Neural Volumetric Envelopes

Aditya Ganeshan* Brown University
Anh Truong* Brown University
Marc Mapeke* Brown University
Mikhail Okunev* Brown University

Abstract

While neural networks have been widely deployed to estimate Signed Distance Fields (SDFs) for objects, their wide spread usage remains low due poor editability as well as high computational and memory cost. In this work, we propose Neural Volumetric Envelopes (NVE), a novel method for constructing SDFs which addresses these issues. We build on two important insights: 1) Encapsulating the input shape in a coarse bounding volume allows us to discard SDF evaluation beyond the bounding volume, and 2) Shapes consist of re-occurring local SDF patterns which can be shared across instances. NVEs store the sign distance field with a coarse volumetric partitioning of the input shape, where each partition extracts feature vectors at its vertices which are then used to predict the sign distance field defined within the partition. Additionally, the extracted feature vectors are mapped to a fixed-size codebook using Vector Quantization. On a set of 50 *shapes high resolution meshes, our method yields a* ~ 1000 *times compression while modelling all the shapes with high accuracy.*

1. Introduction

Traditionally in computer graphics 3D geometry is represented with triangular meshes. This representation offers multiple advantages. It is conceptually simple, can approximate arbitrary shapes and well-supported by the hardware. However, there are also certain disadvantages that meshes have. Since mesh is essentially a linear approximation of geometry, a large number of triangles might be required to represent a nonlinear shape, resulting in a significant memory cost for high-quality models. This work explores an alternative representation called Neural Volumetric Envelopes (NVE) that achieves both high reconstruction quality and low storage cost.

To achieve this goal Neural Volumetric Envelopes partition space into disjoint regions called envelopes and represent geometry inside each envelope with Signed Distance Fields (SDFs). SDFs are an implicit representation that encode surface as a zero level of a learned function $f(x, \theta) = 0$ and can be parameterized efficiently with neural networks. To achieve high compression rate we learn a compact yet general set of SDFs over a subset of ShapeNet dataset which is reused to represent any new shape.

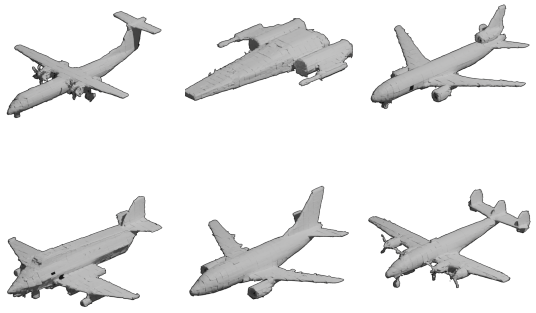


Figure 1. Our method, NVE simultaneously models a large dataset of shapes, while compressing the memory requirements to store these shapes by a factor of ~ 1000.

In summary, we make the following contributions:

1. The formulation and the training method for a novel geometry representation called Neural Volumetric Envelopes,
2. Demonstration of efficiency of our method on a subset of ShapeNet shapes (planes) where we were able to achieve compression rates on the order of 1000x compared to the original mesh.

2. Related Works

2.1. Implicit Shape Representations

Geometry in computer graphics can broadly be partitioned into two classes: explicit and implicit. Explicit representations, such as point clouds or meshes, directly describe the locations of geometric primitives, making them efficient to process but at the expense of discretization dependence. In contrast, implicit representations encode surfaces as the level sets of fields (e.g. distance or occupancy fields), thus allowing them to be modeled by continuous functions of space. Recently works have successfully used neural networks to parameterize these functions [1, 4, 6]. In partic-

* denotes equal contribution

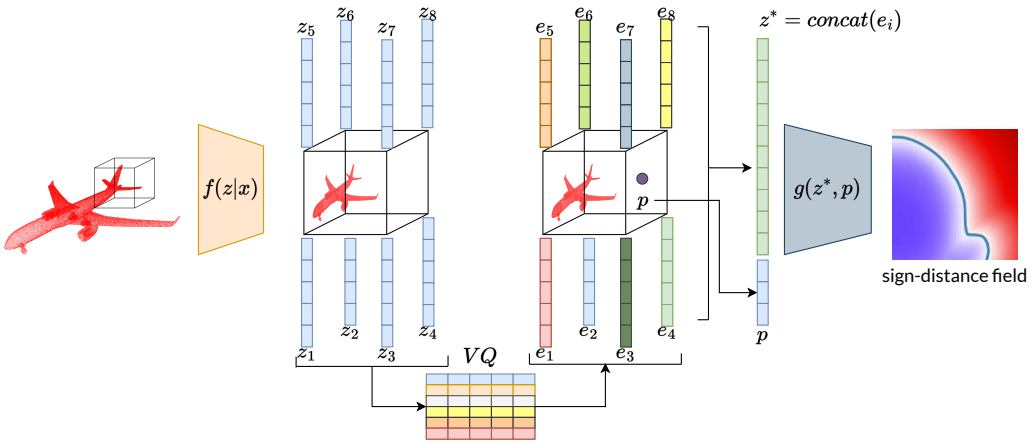


Figure 2. Neural Volumetric Primitives: We first create partitions of the input shape via voxelization. For each voxel, we sample shape surface points which are taken as input by a PointNet [8] model $fz|x)$ to extract feature vectors z_i. These features are then quantized using the codebook, before being used by a two layer MLP $g(z^*, p)$ for predicting the sign distance field at point p.

ular, Park et al [6] learn the surfaces of shapes as the zero level sets of SDFs. Neural network-parameterized functions of space or time such as this belong to a family of networks called Neural Fields [12].

2.2. Hybrid Neural Fields

Many seminal neural field architectures rely on global conditioning [1,4,6], where each output instance can be decoded from a single global latent vector [12]. Later works explored local conditioning, utilizing spatial data structures – such as regular grids [7], adaptive grids [10], point clouds [14], and meshes [2] – to condition field outputs on local latent features. These hybrid representations [12] allow neural fields to reason locally, a property that we exploit in this work.

2.3. Neural Field Compression

To counteract the large memory footprint often required by hybrid neural fields, several recent works explore data compression in feature space. While adaptive spatial data structures can reduce memory usage by discarding features in perceptually irrelevant locations [10], they alone are not sufficient for scaling to large *classes* of shapes or scenes, where representing each instance may still require a unique set of latent features.

An increasingly common approach for neural field compression is to use vector quantization [11] by limiting the bitrate of the latent features, e.g. by discretizing the feature space [3, 5, 9, 13]. For example, Yan et al. [13] use vector quantization to represent shapes' feature grids as sequences of indices into a shared codebook. While a large codebook may be necessary to expressively encode a variety of local geometry, its footprint is amortized across the entire class of shapes: for a large collection of shapes, the size of the codebook is insignificant relative to the collective size of the meshes.

3. Method

Our method consists of three components, an envelope creation procedure as described in section 3.1, a neural network to predict envelope features from the mesh, which we present in section 3.2, and a finally a point estimation network to predict the SDF-value of a given point inside the given envelope as noted in section 3.3.

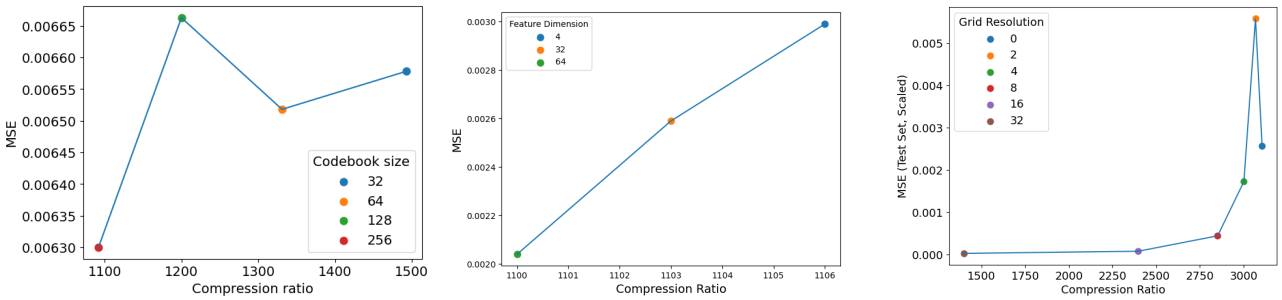
3.1. Creating Envelopes via Voxelization

We aim to create a coarse bounding structure around the input shape such that 1) The input shape is fully enclosed by the structure, and 2) The input shape is partitioned into smaller volumes with roughly similar SDF complexity. To achieve these two goals, we rely on voxelization of the input space into discrete cubic voxels of a fixed resolution N. Now, each voxel contains a partition of the input shape (we discard voxels with no overlap with input shape) and our goal is to estimate the sign distance field inside each voxel, given the enlcosed paritioned input shape.

3.2. Extracting Envelope Features

Each voxel contains a partition of the input shape, and our goal is to estimate the enclosed sign-distance field. Towards this end, we employ a neural network $f(z|x)$ (where x represents the partitioned input shape and z represents the extracted feature), to extract features from the partitioned input shape which can then be supplied to a point estimation network.

Processing mesh inputs requires complex machinery as meshes can contain arbitrary manifold and connectivity.



(a) Codebook Size (K) (b) Code Feature Dimension (L) (c) Grid Size (N)

Figure 3. (a) Codebook size represents a tradeoff between compression and reconstruction quality. (b) Having a larger code-size is beneficial for reconstruction accuracy, at a minimal drop in the compression ratio. (c) Though using lower grid resolution for voxelization yields higher compression, it leads to weaker generalization to novel shape instances.

Therefore, we instead convert the partitioned input mesh into a point cloud by sampling points on its surface. The sampled point cloud is then processed by a PointNet [8] network to extract features. While it is feasible to extract a single feature for each envelope, this does not allow $f(z|x)$ to exploit reflections between different envelopes. Instead, we predict 8 features $\in R^L$, one for each of the vertices in a voxel. We expect the ability to predict 8 different features to help the this network model envelopes which are reflections along a given axis by simply predicting features flipped about the axis of reflection.

As our goal is to compress the dataset of shapes, we perform vector-quantization on the predicted features. Essentially, we build a vector codebook $\in R^{K\times L}$ with K features. Following [11], during the forward pass, each extracted feature $f(x)$ is replaced by its nearest neighbor e in the code book while two additional losses are introduced to ensure the extracted features and codebook features stay consistent. These losses are as follows:

$$L_{VQ} = ||sg[f(x)] - e||^2 + \beta||f(x) - sg[e]||^2, \quad (1)$$

where sg is the stop gradient operator, $f(x)$ is the extracted feature, e is the nearest neighbor of $f(x)$ in the codebook and β is a loss weighting hyperparameter.

3.3. Sign Distance Field Estimation

Finally, our goal is to predict the SDF value at arbitrary points within a given voxel, conditioned on the extracted vector-quantized features $e_i, i \in \{1, 8\}$. Following [10], We use a simple 2 layer fully-connected neural network $g(y|z)$ to predict the SDF-values. This network takes as input the concatenation of features e_i along with a input point $p \in [-1, 1]^3$ to predict $sdf(p)$. We now train all our components: network $g(.)$, the codebook, and network $f(.)$ end-to-end using the ground-truth SDF values $sdf(p) = y$ with the following loss:

$$L_{tot} = ||y - g(VQ[f(x)], p)||^2 + L_{VQ}, \quad (2)$$

where $VQ[.]$ is the vector-quantization process as described in section 3.2.

4. Experiments

4.1. Implementation Details

For our experiments, we voxelize all input shapes into a 32^3 grid. We sample 1024 surface points in each envelope/voxel (used by $f(z|x)$). Additionally, we sample points along with their ground truth SDF values for training our model. These points are sampled close to the surface, as well as uniformly across each voxel. As a preprocessing step, we ensure that all our meshes are water-tight. We train our model end-to-end using Adam with a learning rate of 0.003 with a batch size of 256 envelopes. Additionally, we weigh the vector quantization commitment loss by 0.5. The training takes $\tilde{2}4$ hours on a RTX 3090 GPU to converge.

Dataset: We use a subset of ShapeNet Core dataset for our experiments. We use 50 Instances of the Plane object category. Additionally, we use a seperate set of 10 Instances for the experiments in Figure 3c.

Evaluation Metrics: We measure the reconstruction accuracy via the mean squared error betweeo predicted SDF and ground truth SDF. We measure compression as a ratio of the old dataset size (size of all .obj files) to the new dataset size (size of new obj files, codebook & neural networks).

4.2. Ablations

In order to further explore the relationship between compression and accuracy within our model, we ablate over three hyparameters: the number of features in the codebook (K), the input space grid resolution (N), and the dimension of latent featuer codes (L).

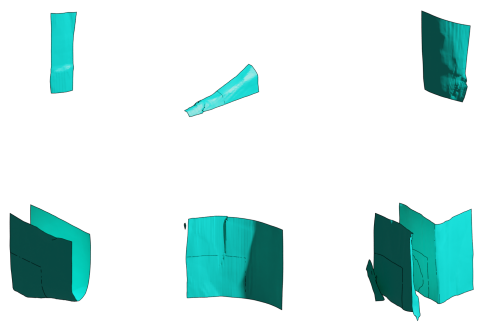


Figure 4. We visualize the surface extracted using marching cubes in different envelopes. These primitives are composed together to create entire shapes.

4.2.1 Codebook Size (K)

To explore the importance of codebook size we trained the system with $K = \{32, 64, 128, 256\}$. Our findings, summarized in Figure 3a are that codebook size represents an important tradeoff between compression and reconstruction quality. The reconstruction quality is better quantitatively and qualitatively with $K = 256$, especially for reconstructing high frequency details. However the compression ratio is worse by about 40% compared to smaller codebook sizes.

4.2.2 Code Feature Dimension (L)

We additionally explore the effect of the having different Code Feature Dimension L on our method. As shown in Figure 3b, changing L has minimal effect on the compression, as it only effects the codebook size, which is shared across the dataset. However, using smaller values for L does lead to a drop in the reconsturction accuracy.

4.2.3 Grid Resolution (N)

For this experiment, we trained six separate models on a single plane from the dataset. The input space voxelization for each model had a different grid resolution, $N = \{0, 2, 4, 8, 16, 32\}$. At test-time, the models were evaluated on ten planes that were unseen during training. The results of this ablation visualized in Figure 3c.

Figure 3c shows that higher grid resolutions in the input space lead to better model generalization at test-time. Finer voxelizations allow the model to learn proper encodings for local geometry, which can then be used to encode unseen shapes. As expected, there is also an inverse relationship between grid resolution and the compression ratio.

In contrast, some other methods for neural implicit shape representation, such as Park et al [6], rely on both global conditioning and auto-decoding. In order to represent a novel object with DeepSDF, one would have to optimize and store the high-dimensional feature codes.

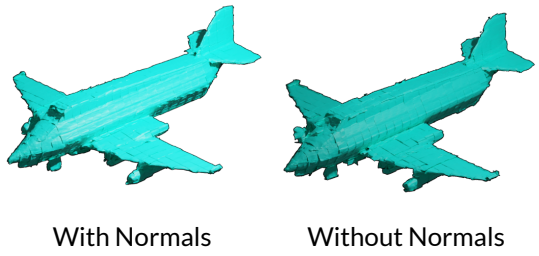


Figure 5. We compare mesh extracted from model trained (left) with normals, and (right) without normals. Providing surface normals, increases accuracy noticeably.

For our method to represent a novel object, we would only need to create the envelopes via voxelization, and then compute a forward pass with our pre-trained encoder to extract features for each envelope. Overall, this process is faster than auto-decoding latent features.

This framework also creates the possibility for generalization on out-of-distribution data, such as shapes in other classes, or scenes comprising of multiple types of shapes.

5. Conclusions

Shapes consist of many local patterns which are repeated across a shape as well as a shape category. What would happen if we replaced standard primitives used to represent shapes, such as triangles and quadrilaterals, with a set of learned and compressible primitives?

In this work, we present Neural Volumetric Envelopes, a hybrid neural representation for estimating and representing sign distance fields across a shape class. We demonstrate that Neural Volumetric Envelopes are able to accurately reconstruct a set of shapes from a single class. We also demonstrate that NVEs represent a predictable tradeoff between compression ratio and accuracy.

One limitation of our work is the disconnected geometry between neighboring envelopes. This can be fixed by overlapping our envelopes, which would equate to oversampling SDF values near envelope boundaries. Another limitation is the loss of geometric detail when converting the input mesh into a point cloud. Because our formulation is modular, we can replace the Envelope Feature extractor with a mesh network, but this would require complex preprocessing to subdivide triangles across envelope boundaries.

In future work, we plan to investigate different formulations for the creation and representation of our envelopes. This includes experimenting with different envelope structures, such as tetrahedrals instead of voxels, and also supporting adaptive envelopes to better represent rotational and translational symmetry across shapes.

References

[1] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *CoRR*, abs/1812.02822, 2018. 1, 2

[2] Jun Gao, Wenzheng Chen, Tommy Xiang, Clement Fuji Tsang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3d reconstruction. *CoRR*, abs/2011.01437, 2020. 2

[3] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Liefeng Bo. Compressing volumetric radiance fields to 1 mb, 2022. 2

[4] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *CoRR*, abs/2003.08934, 2020. 1, 2

[5] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. Autosdf: Shape priors for 3d completion, reconstruction and generation, 2022. 2

[6] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *CoRR*, abs/1901.05103, 2019. 1, 2, 4

[7] Songyou Peng, Michael Niemeyer, Lars M. Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. *CoRR*, abs/2003.04618, 2020. 2

[8] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2016. 2, 3

[9] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. Variable bitrate neural fields, 2022. 2

[10] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles T. Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. *CoRR*, abs/2101.10994, 2021. 2, 3

[11] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *CoRR*, abs/1711.00937, 2017. 2, 3

[12] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *CoRR*, abs/2111.11426, 2021. 2

[13] Xingguang Yan, Liqiang Lin, Niloy J. Mitra, Dani Lischinski, Danny Cohen-Or, and Hui Huang. Shapeformer: Transformer-based shape completion via sparse representation. *CoRR*, abs/2201.10326, 2022. 2

[14] Biao Zhang, Matthias Nießner, and Peter Wonka. 3dilg: Irregular latent grids for 3d generative modeling, 2022. 2

Appendix

6. Contributions

All team members equally contributed for the report writing and project presentations.

Aditya Ganeshan: Code Architecture, training, evaluation & compression calculation scripts. Blender rendering scripts for shapes/envelopes. Analysis of codebook and envelopes.

Marc Mapeke: Envelope to Feature Network, Marching Cubes Visualization, Grid Size Ablation, Dataloader, Debugging.

Michael Okunev: Feature to point network. Logging. Codebook size ablation.

Anh Truong: Mesh data cleaning. Training point/SDF data generation. Surface point normal estimations. Data processing for per-evenelope data. Feature dimension ablation.