3D Shape Completion with Test-Time Training

Michael Schopf-Kuester [1] **Zorah Lähner** [1 2 3] **Michael Moeller** [1]

Editors: *S. Vadgama, E.J. Bekkers, A. Pouplin, S.O. Kaba, H. Lawrence, R. Walters, T. Emerson, H. Kvinge, J.M. Tomczak, S. Jegelka*

Abstract

This work addresses the problem of *shape completion*, i.e., the task of restoring incomplete shapes by predicting their missing parts. While previous works have often predicted the fractured and restored shape in one step, we approach the task by separately predicting the fractured and newly restored parts, but ensuring these predictions are interconnected. We use a decoder network motivated by related work on the prediction of signed distance functions (DeepSDF). In particular, our representation allows us to consider *test-time-training*, i.e., finetuning network parameters to match the given incomplete shape more accurately during inference. While previous works often have difficulties with artifacts around the fracture boundary, we demonstrate that our overfitting to the fractured parts leads to significant improvements in the restoration of eight different shape categories of the ShapeNet data set in terms of their chamfer distances.

1. Introduction

Partial objects are very common in shape analysis and pose special challenges when further processing the geometry. The partiality is often due to incomplete views when scanning an object which leaves holes in the reconstructed surface. Many methods exist to complete these kinds of surfaces, both traditional and learning-based methods (Sharf et al., 2004; Anguelov et al., 2005; Wu et al., 2015). Another setting is the completion of partial volumetric shapes. In contrast to holes in surfaces, the boundaries of partiality are not obvious in this case and learned information about the space of shapes is crucial for obtaining good results. While such shapes can result from volumetric reconstruction algorithms (Slavcheva et al., 2017), we focus on the case of *shape restoration* in this paper. The assumption is that a broken object was scanned and the parts needed to complete it should be reconstructed (see Figure 1). In addition to learning a distribution of complete shapes, it is necessary to reconstruct shapes that tightly align with the given input shape, for example to 3D print a fitting replacement part. This is, for example, important in medical applications from partial organ scans (Gafencu et al., 2024) or for 3D printing parts for a broken object (Lamb et al., 2022a), e.g. for cultural heritage applications.

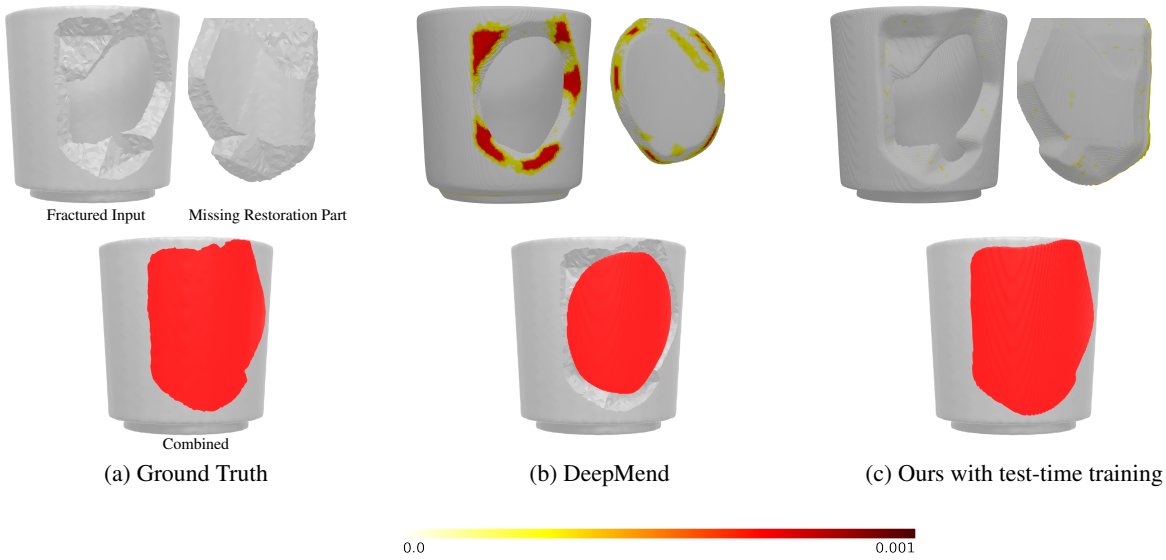
General shape completion methods focus on the appearance of the full reconstruction, which often looks good, but struggle with precision w.r.t. the input shape and details (Park et al., 2019). We build our work upon (Lamb et al., 2022a) which tackles this problem and learns to complete fractured shapes in a certain class by separating the fractured part from the restoration part. However, (Lamb et al., 2022a) still struggles to accurately align details, likely due to the complexity of learned classes whose details are hard to capture in a joint latent space. To overcome this issue we propose carefully designing the latent space and implementing test-time training to take the given geometric information of the input space into account directly. This allows us to produce restoration shapes that accurately represent the fractured area and produce a consistent joint shape, see Figure 1.

Contributions. We make the following contributions:

- A new pipeline for generating accurate restoration shapes in the setting of volumetric shape completion.
- An analysis and optimization of the network architecture for shape restoration proposed in (Lamb et al., 2022a).
- The introduction of test-time training for shape restoration, which requires precise alignment and greatly benefits from this approach.

[1]University of Siegen, Germany [2]University of Bonn, Germany [3]Lamarr Institute, Germany. Correspondence to: Michael Schopf-Kuester <michael.schopf@uni-siegen.de>.

Proceedings of the Geometry-grounded Representation Learning and Generative Modeling at the 41st International Conference on Machine Learning, Vienna, Austria. PMLR Vol Number, 2024.



(a) Ground Truth (b) DeepMend (c) Ours with test-time training

Figure 1. Overview about our method. While DeepMend gets a good rough estimation for the fractured (upper left) and restoration shape (upper right), we get via test-time training much sharper and more detailed shapes, especially w.r.t. the break surface. This results in restoration shapes that fit much better to the original input fractured shape (bottom). In particular, 3D printing as an application case will benefit from our approach.

- Code for Test-Time Training can be found here: https://github.com/mschopfkuester/shape_completion_ttt
- Experiments on several datasets showing the advantages of the introduced changes.

2. Notation and Problem Description

In this section, we introduce the notation used throughout the paper and formalize the problem of shape restoration.

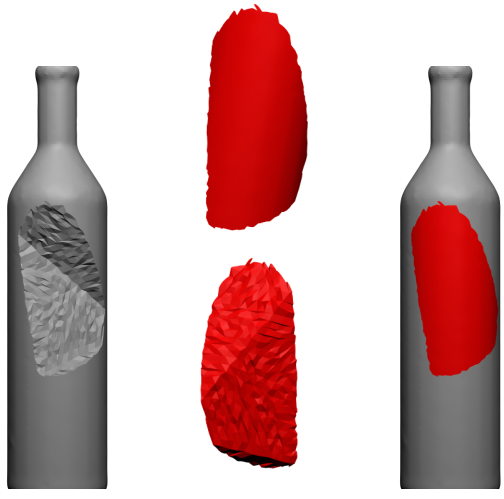
We assume that a *fractured shape* $F \subset \mathbb{R}^3$ is given as input which is a partial version of a *complete shape* $C \subset \mathbb{R}^3$. Therefore, it holds $F \subset C$. Our goal is to find the (with F disjoint) *restoration shape* $R \subset \mathbb{R}^3$ which completes F when merged together, i.e. $C = F \dot{\cup} R$. See Figure 2 for a visualization.

We do not predict the complete shape directly, but the fractured and restoration shape separately. This ensures we can learn structural details about the relationship between the fractured and restoration part. We need a few considerations to reformulate the problem accordingly. Therefore we introduce a so called *break set* $B \subset \mathbb{R}^3$ such that both, the fractured shape F and restoration shape R, can be described as the intersection of the predicted complete shape and the break set (or its complement), i.e.

$$F = C \cap B \text{ and } R = C \cap B^C. \tag{1}$$

This allows to model F and R separately (and pairwise disjunctive) from each other but also guarantees an (inner) relationship between both shapes.

We describe the surface of all shapes by occupancy functions $o_S(x) : \mathbb{R}^3 \to \{0, 1\}, S \in \{F, R, C, B\}$, such that for a



(a) Fractured (b) Restoration (c) Complete

Figure 2. Visualization of the problem setting: Given a fractured shape F (a), we want to predict the missing restoration shape R (b) such that we get the complete shape $C = F \cup R$ (c).

certain point $x \in \mathbb{R}^3$ it holds that

$$o_S(x) = \begin{cases} 1, & x \text{ is inside the shape } S, \\ 0, & \text{other.} \end{cases} \tag{2}$$

In the binary formulation, o_F and o_R can be easily rewritten in the following from o_C and o_B:

$$o_F(x) = o_C(x) \cdot o_B(x), \tag{3}$$
$$o_R(x) = o_C(x) \cdot (1 - o_B(x)). \tag{4}$$

This allows an efficient on-the-fly conversion between all entities.

3. Related Work

In this section, we provide an overview of general 3D generative models as well as shape completion approaches.

3.1. 3D Generative Models

Generating 3D geometry has been a widely studied field (Dai et al., 2017; Luo & Hu, 2021; Qiu et al., 2023) which has greatly benefited from the advances in implicit representations due to their flexibility (Park et al., 2019; Erkoç et al., 2023). These methods are fundamentally different from mesh-based methods like deformation models (Loiseau et al., 2021) or parametric shape models (Zuffi et al., 2018) which rely on a fixed template or topology and, thus, are limited in the shapes they can represent. Combining both can lead to great results but might suffer from inconsistencies between them (Poursaeed et al., 2020; Mehta et al., 2022). Generative models based on neural fields can handle different classes and be conditioned with arbitrary modalities, for example, images (Gao et al., 2022; Liu et al., 2023), text (Qiu et al., 2023; Lin et al., 2023), or latent code manipulation (Park et al., 2019; Zeng et al., 2022; Hu et al., 2024). A special case of generative modeling is *shape completion*, which takes an incomplete or broken shape as input and aims to generate the full object.

3.2. Shape Completion

The partial input can either be an incomplete point cloud or a partial volumetric shape for which semantic information is necessary for completion. A partial point cloud or mesh has identifiable holes which can be closed without any information about the object class, for example by Poisson reconstruction which generates water-tight surfaces out of any oriented point cloud (Kazhdan et al., 2006), by filling the hole with structurally fitting patches (Hanocka et al., 2020) or using learned class features (Chibane et al., 2020). Learning class-based priors for point cloud completion allowing to complete more severe degradation is also possible (Sun et al., 2022; Zhu et al., 2023; Cui et al., 2024).

In a volumetric representation, the boundaries of holes are harder to identify, and semantic information about the properties of the full shapes in this class is necessary. One of the first works in this direction applied convolutions on voxel grids in a coarse-to-fine manner to complete incomplete depth fusions (Dai et al., 2017). However, voxel methods require a lot of memory to represent fine details. With (Sellán et al., 2022) and (Lamb et al., 2023) two datasets with complex fractured shapes were published recently. A huge step towards more efficient detail generation was made in DeepSDF (Park et al., 2019) which learns to predict a signed distance function (SDF) with a neural network. DeepSDF can do both, sample new instances from a class as well as complete partial shapes. However, a strong class prior learned by the network can lead to semantically meaningful but not well-aligned completed shapes.

In applications where a tight fit to the input is necessary, for example, 3D printing replacement parts of broken objects (*shape restoration*), this is a problem. This can be avoided by requiring additional information, like an image of the complete shape (Galvis et al., 2024). Another solution was proposed by DeepMend (Lamb et al., 2022a) and DeepJoin (Lamb et al., 2022b) in explicitly modeling the fractured region and the fit of the generated part to the fracture. Due to the more general setup, where only the occupancy of the shapes is needed, we focus on (Lamb et al., 2022a). We provide a more detailed description of DeepMend in Section 4.1. But like many generative methods, DeepMend suffers from over-smoothing behavior from having a single network for many objects, as we can see in our numerical experiments. To that end, we propose using test-time training for shape restoration which can adjust the network weights to the input and ensure that a tight fit is possible.

3.3. Test-Time Training

The power of learning methods normally comes from huge generalization capabilities based on the training data. However, examples that stray from the training distribution might only be captured sufficiently but not perfectly – often seen in slightly too smooth test results. Finetuning the network (or parts of it) during interference can prevent this in applications, where it is possible to do self-supervised adaption and where inference time is not critical. This so-called test-time training was proposed in (Sun et al., 2020) for generalizing under distribution shifts and has been applied in different applications like robust classification (Gandelsman et al., 2022) and sketch-based image retrieval (Sain et al., 2022). To the best of our knowledge, this work is the first to use test-time training to improve the geometric consistency in learning-based shape completion approaches by overfitting the network weights to match the (fractured) input shape.

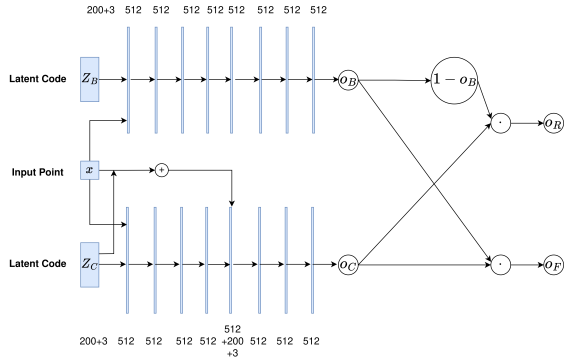


Figure 3. Network architecture of (Lamb et al., 2022a). The architecture is separated into two parts which predict the occupancy of the complete shape o_C and the break set o_B respectively. The input for both parts is the point coordinate $x \in \mathbb{R}^3$ and a latent code describing the geometry. Via Equation (3) and Equation (4) we can calculate the occupancies of the fractured shape o_F and the restoration shape o_R. We define the skip connection with $\oplus$ and the multiplication of the outputs of the two networks with $\odot$.

4. Background

In this section, we introduce the most relevant previous work, DeepMend (Lamb et al., 2022a), which tackles the shape restoration problem and on whose network architecture our work is built, in more detail.

4.1. DeepMend

DeepMend (Lamb et al., 2022a) learns to predict the occupancy functions o_C and o_B, as defined in Section 2, via neural networks and thus learns a representation for the relationship between the fractured and restoration shape that generalizes well to new class instances. The network architecture is an auto-decoder based on DeepSDF (Park et al., 2019) which has as input parameters a latent code and a point $x \in \mathbb{R}^3$ and predicts the corresponding occupancy values $o_C(x)$ and $o_B(x)$. The latent code is optimized directly on every instance without a trained encoder with a decoder-only framework, see DeepSDF for details.

To predict the occupancy of the complete shape $o_C(x)$, (Lamb et al., 2022a) uses a network f_{θ_1} consisting of a 8-layer MLP with a skip connection after the fourth layer and a 128-dimensional latent code z_C as input. A similar network g_{θ_2} is used to predict the break set but without a skip connection and a smaller latent code z_B as input. The exact architectures can be found in (Lamb et al., 2022a). See also Figure 3 for an overview.

Training and Inference For model training (Lamb et al., 2022a) considers the binary cross-entropy loss (BCE) $\mathcal{L}_S$ between the true target $\hat{S}$ and the corresponding prediction S for each of the four sets F, R, C, B:

$$\mathcal{L}_S = \sum_x BCE(o_S(x, z_B, z_C, \theta_1, \theta_2), o_{\hat{S}}(x)). \quad (5)$$

The specific terms can be derived from Equation (3) and Equation (4). Each object class is trained separately and until the chamfer distance error on the validation set is minimal. Dropout is used on all hidden layers. During training, the network parameters θ_1 and θ_2 are jointly optimized with the instance-specific latent codes z_B and z_C:

$$\begin{aligned}\{\hat{z}_B^j\}, \{\hat{z}_C^j\}, \hat{\theta}_1, \hat{\theta}_2 &= \underset{\{z_B^j\}, \{z_C^j\}, \theta_1, \theta_2}{\operatorname{argmin}} \mathcal{L}_{\text{train}} \\ &= \underset{\{z_B^j\}, \{z_C^j\}, \theta_1, \theta_2}{\operatorname{argmin}} \mathcal{L}_C + \mathcal{L}_B + \mathcal{L}_F + \mathcal{L}_R, \end{aligned} \quad (6)$$

where $\mathcal{L}_C, \mathcal{L}_B, \mathcal{L}_F, \mathcal{L}_R$ are the respective binary cross-entropy losses for the complete shape, break set, fractured, and reconstruction shape for the training examples $j = 1, \ldots, n$. The losses are taken from Equation (5). During inference, only the latent codes are optimized and the fractured shape is given as input, such that we calculate

$$\hat{z}_B, \hat{z}_C = \underset{z_B, z_C}{\operatorname{argmin}} \mathcal{L}_F + \mathcal{L}_{reg} \quad (7)$$

for each test object separately. Here $\mathcal{L}_{reg}$ is a combination of different penalty terms to ensure that the predicted restoration shape is not empty and near the fractured shape. The final shape is then inferred as $f_{\theta_1}(\hat{z}_C)$. For more information see (Lamb et al., 2022a).

5. 3D Shape Restoration

In this section, we introduce our method to complete fractured shapes based on test-time training. We build upon the fact that while predicting the rough geometry of the whole shape is important, it is equally important to fit the prediction *exactly* to the partial input to not create artifacts around the break points and, for example, allow a properly fitting spare part to be printed, see Figure 1. To that end, we apply test-time training on the network weights to allow a tight fit to the input geometry, see Section 5.1. Additionally, we perform an analysis of the network architecture and loss functions used in (Lamb et al., 2022a) to increase the performance, see Section 5.2.

5.1. Test-Time Training

During test-time training, the model's weights are finetuned using input data during inference (Sun et al., 2020). Models trained on complex classes often overlook finer details, resulting in smoothed edges and missing details, as seen in (Lamb et al., 2022a). By incorporating the detailed geometric information from the input fractured shape into

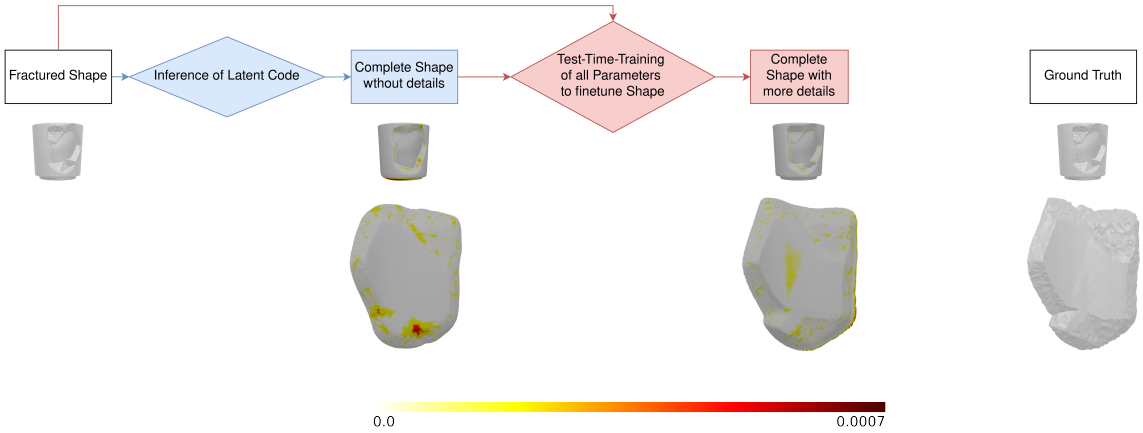


Figure 4. Pipeline of our method with test-time training. After only optimizing the latent code to get a rough prediction of the restoration shape (blue, this is the pipeline of DeepMend (Lamb et al., 2022a)), we use the predicted complete shape as well as the input fractured shape to finetune all network parameters and therefore get a more detailed restoration shape (red, our addition).

test-time training, we achieve better geometric consistency, as demonstrated in our experiments.

Since the full shape is unknown during inference, the training procedure must be adapted to a self-supervised loss function. To that end, we predict the complete shape C using the normally trained network, conditioned on the input fractured shape $\hat{F}$. Then, we get the corresponding restoration shape $\hat{R} := C \setminus \hat{F}$ and use $\hat{F}$ as well as $\hat{R}$ to finetune the network with two loss functions, preserving their geometry:

$$\begin{aligned} \hat{z}_B, \hat{z}_C, \hat{\theta}_1, \hat{\theta}_2 &= \underset{z_B, z_C, \theta_1, \theta_2}{\operatorname{argmin}} \mathcal{L}_{\text{TTT}} \\ &= \underset{z_B, z_C, \theta_1, \theta_2}{\operatorname{argmin}} \mathcal{L}_F + \alpha \cdot \mathcal{L}_R. \end{aligned} \tag{8}$$

In L_F the geometry of the input shape is aligned and incentives the network to tightly fit to the known details. To prevent complete overfitting onto $\hat{F}$ and an empty restoration shape, the class information from the pretrained network is preserved implicitly in $\mathcal{L}_R$. We choose $\alpha = 0.1$ in all experiments. The network is trained for 3000 epochs during test-time training.

5.2. Network Architecture

In addition to the test-time training, we conducted an analysis of the network architecture used in (Lamb et al., 2022a) and propose two changes that increase the performance:

1. First, we increase the dimensionality of the latent codes z_C and z_B. To properly capture geometric details a larger latent space is beneficial, especially if the details can be fine-tuned during test-time training. We see optimal results with $\dim z_C = \dim z_B = 200$.
2. We change the design of the skip connection. In DeepMends architecture the skip connection replaces part of the neurons of the hidden layer instead of concatenating them. This architecture design reduces the usable latent dimensionality as well as the expressivity of the network. Instead we concatenate the information such that the hidden dimension is $512 + \dim z + \dim x = 715$ instead of 512.

Overall, these changes increase the number of parameters of our network design, which makes it harder to train (see Table 1 DeepMend vs. Ours w/o TTT). However, in combination with the test-time training the additional degrees of freedom allow us to represent the input data a lot more accurately.

6. Experiments

6.1. Implementation Details

Experiments are run in Pytorch 2.0.1 with CUDA version 11.7. All experiments are trained and evaluated on one NVIDIA GeForce RTX 3090 and AMD Ryzen 9 5900. Training one model took 8 hours, inference of the latent code 20 minutes and test-time training additional 30 minutes. The inference time for (Lamb et al., 2022a) was 45 minutes. Our model has 3,206,652 parameters and needs $2.03 \cdot 10^9$ additions and multiplications for one forward pass while (Lamb et al., 2022a) has 2,931,964 parameters and needs $1.896 \cdot 10^9$ additions and multiplications for one forward pass. We used Adam optimization with a learning rate of $5 \cdot 10^{-4}$ for the network parameters and 10^{-3} for both latent codes. The choice of the hyperparameters for inference are taken over from (Lamb et al., 2022a) and can be found there. For all experiments, we use Marching Cubes (Lorensen & Cline, 1987) to generate meshes from the implicit representation.

	DeepSDF		DeepMend		Ours w/o TTT		Ours w/ TTT	
	Mean	Median	Mean	Median	Mean	Median	Mean	Median
airplanes	2.36	0.61	6.37	1.92	4.67	0.98	**1.78**	**0.49**
bottles	**4.37**	1.43	5.94	0.45	6.56	**0.35**	5.09	**0.35**
cars	5.18	3.07	4.70	2.60	5.21	2.82	**3.15**	**1.14**
chairs	**8.36**	5.31	20.08	9.50	21.69	12.57	11.13	**1.5**
jars	**13.65**	5.93	37.92	6.65	28.52	6.95	16.40	**1.50**
mugs	4.26	**1.00**	5.27	3.02	5.34	1.02	**3.24**	1.21
sofas	**4.29**	2.60	8.10	3.57	9.39	4.13	5.60	**1.73**
tables	**8.41**	**3.35**	28.23	9.59	26.74	12.28	17.25	4.98
Mean	**6.36**	2.91	14.58	4.65	13.52	5.14	7.96	**1.43**

Table 1. Chamfer Distance ($\cdot 10^{-4}$) for the ShapeNet Dataset (Chang et al., 2015) on Dataset 1 in which the complete shapes have $5-20\%$ percent of their volume removed. Lower is better. The best value within each class is set **bold**.

	DeepSDF		DeepMend		Ours w/o TTT		Ours w/ TTT	
	Mean	Median	Mean	Median	Mean	Median	Mean	Median
airplanes	12.23	4.29	8.35	1.49	5.66	**1.1**	**4.18**	1.41
bottles	15.79	**8.07**	**11.84**	9.41	13.85	9.37	13.7	9.04
chairs	49.97	29.88	**22.34**	**10.1**	24.23	16.27	24.67	12.30
jars	74.86	69.97	35.29	22.82	38.27	25.51	**32.63**	**17.04**
mugs	42.13	11.18	**15.09**	6.2	16.19	5.06	15.23	**4.71**
sofas	27.10	17.36	11.44	7.6	11.31	6.0	**9.84**	**4.78**
tables	43.29	28.52	27.63	20.57	21.89	13.08	**18.79**	**7.62**
Mean	33.62	24.18	18.85	11.17	18.77	10.21	**17.00**	**8.13**

Table 2. Chamfer Distance ($\cdot 10^{-4}$) for the ShapeNet Dataset (Chang et al., 2015) on Dataset 2 in which the complete shapes have $45-55\%$ of their volume removed. Lower is better. The best value within each class is set **bold**.

6.2. Dataset

We evaluate our method on ShapeNet (Chang et al., 2015) which contains over 50 000 real-world scanned 3D objects of different classes. We train each object class separately and use a $70/15/15$ train/validation/test-split and 240 objects per class.

For preprocessing we fracture all objects using the approach of (Lamb et al., 2021). For different levels of complexity we create two different settings in which the percentage of removed volume varies. In **Dataset 1** the complete shapes have $5-20\%$ of their volume removed, and in **Dataset 2** it is $45-55\%$. We sample random points within the unit cube, calculate the signed distance function to determine if each point is inside or outside the mesh, and use this information to compute the occupancy of the the shape at the given points.

6.3. Metric

As evaluation metric we use the Chamfer distance (CD) between the ground-truth and predicted complete shape. It is defined as follows:

$$d_{\text{CD}}(X,Y) = \frac{1}{|X|}\sum_{x\in X}\min_{y\in Y}\|x-y\|_2^2 + \frac{1}{|Y|}\sum_{y\in Y}\min_{x\in X}\|x-y\|_2^2, \quad X,Y\subset\mathbb{R}^3. \tag{9}$$

6.4. Comparison

We compare ourselves to two main competitors. First, the classic method of DeepSDF (Park et al., 2019) which predicts the full SDF from a given partial shape. Second, the approach of DeepMend (Lamb et al., 2022a) which, while based on DeepSDF, proposed the separation into fractured and restoration shape. All tables also contain "Ours w/o TTT" which is our adapted network architecture of DeepMend (see Section 5.2) but without test-time training during inference. Notice that test-time training cannot be applied to DeepSDF directly as fine-tuning the complete shape to the input directly would lead to a complete overfitting of the input.

Our test-time training does add additional compute during inference. To make the comparison fairer, we reduce the number of training iterations for our method by the amount we use for test-time training. This means for every single

Figure 5. Qualitative examples of the different methods. Rows 1-3 and 6 are taken from Dataset 1, Rows 4-5 are from Dataset 2. Even though the Chamfer distance does not change much, the ability of our method to fit the fractured shape well, does visually a huge difference.

example all methods had the same computing power at their disposal.

6.5. Results

Dataset 1, Low Partiality Table 1 summarizes the errors on Dataset 1, which contains examples in which $5-20\%$ of the shapes are removed, for DeepSDF, DeepMend and our model with and without test-time training. For each class, the mean and median of the Chamfer distance over all test objects within one class is depicted. We can observe that the small adjustments of the network architecture (dimension of latent code, skip connection, see Section 5.2) led to improvements over DeepMend in most classes. Using test-time training, we improve the CD of our model in each category and overall by 41 percent (by even more for the median distance). Even though DeepSDF beats our model w.r.t. the mean most of the time, our median is lower in 6 of 8 categories. For a better understanding of the different behavior for mean and median, we refer to cumulative error curves in Figure 6. We see that our model produces more objects with a lower error but also a few bad outlier which

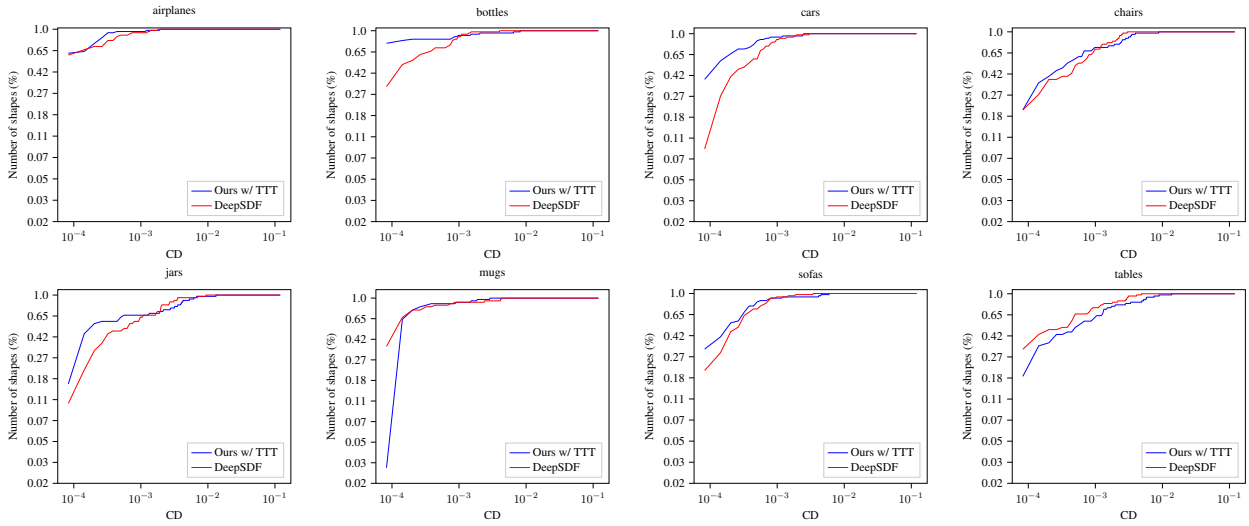


Figure 6. Logarithmic cumulative error curves (higher is better) of the chamfer distance (CD) for the different classes of Dataset 1. We compare our approach with test-time training against DeepSDF. We see that our model produces for more objects a smaller Chamfer distance, even though our model has more outliers.

influences the mean a lot. An example for this behaviour can be found in Row 6 of Figure 5.

Dataset 2, High Partiality The second dataset presents a bigger challenge as the removed volume is increased to $45 - 55\%$. The results in Table 2 show that, as for Dataset 1, our method tends to perform slightly better than DeepMend before test-time training and significantly better after. However, test-time training only improves by 9.4% which is due to the fact that the restoration part takes up around half of the shape on which the test-time training has less influence. Interestingly, DeepSDF performs much worse in this category than with less partiality. Our assumption for this behaviour lies in the different frameworks of the methods: While DeepMend as well as our model is specifically designed to learn a meaningful relationship between the fractured and restoration shape, DeepSDF has to encode the entire class geometry in a single model which leads to complex problems under harsh partiality. An indicator for this is that DeepSDF does perform quite well on the bottles class, which has the least intraclass variations, and is, thus, easier to capture in a single model.

Qualitative Results We show some qualitative examples of all methods in Figure 5. In the first and second row, we can see that all base models tend to overestimate the restoration part, but this behavior is drastically reduced with test-time training. Row 3 shows an example with a very fine detail that is not predicted by any of the base models. While without test-time training the beak of the vase is ignored, our model can adapt to these details such that we obtain better results for the restoration shape. Cases like this happen quite often and do not change the reconstruction error significantly because the details are quite small, but qualitatively the preservation of such details makes a big difference.

A similar but more extreme effect can be observed in row 4 where the vase contains a plant, a case which is not covered in the training data. Due to the test-time training, we can adapt to this while the other methods cannot. The shape of the complete vase is not quite correct but there is also no information in the original fractured shape about the geometry of the lower part.

6.6. Ablation Study

We evaluate the best latent dimensionality as well as the effect of the test-time training. The difference between using test-time training and not using it is reported in the complete results of Table 1 and Table 2.

To evaluate the effect of the size of the latent space, we choose $\dim z_C = \dim z_B$ and train all models on the mugs object class. We find that the model performance decreases when we increase the dimensionality of the latent code drastically. One explanation for this observation could be that too many degrees of freedom prevent learning of class information and instead overfit on the training data. The results are reported in Table 3. We used the best result of the ablation study, $\dim z_C = \dim z_B = 200$ for all our experiments.

$\dim z_C$	$\dim z_B$	$\dim z_C + z_B$	CD
128	64	192	1.9
100	100	200	1.4
200	200	400	**1.2**
300	300	600	3.6
400	400	800	3.1
500	500	1000	3.8
600	600	1200	3.9
1000	1000	2000	4.4
1300	1300	2600	6.0

Table 3. Ablation study for the dimension of the latent Code.

7. Conclusion

We proposed a new framework to solve the 3D volumetric shape restoration problem by using test-time training. The problem of shape restoration requires a combination of broad class knowledge but also the ability to very accurately fit to the input shape which is a setting particularly suited for test-time training. A simple class-wise trained network does often dismiss small geometric details, but we are able to retain them through finetuning during inference. Additionally, we have proposed several network improvements for DeepMend (Lamb et al., 2022a) and shown the effectiveness of our new pipeline on ShapeNet with two different levels of partiality.

Impact Statement. This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements. MM gracefully acknowledges funding of a Lamarr Fellowship of the Ministry for Culture and Science of the State of North Rhine-Westphalia. ZL was funded by a KI-Starter project of the Ministry for Culture and Science of the State of North Rhine-Westphalia during the majority of the project. The experiments in this paper were partially conducted using GPU resources of the OMNI cluster at the University of Siegen, which we were kindly given access to by the HPC team.

References

Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., and Davis, J. Scape: shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pp. 408–416, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 9781450378253. doi: 10.1145/1186822.1073207. URL https://doi.org/10.1145/1186822.1073207.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

Chibane, J., Alldieck, T., and Pons-Moll, G. Implicit functions in feature space for 3d shape reconstruction and completion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Cui, R., Liu, W., Sun, W., Wang, S., Shang, T., Li, Y., Song, X., Yan, H., Wu, Z., Chen, S., Li, H., and Ji, P. Neusdfusion: A spatial-aware generative model for 3d shape completion, reconstruction, and generation. *arXiv: 2403.18241*, 2024.

Dai, A., Qi, C., and Niessner, M. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Erkoç, Z., Ma, F., Shan, Q., Nießner, M., and Dai, A. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *European Conference on Computer Vision (ECCV)*, 2023.

Gafencu, M.-A., Velikova, Y., Saleh, M., Ungi, T., Navab, N., Wendler, T., and Azampour, M. F. Shape completion in the dark: Completing vertebrae morphology from 3d ultrasound. *arXiv: 2404.07668*, 2024.

Galvis, J. D., Zuo, X., Schaefer, S., and Leutengger, S. Scdiff: 3d shape completion with latent diffusion models. *arXiv:2403.12470*, 2024.

Gandelsman, Y., Sun, Y., Chen, X., and Efros, A. A. Test-time training with masked autoencoders. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Gao, J., Shen, T., Wang, Z., Chen, W., Yin, K., Li, D., Litany, O., Gojcic, Z., and Fidler, S. Get3d: A generative model of high quality 3d textured shapes learned from images. In *Advances In Neural Information Processing Systems*, 2022.

Hanocka, R., Metzer, G., Giryes, R., and Cohen-Or, D. Point2mesh: A self-prior for deformable meshes. In *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 2020.

Hu, J., Fei, B., Xu, B., Hou, F., Yang, W., Wang, S., Lei, N., Qian, C., and He, Y. Topology-aware latent diffusion for 3d shape generation. *arXiv:2401.17603*, 2024.

Kazhdan, M. M., Bolitho, M., and Hoppe, H. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, volume 256, pp. 61–70. Eurographics Association, 2006.

Lamb, N., Wiederhold, N., Lamb, B., Banerjee, S., and Banerjee, N. K. Using learned visual and geometric features to retrieve complete 3d proxies for broken objects. In *Symposium on Computational Fabrication*, pp. 1–15, 2021.

Lamb, N., Banerjee, S., and Banerjee, N. K. DeepMend: Learning occupancy functions to represent shape for repair. In *European Conference on Computer Vision (ECCV)*, 2022a.

Lamb, N., Banerjee, S., and Banerjee, N. K. Deepjoin: Learning a joint occupancy, signed distance, and normal field function for shape repair. *ACM Transactions on Graphics (TOG)*, 41(6):1–10, 2022b.

Lamb, N., Palmer, C., Molloy, B., Banerjee, S., and Kholgade Banerjee, N. Fantastic breaks: A dataset of paired 3d scans of real-world broken objects and their complete counterparts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023.

Lin, C.-H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.-Y., and Lin, T.-Y. Magic3d: High-resolution text-to-3d content creation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Liu, M., Xu, C., Jin, H., Chen, L., Xu, Z., Su, H., et al. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances In Neural Information Processing Systems*, 2023.

Loiseau, R., Monnier, T., Aubry, M., and Landrieu, L. Representing shape collections with alignment-aware linear models. In *International Conference on 3D Vision (3DV)*, 2021.

Lorensen, W. and Cline, H. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21:163–, 08 1987. doi: 10.1145/37401.37422.

Luo, S. and Hu, W. Diffusion probabilistic models for 3d point cloud generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Mehta, I., Chandraker, M., and Ramamoorthi, R. A level set theory for neural implicit evolution under explicit flows. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*, pp. 711–729, 2022.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Poursaeed, O., Fisher, M., Aigerman, N., and Kim, V. G. Coupling explicit and implicit surface representations for generative 3d modeling. In *European Conference on Computer Vision*, pp. 667–683. Springer, 2020.

Qiu, L., Chen, G., Gu, X., zuo, Q., Xu, M., Wu, Y., Yuan, W., Dong, Z., Bo, L., and Han, X. Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3d. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Sain, A., Kumar, A., Potlapalli, B. V., Chowdhury, P. N., Xiang, T., and Song, Y.-Z. Sketch3t: Test-time training for zero-shot sbir. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Sellán, S., Chen, Y.-C., Wu, Z., Garg, A., and Jacobson, A. Breaking bad: A dataset for geometric fracture and reassembly. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Sharf, A., Alexa, M., and Cohen-Or, D. Context-based surface completion. In *ACM SIGGRAPH 2004 Papers*, pp. 878–887. 2004.

Slavcheva, M., Baust, M., Cremers, D., and Ilic, S. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Sun, B., Kim, V., Aigerman, N., Qixing, H., and Chaudhuri, S. Patchrd: Detail-preserving shape completion by learning patch retrieval and deformation. In *European Conference on Computer Vision (ECCV)*, 2022.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A., and Hardt, M. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning (ICML)*, 2020.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for

volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Zeng, X., Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., and Kreis, K. Lion: Latent point diffusion models for 3d shape generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Zhu, Z., Chen, H., He, X., Wang, W., Qin, J., and Wei, M. Svdformer: Complementing point cloud via self-view augmentation and self-structure dual-generator. *arXiv:2307.08492*, 2023.

Zuffi, S., Kanazawa, A., and Black, M. J. Lions and tigers and bears: Capturing non-rigid, 3d, articulated shape from images. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.