method), IM-GAN [5], which is a latent-GAN that works on global implicit functions, as well as PC-GAN [1]. Note that a comparison here is difficult, since this method produces point clouds that need to be transformed to meshes first. For all other methods, meshes are created from generated or sampled voxel grids with marching cubes [18]. The quantitative results for previous methods are taken from [5]. Furthermore, following [5] all measures are evaluated on meshes that have been extracted from a voxelisation with resolution 64^3. Therefore, the different methods have been trained and sampled at resolution 64^3 as well. The size of the generated dataset is always 5 times the size of the test dataset. Note that while IM-GAN and PC-GAN trained specific AEs for each separate class, we trained a single AE on the train split of the entire shapenet dataset. Table 2 shows that our method outperforms the state of the art when it comes to coverage on all but one class, showing the higher diversity of our generated samples. Furthermore, we always have a lower MMD showcasing the higher quality of our sampled shapes.

We also compare against 3DGAN and IM-GAN on the introduced ECD (Table. 3) Here we average the score of 10 random sub-samplings of the generated set with equal number to the test set. Qualitative results of the different methods can be seen in Figure 3. Although the results of IM-GAN sometimes appear smoother than the ones our method produces, this comes at the cost of less diversity.

As we can see in Figure 4 on the example of the rifle dataset, IM-GAN does not fully capture the distribution of the test set, whereas for our method both distributions are well mixed. This is supported by our quantitative evaluation, where we strongly outperform IM-GAN on the ECD even though our qualitative results are comparable.

A reason for the weaker quantitative results of IM-GAN might be that due to problems during GAN training we are observing mode collapse. Another possible explanation for this effect might be found with the strong shape bias the AE enforces, as this bias can make it harder for the GAN to faithfully capture the data distribution. This could be due to the AE mapping larger regions of the latent space to similar shapes. The localized latent grid structure we employ is less prone to such problems, as it is able to represent a much wider variety of shapes.

Conditional Generation As described in Section 3.3, our method is able to incorporate spatial guidance into the generation process. As we make use of the grid structure of our latent space for this, approaches based on a global latent

(a) 3DGAN

(b) IM-GAN

(c) our results

Figure 3: Typical generative results for each model on each category. For 3DGAN we do not show a plane model, since it was not trained on the plane dataset.

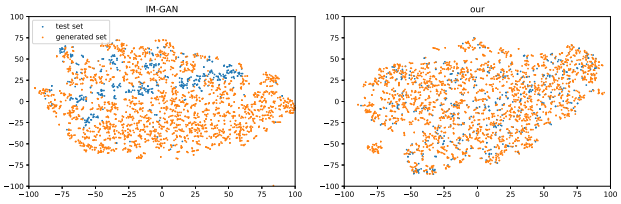

Figure 4: t-SNE [19] (perplexity = 10) low dimensional embeddings of the combined generated and test set from the rifle dataset. On the left distinct clusters are visible, hinting at different densities of the sets. This is not the case on the right.

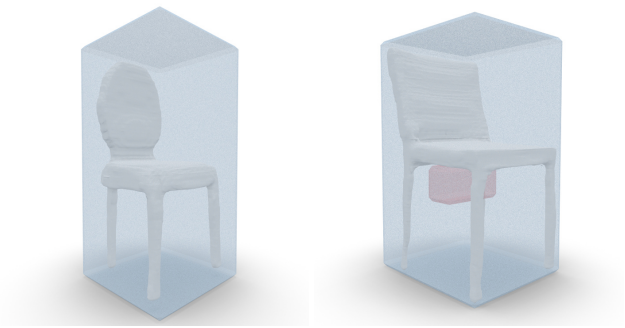

Figure 5: Conditionally generated examples shown with their input bounding boxes. On the right side, we added an additional "negative" box, that is supposed to stay empty.

space are unable to offer such guidance in a straightforward manner. This guidance can be provided in many different ways. This information can take the form of binary values, class labels, or even complex encodings. This flexibility allows for several applications, some of which we will present in the following.

The simplest possibility to offer guidance is to provide a single binary variable for each grid cell. With this we can for example generate shapes that fit into a predefined bounding box. This bounding box is simply discretized to our grid resolution so that each cell gets the information whether it is inside or outside of it. Exemplary results of this generation process, as well as the bounding boxes they were conditioned on, can be seen in Figure 5. We can not only prescribe a bounding box, that should contain all of the geometry, but allow the user to further restrict the shape space, by specifying "negative" bounding boxes, that should not contain any geometry.

If semantic information is available (in our case given by the PartNet dataset [24]), this can be used for more fine grained generation (Figure 6). This means we can prescribe a bounding box for each semantic part of the object. In this

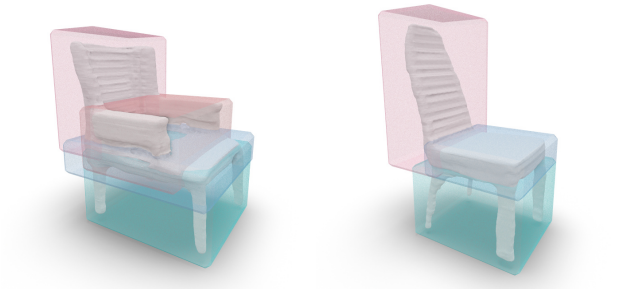

Figure 6: Example for shapes conditioned on part based bounding boxes. The differently colored bounding boxes represent different semantic parts.

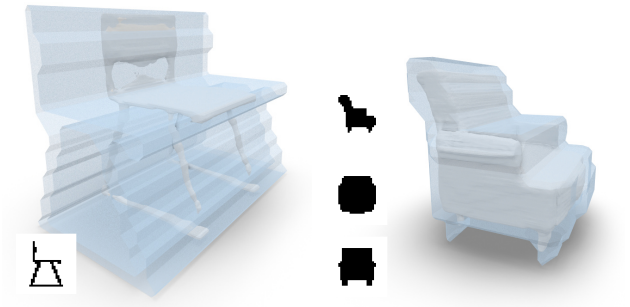

Figure 7: Example for silhouette based generation. The 3D masks are extracted from the low resolution silhouettes shown in the insets.

case, we do not provide binary data, but part labels per grid cell.

Bounding boxes are however not the only option to provide this simple guidance. Another possible application, would be to compute this information from a silhouette, given as a binary 2D image (Figure 7). For simplicity, we assume the viewing direction to be axis aligned. The information can be presented as a single image, or as multiple images corresponding to different axes.

In all previous examples, we provided one-hot vectors per cell. However, complex encodings can be used for this kind of conditioning as well. As an example, we show the application of shape completion on chairs. For this we cut out one semantic part from the object (backrest, armrest, seat or legs) and train the network to insert this missing part. This is implemented by conditioning the GAN on the geometry that is still available. We provide this information by using our AE to encode the present geometry into latent vectors for each cell (Figure 8) As can be seen, our generated shape follows the provided guidance, where it is given, and is reasonably completed, where it is not. For all examples we obtained the supervision masks from unseen examples of the test set.

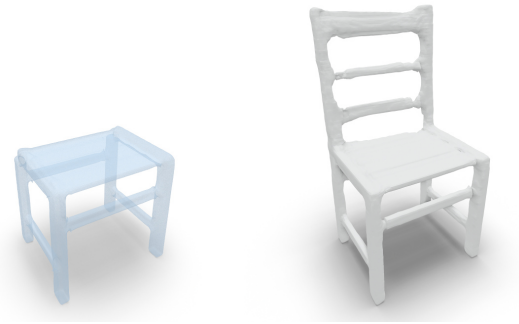

Figure 8: Example for shape completion. On the left we see the given conditioning, on the right the generated chair.

5. Conclusion

We introduced a GAN that generates piecewise implicit functions organized in grids to represent 3D shapes. By learning on localized latent representations instead of global ones (as in previous work) we are able to model the data generating distribution more closely than prior methods. We showed this by evaluating our method quantitatively with measures from prior work as well as with the proposed usage of ECD. Due to the convolutional nature of our GAN architecture we are able to incorporate spatial guidance (in a wide variety of forms) in the generation process. We only showed few examples, but expect this approach to viable for a wide range of tasks.

The results generated from global latent representations appear smoother since they are less likely to capture the details of the ground truth. As the AE part in our generation process is able to learn fine details of the underlying shapes, it tends to reproduce voxelization artifacts from the ground truth. We therefore expect higher resolution ground truth shapes to lead to even better results for our method.

Acknowledgements This work was supported by the Gottfried-Wilhelm-Leibniz Programme of the Deutsche Forschungsgemeinschaft DFG, project number KO2064/6-1, as well as DFG project number KO2064/9-1

References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. J. Guibas. Learning representations and generative models for 3d point clouds. *International Conference on Machine Learning*, 2018.

[2] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[3] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung. On visual similarity based 3d model retrieval. *Computer Graphics Forum*, 22(3):223–232, 2003.

[4] H. Chen and J. H. Friedman. A new graph-based two-sample test for multivariate and object data. *Journal of the American statistical association*, 112(517):397–409, 2017.

[5] Z. Chen and H. Zhang. Learning implicit fields for generative shape modeling. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019.

[6] J. Chibane, T. Alldieck, and G. Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[8] T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[9] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[10] C. Häne, S. Tulsiani, and J. Malik. Hierarchical surface prediction for 3d object reconstruction. *2017 International Conference on 3D Vision (3DV)*, pages 412–420, 2017.

[11] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[13] C. M. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser. Local implicit grid representations for 3d scenes. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2020.

[14] A. Jolicoeur-Martineau and I. Mitliagkas. Connections between support vector machines, wasserstein distance and gradient-penalty gans. *arXiv preprint arXiv:1910.06922*, 2019.

[15] M. Kleineberg, M. Fey, and F. Weichert. Adversarial generation of continuous implicit shape representations. *arXiv preprint arXiv:2002.00349*, 2020.

[16] C. Li, M. Zaheer, Y. Zhang, B. Póczos, and R. Salakhutdinov. Point cloud GAN. *Deep Generative Models for Highly Structured Data, ICLR 2019 Workshop, New Orleans, Louisiana, United States, May 6, 2019*, 2019.

[17] I. Lim, M. Ibing, and L. Kobbelt. A convolutional decoder for point clouds using adaptive instance normalization. *Computer Graphics Forum*, 38(5):99–108, 2019.

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

[19] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[20] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, 2018.

[21] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019.

[22] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[23] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[24] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019.

[25] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019.

[26] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[27] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[28] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann. Stabilizing training of generative adversarial networks through regularization. *Advances in Neural Information Processing Systems*, pages 2018–2028, 2017.

[29] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[30] D. W. Shu, S. W. Park, and J. Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. *Proceedings of the IEEE International Conference on Computer Vision*, pages 3859–3868, 2019.

[31] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani. Surfnet: Generating 3d shape surfaces using deep residual networks. *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 6040–6049, 2017.

[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[33] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.

[34] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in Neural Information Processing Systems*, pages 82–90, 2016.