Deep 3D-Shape Compression

Leonard Freißmuth
l.freissmuth@tum.de

Philipp Wulff
philipp.wulff@tum.de

Abstract

Implicit shape representations describe 3D-geometry in a differentiable manner, which is useful for downstream deep learning applications. An interesting property of these methods is their applicability as efficient compression tools. We experiment with periodic activations to improve key compression-properties of current implicit neural representations. Herein, we specifically compare memory-use and compression speed with the expressive power of the method. Our code is available here.

1. Introduction

Neural shape representations store shapes implicitly within the weights of a neural network. An effective representation is expressive, efficient and compact [8]. It requires rich *expressive capacity* to reconstruct shapes with high fidelity, *efficiency* to allow fast rendering at inference time, and *compactness* to scale to large scenes with low memory-footprint, while at the same time *generalizing* well across a variety of different shape classes.

Reducing the compactness of neural representations is particularly interesting because it creates new use cases in data compression, e.g. for streaming. Prior work looked at storing spatial data efficiently through encoding hierarchical sparse grids of latent codes [7, 10] and learning shape-priors that are shared between shapes from various semantic categories [4]. Shape priors can be understood as implicit knowledge about geometric features defining the appearance of a certain object, e.g. toroidal parts of an airplane's landing gear.

In this work, we leverage shape-priors to build an efficient, yet accurate implicit representation. We use a multi-layered-perceptron (MLP) to encode prior knowledge about a semantic class and query shapes via compact per-shape latent codes. In large, we research the relationship between the expressiveness and compactness of our implicit representation and find that using improve periodic activation functions boosts its expressiveness significantly.

2. Related Work

Traditional shape representations store spatial information explicitly in a 3D voxel-grid with occupancy or distance values, in a point cloud, or as a mesh, representing surfaces through a set of polygons. Implicit methods regress the signal of interest from coordinates in order to achieve a more memory-efficient representation compared to explicit methods. The sinusoidal representation network (SIREN) proposed in [9] uses an MLP with periodic activations in the form of sines to fit a function to input coordinates. Using oriented point clouds sampled from a mesh, SIREN accurately fits complex scenes with a five layer MLP. Furthermore, the authors learn shape-priors through a hyper-network that regresses the weights of a SIREN.

DeepSDF [8] represents shapes using a probabilistic auto-decoder conditioned on shape-specific latent codes. The auto-decoder is a positional MLP that regresses a 3D signed-distance field (SDF) and and is trained on individual ShapeNet classes [2]. Herein, the representation learns the SDF from the combination of the per-shape latent code and 3D positions, but is not explicitly aware of small-scale geometric details that are shared between shapes. Thus, DeepSDF does not generalize well to objects from classes not seen during training.

Other methods [4, 7, 10] combine explicitly stored shape codes with implicit representations in order to strike a favorable balance between memory-efficiency and fast inference time. ACORN [7] and NGLOD [10] utilize a multi-scale approach using a sparse voxel octree to allocate network capacity to regions of high complexity. Both methods achieve fast inference through compact decoder networks, but do not leverage the idea that most 3D surfaces share geometric details at some scale. LIG [4] trains an auto-encoder on local parts of a truncated SDF (tSDF), such that the network is able to extract the local geometry into a part-latent-code. Since local parts and their latent codes are shared between shapes, this representation generalizes well to shapes from different classes.

3. Method

We learn a positional mapping from latent code $\mathbf{z}$ to a continuous SDF as a 3D shape. By conditioning this mapping on a per-shape latent code, we can model multiple shapes' SDFs with a single neural network that learns the common properties and embeds them in a low-dimensional latent space.

3.1. Representation of SDFs

We base our work on DeepSDF's [8] auto-decoder-based shape-coded formulation. Formally, given some shape indexed

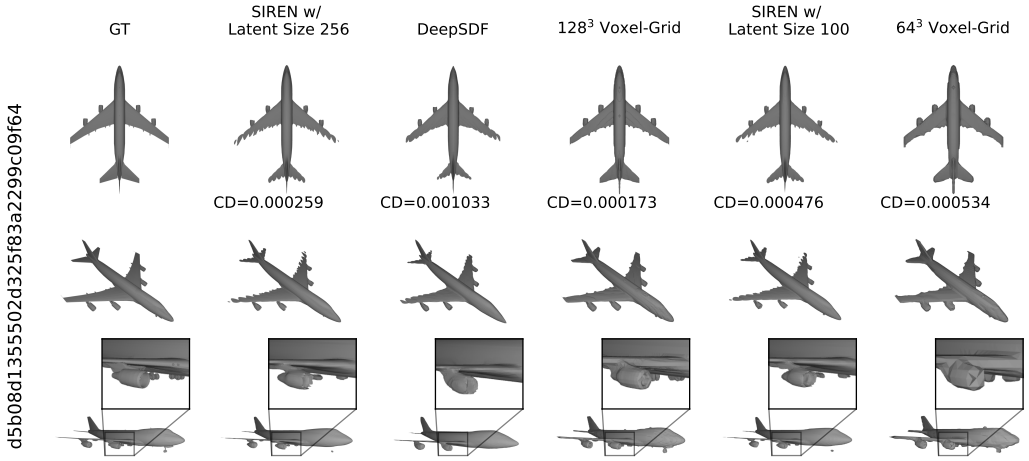

Figure 1. **Qualitative evaluation results** of reconstructions on the test set for the DeepSDF [8] baseline, our sine-based MLP with 100- and 256-sized latent codes, and voxel grid representations of resolutions 64 and 128. We report the CD for each reconstruction.

by i, we learn the function f_θ which maps a latent vector $\mathbf{z}_i$ and a 3D location $\mathbf{x}$ to the respective signed-distance value $\text{SDF}^i(\mathbf{x})$, i.e. $f_\theta(\mathbf{z}_i,\mathbf{x}) \approx \text{SDF}^i(\mathbf{x})$. Given a dataset of N shapes represented via their signed-distance function $\text{SDF}^i{}_{i=1}^N$, we prepare a set of K per-shape point samples with corresponding signed-distance values sampled both near the surface and uniformly:

$$X_i = \left\{ (\mathbf{x}_j, s_j) : s_j = \text{SDF}^i(\mathbf{x}_j) \right\}_{j=1}^K. \tag{1}$$

Loss Function Formally, we optimize f_θ by maximizing the posterior over the shape code $\mathbf{z}_i$ given the sample X_i, i.e. $p_\theta(\mathbf{z}_i|X_i)$ [8]. In practice, this is equivalent to minimizing the deviation between the network prediction $f_\theta(\mathbf{z}_i,\mathbf{x})$ from the actual signed-distance value s_i. We use the following clamped L_1 loss function:

$$\mathcal{L}(f_\theta(\mathbf{z}_i,\mathbf{x}), s_i) = |\text{clamp}(f_\theta(\mathbf{z}_i,\mathbf{x})) - \text{clamp}(s_i)| \tag{2}$$

with $\text{clamp}(x,\delta) := \min(\delta, \max(-\delta, x))$, where δ controls at which distance from the zero-level we truncate the SDF. Thus, larger values of δ allow faster ray-tracing because more samples contain information about safe step sizes and smaller values of δ focus the network capacity near the surface. We set $\delta = 0.1$ in all results presented hereafter. The complete training loss over all point samples for all training shapes becomes:

$$\mathcal{L}_{\text{epoch}} = \sum_{i=1}^{N} \underbrace{\left(\sum_{j=1}^{K} \mathcal{L}(f_\theta(\mathbf{z}_i,\mathbf{x}), s_i) + \lambda \|\mathbf{z}_i\|_2 \right)}_{\text{per-shape loss } \mathcal{L}_i}, \tag{3}$$

where λ weights the regression loss on the latent code length and is set to $\lambda = 10^{-4}$. We found the addition unit gradient constraint on the learned SDF (*Eikonal loss* [9]) not to be beneficial.

At training time, we minimize $\mathcal{L}_{\text{epoch}}$ w.r.t. the respective latent codes $\{\mathbf{z}_i\}_{i=1}^N$ and network parameters θ. At inference time, with fixed network parameters θ, we estimate a new shape code $\hat{\mathbf{z}}_i$ for a hold-out test shape X_i by minimizing $\mathcal{L}_i$ w.r.t. the latent code parameters $\mathbf{z}$.

Sine Activations In order to convert DeepSDF's auto-decoder into a SIREN after Sitzmann et al. [9], we replaced ReLU activations with sine activations and initialized all weights uniformly between $\pm\sqrt{\frac{6}{30^2 * N_{in}}}$, where N_{in} is the number of neurons in that layer.

3.2. Architecture, Training and Testing Details

We stack the latent code $\mathbf{z}_i$ and position vector $\mathbf{x}$ and feed them into a fully-connected (MLP) with constant width in all hidden layers. Every hidden layer is followed by a ReLU or sine activation layer, except for the output layer, which has no activation. In addition to the input layer, we feed the position vector into every hidden layer and optionally pass the latent code into selected hidden layers. We use Adam [5] with default parameters for the optimization. We use an L_1 instead of an L_2-based loss, since this results in more robust performance at inference time, likely due to outliers in the data. All models are trained for 2,000 epochs.

3.3. Data Set

We train our models on SDF samples X (Eq. 1) retrieved from ShapeNet [2] meshes. Herein, we use DeepSDF's [8] preprocessing pipeline, which computes signed-distance values at spatial locations $\mathbf{x}$ by first setting up equally spaced virtual cameras around the object. Then surface intersection with per-pixel rays are computed and stored together with the local surface normals pointing towards the camera. Finally, each point $\mathbf{x}$'s signed-distance value is computed by calculating its distance to the closest surface sample x_c with normal n_c. The sign equals the one of the dot product $n_c^T(x - x_c)$.

To generate training data, we sample more aggressively near the mesh surface in order to accurately model the SDF's zero level set. To retrieve SDF values for evaluating a voxel grid, we use SDFGen [3], which employs the same algorithm.

For our training, we focused on the class of planes, because they contain large regions with low-frequency features, as well as many high-frequency details compared to other classes. We split the shapes into a train set of size 1,780 and a test set of size 456. Passenger planes, airliners and private jets are dominant subclasses, but there are many "outlier classes" containing few shapes, such as bombers or cartoon style planes. To be more robust to these outlier shapes, we optimize the L_1 loss, which we also use for our assessments of the decoder's performance.

3.4. Evaluation

We measure the similarity of two meshes using the Chamfer Distance (CD) [1]. The Chamfer distance d_{CD} is the mean of the pairwise smallest bidirectional distances between two point clouds X and Y of size N:

$$d_{\text{CD}} = \frac{1}{N}\sum_{x\in X}\min_{y\in Y}||x-y||_2^2 + \frac{1}{N}\sum_{y\in Y}\min_{x\in X}||x-y||_2^2. \qquad (4)$$

We apply CD to meshes by sampling surface points and computing the CD on the resulting point clouds. A disadvantage of using Chamfer distance is that it weighs low-frequency features covering large surfaces more than high-frequency details, since these occupy less surface area and are therefore represented by fewer surface samples than the low-frequency features. To this end, we propose a curvature-weighted sampling strategy using mean Gaussian curvature as an additional weight to the surface area. Since curvature is proportional to frequency, we use this *curvature Chamfer distance* to evaluate reconstruction quality of high frequency features in specific.

4. Results

We conduct three main experiments to examine our leading research questions. Namely, about the relationship between network capacity and expressiveness and the impact of using periodic activations in implicit representations of SDFs. More specifically, we

- train auto-decoder networks of increasing capacity with sine and ReLU activations and evaluate their reconstruction quality on train and test shapes (Sec. 4.1),
- evaluate the potential of sine activations at representing high-frequency surface details by measuring the curvature-based CD of the test-set reconstructions (Sec. 4.2), and
- compare our implicit shape-coded representation against explicit dense and sparse voxel grids w.r.t. storage space and reconstruction quality (Sec. 4.3).

4.1. Sine Activations Boost Network Expressiveness

First, we evaluate the relationship between the capacity of our implicit representation and its representative performance with sine or ReLU activations. To this end, we scale the number of network parameters by changing the number of layers for three fixed aspect ratios of the decoder network. In total, we trained 8 ReLU-based decoders and 9 sine-based decoders with 37k to 1.84M network parameters with a constant 200-dimensional latent space.

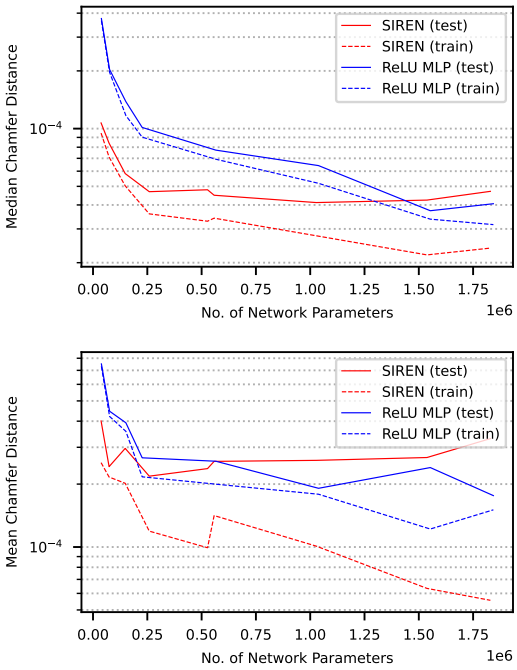

Figure 2. Relationship between the number of decoder parameters and the reconstruction quality, measured using the median CD (top) and mean CD (bottom) on train- and test-set shapes. All architecture were trained for 2000 epochs and use latent codes of length 200.

Fig. 2 reports the *median* CD of train-set and test-set reconstructions for decoders of increasing size. Herein, networks of any size that use sine activations achieve better CDs on *known* shapes from the train set than the respective equally-sized ReLU MLP. Moreover, the CD curves of either architecture on the train set feature a fast initial drop-off, followed by a steady decline. We find that training large SIREN requires small learning rates compared to ReLU MLPs, such that the largest SIREN was likely not trained long enough. Smaller SIRENs beat ReLU MLPs on reconstructing *unknown* test-set shapes, but when using more than ∼1.5M parameters, ReLU MLPs perform better. Because of the difference in median CD on the train- and test-sets for SIREN, we hypothesize that they *overfit* to the train shapes. Given that a SIREN has greater capacity for representing train shapes than an equally sized ReLU MLP, we argue that this additional capacity results in this overfit.

Method\ Metric	CD, mean ↓	CD, med. ↓	CD, curv. med. ↓	No. of netw. Parameters
DeepSDF [8]	**0.1467**	0.0400	0.0404	1.84M
SIREN (ours)	0.1662	**0.0318**	**0.0319**	586k
SIREN [9] w/ int. activations	0.2493	0.0426	0.0426	457k

Table 1. **Quantitative evaluation results** of test-set shape reconstructions for the DeepSDF [8] baseline, our sine-based MLP and our method with interpolated activations. All results are multiplied with 10^3.

Outperforming DeepSDF We train a SIREN with 8 equally-sized 256-neurons-wide layers and latent code size 256 and compare this against DeepSDF with more than 3 times as many parameters. In addition to the first layer, we feed the $\mathbf{x}$ location vector into all and the latent vector into the 4th hidden layer. Our method outperforms DeepSDF on the median CD, but produces a worse mean CD, see Table 1. The increase in mean CD is likely due to the overfit described earlier. As shown in Fig. 2 (bottom), the mean CD on the train-set is smaller for SIRENs than for ReLU MLPs. Because the divergence between the mean CDs for SIRENs of increasing network size is larger than between the median CDs, the network overfits to the majority of the shapes while ignoring outliers. This behavior is encouraged by our L_1-based loss function. We hypothesize that appropriate regularization of the SIREN results in a competitive (or better) mean CD compared to DeepSDF.

Interpolating Activations We conduct an experiment in which the network chooses its activation by learning the the mixing weights of linear interpolations between sine and ReLU activations. When initializing all interpolation weights at 0.5 we find that this architecture does not beat our best SIREN model on test-set reconstructions (Tab. 1). The interpolation weights in all layers except the final one move towards ReLU activations during training. We believe, that numerical instability of the interpolation layer biases the optimization towards ReLU activations.

4.2. Reconstructing High Frequency Features

Test-set reconstructions from DeepSDF [8] lack regions of high detail, e.g. turbine inlets or landing gear, as seen in Fig. 4. The authors of [9] propose sine activations to increase capacity for representing high frequency features. To this end, we measure the reconstruction quality of surface details using our curvature-based CD metric (from Sec. 3.4) and report the results in Table 1. SIREN outperforms DeepSDF by ∼21% on this metric, as opposed to ∼20.6% on the median CD. Moreover, visual inspection of the results in Fig. 4 shows that our SIREN reconstructs surface details such as the landing gear, which is not present with DeepSDF. Herein, our SIREN uses less than 1/3 of DeepSDF's parameters.

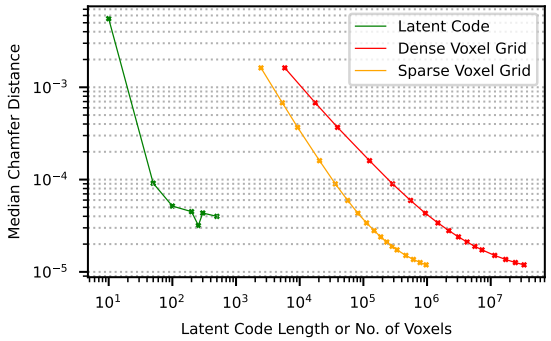

Figure 3. Relationship between the latent representation's size and its reconstruction quality measured using the median CD on the test set, compared against dense voxel grids and sparse voxel grids.

4.3. Comparing Shape Codes to Voxel Grids

Finally, we compare our implicit SDF representation against 3D grids with signed-distance values. We use our best SIREN architecture from Table 1 and train models with 10 and 500 latent code parameters. We extract dense voxel grids with resolutions between 16 and 320 using the method outlined in Sec. 3.3 and retrieve reconstructions by applying marching cubes (MC) [6]. To compare against sparse voxel representations, we imitate them by discarding all values that are further than one voxel diagonal away from the zero-level set. This produces a lower bound on the storage requirements of sparse voxel grids. Results from this experiment are plotted in Fig. 3: while shape-coded auto-decoder models with latent codes larger than 20 outperform even the smallest voxel grids, the implicit representation's performance begins to saturate before the voxel grids reach their peak representative power. Our best model with latent code size of 256 is competitive to a 128-cubed sparse voxel grid which stores 575 times as many values.

5. Limitations, Conclusion and Future Work

We note that SIRENs are more sensitive to training hyperparameters, such as batch size, learning rate and network size, than comparable ReLU MLPs. Moreover, we struggled with overfitting when using sine activations, which we attribute to a higher expressive power. Because of this, we conclude that use of sine activations requires strong regularization and that this needs further research, for instance on the improvement through data augmentation.

Despite the overfit, we achieve competitive results to DeepSDF with 1/3 of network parameters. This reduces training and inference time and is beneficial for training larger models on more data. Finally, we have proven that shape-coded auto-decoders are a memory-efficient alternative to low-resolution voxel grids for applications with few semantic classes. It is interesting to expand to more diverse classes in future work.

References

[1] Gunilla Borgefors. Distance transformations in arbitrary dimensions. Computer Vision, Graphics, and Image Processing, 27(3):321–345, 1984. 3

[2] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 1, 2

[3] GitHub. christopherbatty/sdfgen: A simple commandline utility to generate grid-based signed distance field (level set) generator from triangle meshes, using code from robert bridson's website, 27.03.2023. 3

[4] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6001–6010, 2020. 1

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 2

[6] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. SIGGRAPH Comput. Graph., 21(4):163–169, aug 1987. 4

[7] Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation. arXiv preprint arXiv:2105.02788, 2021. 1

[8] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 165–174, 2019. 1, 2, 4, 5

[9] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. Advances in Neural Information Processing Systems, 33:7462–7473, 2020. 1, 2, 4

[10] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 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, pages 11358–11367, 2021. 1

A. Appendix

A.1. Reconstructions

We present additional good reconstructions in Fig. 4 and failure cases in Fig. 5 of our best performing model compared against DeepSDF [8] and voxel grid reconstructions with 64^3 and 128^3 grid resolutions.

A.2. Hyperparameters

The exact hyperparameters we used to train all our architectures can be found in the "examples" directory of our public GitHub repository.

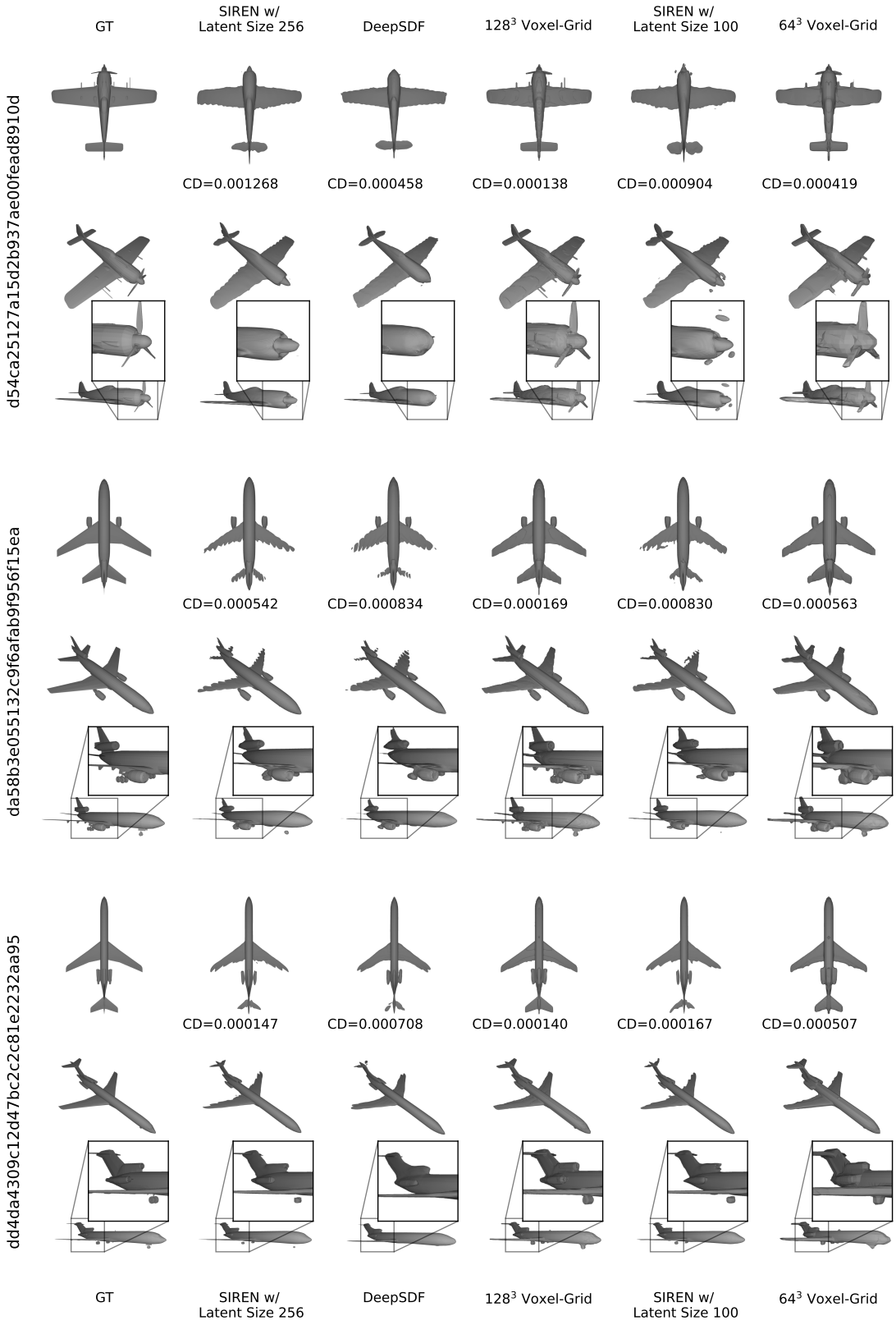

Figure 4. **Success cases** of our model and DeepSDF.

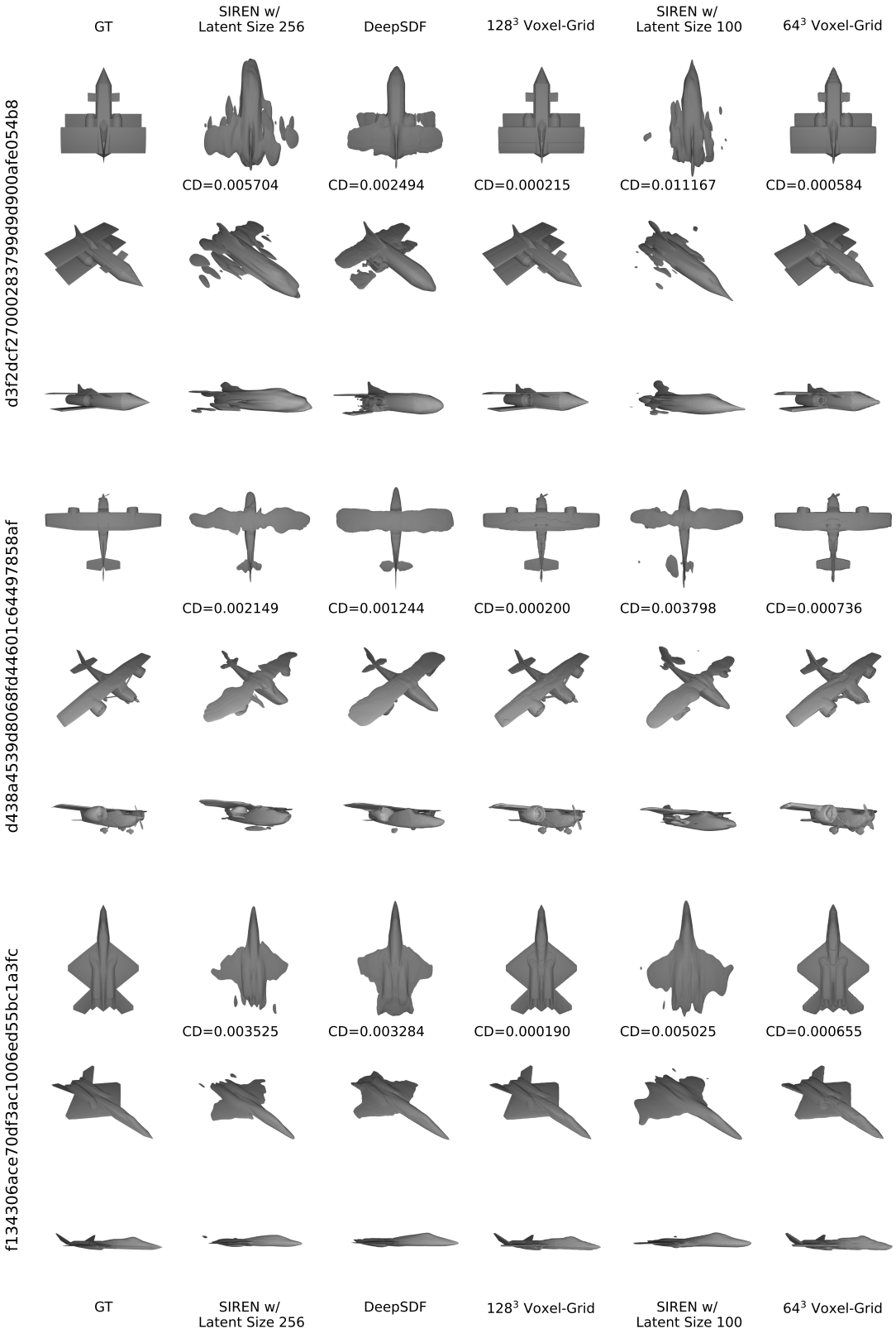

Figure 5. **Failure cases** of our model and DeepSDF.