Deep generative models for engineering design

Fan, J.

Chapter 4

Shape Generation with Learning-free Decomposition

Different from previous chapters, from this chapter on we start targeting on generation of 3D models. The AI community has made substantial progress in learning mesh data, however, introducing current mesh-based DGMs into industrial design processes is not trivial, because industrial meshes tend to be extremely high-dimensional. To address this, we present a new pipeline that enables learning on meshes by decomposing them into low-dimensional variables, without explicit training, answering the research question 3: *How to use DGMs to generate high-dimension designs more efficiently?* The content of this chapter has been published in the paper [168].

Figure 4.1: A gallery of 3D meshes generated by SpoDify.

4.1 Introduction

Today, 3D modeling has become a more convenient method in the industrial design process with the help of powerful CAD software, whereas traditional 2D blueprints have less advantages over 3D shapes, e.g., B-Reps and meshes, in terms of simulating usability, efficiency of design modification, and accuracy of representation. While B-Rep data remains a challenge in learning and generating, meshes [96, 154] are by far the most commonly used form in industry, as the native representation of many CAE software and finite element tools [138]. Besides, generating 3D shapes represents one of the major challenges of the deep generative modeling community, where the researchers have made substantial progress in directly learning on mesh data [135, 30, 199]. However, as a non-monotonous data representation, a mesh contains multiple modal data forms and their lengths vary with samples, where deep learning methods generally perform poorly. Most recently, research [32, 151, 64] in this field enabled the generation of meshes with signed distance field (SDF) [162], a powerful implicit representation that encodes the source mesh into a voxel, where the value of each voxel grid indicates a distance value from the grid position to the nearest surface of the source mesh. Here, a negative value indicates that the point is inside the shape, while a positive value indicates that the point is outside the shape. Representing a mesh in the voxel form allows the implementation of 3D convolutional neural networks (CNNs), which addresses the challenge of using deep learning on meshes. However, to produce high-fidelity shapes, a large dimension of the voxel-shaped SDF is often required, e.g., 256^3 [64, 123], which poses computational and temporal challenges for directly learning with deep generative models (DGMs).

Meanwhile, the trend in high-dimensional data generation has shifted toward encoding the source data into a low-dimensional latent space so that DGMs can efficiently learn from compact latent codes. Using a trained autoencoder is the most commonly used methodology. It has yielded powerful DGMs, e.g., latent diffusion models (LDMs) [139, 194], which can generate high-dimension data with much reduced computational cost. Encoding high-dimensional data has also been explored in the context of SDF representations [105, 90, 29]. However, the quality of the results generated by such pipeline largely depends on the performance of the autoencoder introduced, which remains challenging and computationally intensive to train. In fact, the amount of training samples needed to properly train an autoencoder drastically increases with the dimensionality and diversity of the target data, which tends to be impossible for real-world design cases.

Several existing approaches have used a learning-free encoding pipeline to obtain the latent variables of high-dimensional data [68, 22, 64]. Among them, neural wavelet-domain diffusion [64] (referred to as NWD in this chapter) achieves state-of-the-art (SOTA) performance in generating complex topology and structures with clean surfaces and fine details. However, the introduced diffusion model has to be trained on 3D voxels of dimension 130^3 for a single-level wavelet transformation. To fill the gap between mesh generation and neural latent learning, a fully deterministic approach can be used, such as singular value decomposition (SVD), which has been historically used for dimensionality reduction in deep learning tasks including data classification [74, 190] and image generation [73]. [73] leverages SVD eigenvalues as a loss regularization term for GANs training. Furthermore, SVD guarantees minimum information loss during the encoding, which is an essential characteristic for accurately reconstructing complex models. Even though SVD guarantees minimum information loss without the need for training, its application in generative 3d modeling remains understudied.

In this chapter, we propose to exploit this idea and design a novel mesh generation method, spectral-domain diffusion for high-quality shape generation (SpoDify), which uses a learning-free pipeline to encode meshes into low-dimensional spectral features that serve subsequently as the latent variables for training the diffusion-based DGM. We display our results in Figure 4.1, which are generated by learning on the ShapeNet dataset [26]. Compared to SOTA methods (3DShape2VecSe [195], NWD [64]) that rely on deep learning-based encoders or large data representations, our SpoDify can produce comparable results, and in some cases even superior, by using generative modeling in a 512-dimensional spectral space.

4.2 Method

Our method SpoDify is inspired by NWD [64], where we additionally introduce an SVD-based decomposition approach to achieve a more interpretable and computationally efficient encoding of mesh representations. We show a diagram of SpoDify in Figure 4.2.

4.2.1 Spectral Representation of Mesh

Clustering Our approach remains effective even with a limited number of training samples, provided the samples are representative of the larger dataset. We explain

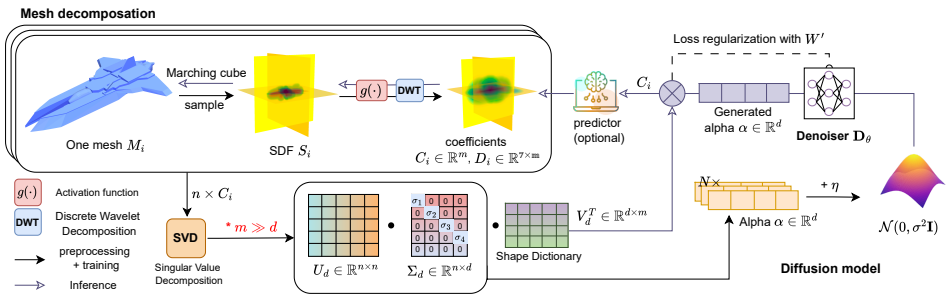


Figure 4.2: Diagram of SpoDify. We apply singular value decomposition on a set of the coefficients that are derived by applying a signed distance field and discrete wavelet transformation on source meshes, resulting in the dataset of spectral features. Here, the basis $V_d^\top$ will be stored for later generation; spectral features α will serve as one sample and will be used for training the diffusion model. To generate a new mesh, the trained diffusion model generates new α for a given random noise. The generated α will be denormalized and then multiplied with pre-computed and stored $V^\top$ to obtain new low-frequency coefficients C_i, which can be converted to new mesh M_i.

this in Section 4.2.3. We construct a diverse training set using fewer samples by introducing a clustering process. First, diffusion maps [35] are applied to the complete set of available meshes, using the Chamfer distance as the metric, embedding all meshes into a shared diffusion space that preserves geometric relationships. K-Means clustering is then used to partition the dataset into n clusters, capturing its diversity. From each cluster, one representative mesh is selected, ensuring broad representation even with a small subset. Pairwise Chamfer distances between all 3D meshes quantify shape similarity. The Chamfer distance is implemented using the GitHub repository [178]. The resulting distance matrix, capturing geometric similarities between shapes, is transformed into a similarity kernel matrix using an exponential function to enhance mesh relationships representation. Diffusion maps [35] are applied to reduce high-dimensional representation while preserving intrinsic geometric structure, extracting the top 64 eigenpairs for low-dimensional embedding of each mesh. This embedding captures significant modes of variation in the dataset. Clustering on the diffusion embeddings using K-Means identifies n cluster centroids as representative shapes encapsulating the dataset's diversity.

Implicit representation with SDF Next, we represent the geometry of the mesh with the SDF due to its differentiability and smoothness properties [45]. We sample from the n representative meshes $M_{1,..,n}$. Each mesh M_i is scaled to the range

$[-0.5, 0.5]^3$ to standardize its size and position and then represented as an SDF S_i of resolution 256^3, so that

$$f_{M_i}(x) = \begin{cases} d(x, \partial M_i) & x \in M_i, \\ -d(x, \partial M_i) & x \notin M_i, \end{cases} \tag{4.1}$$

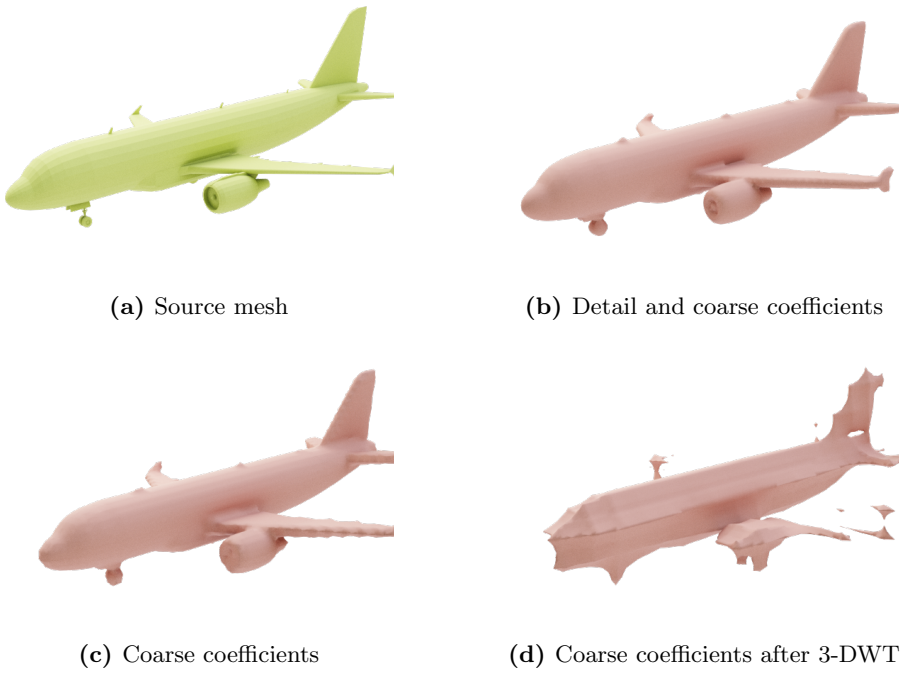
with d being a suitable point-surface distance. When points in the field are far away from the shape surface, their value becomes large and unstable (i.e., with increasing deviation). These points are often identified as irrelevant by the model during training and contribute the least to the prediction of the final shape. To maintain smoothness and avoid discontinuities, [64] truncates the distance values in the SDF to the range $[-0.1, 0.1]$. While this improves the learning, it does not emphasize accurate predictions along the shapes' contours. Unlike them, we introduce a limiting function

$$g(x) = \frac{1}{2} \tanh\left(f_{M_i}(x)\right) - \frac{1}{2}. \tag{4.2}$$

Through this bounding continuous function, SDF values far from the object's surface are constrained to approach zero. Since wavelets, applied in the following step, are inherently sensitive to local variations, this ensures that the resulting coefficients are more responsive to the shape boundaries rather than distant regions. As a result, with distant regions approaching zero, fewer wavelet coefficients are required to encode the surface, leading to a more efficient shape representation.

Pre-encoding with wavelet transformation Furthermore, we apply the 3D discrete wavelet transformation (DWT) on the preprocessed SDFs to extract localized features and patterns from the data. This step is essential for efficiently encoding localized spatial details while reducing redundancy in the representation. DWT can be considered as a particular type of convolutional layer with specific filter banks for extracting multi-scale features [120, 52]. Here, selecting an appropriate wavelet filter is crucial. While Haar wavelet is a popular choice for its simplicity, using it to encode smooth and continuous signals such as the SDF may introduce some voxelization artifacts [64]. For the data representation chosen in this approach, the `Coiflet` wavelet [15] is a suitable choice because, empirically, it provides a good balance between performance (in preserving important geometric features) and reconstruction accuracy. The application of the 3D DWT on each S_i results in two sets of coefficients, i.e., one low-frequency coarse coefficient $C_i \in \mathbb{R}^{130^3}$ (referred to as DWT coefficients in this chapter) and high-frequency detail coefficients $D_i \in \mathbb{R}^{7\times 130^3}$.

For the subsequent process, we drop the detail coefficients, as a single-level wavelet decomposition retains sufficient information in the coarse coefficients for reconstruction. The difference between Figure 4.3b and Figure 4.3c demonstrates this effect. However, training a generative model directly on these coefficients ($C_i \in \mathbb{R}^{130^3}$) remains computationally demanding. To mitigate this, [64] applied hierarchical wavelet transformation for further compression. In such cases, discarding detail coefficients is no longer viable, shown in Figure 4.3d, as they must be further predicted from the coarse coefficients.



(a) Source mesh

(b) Detail and coarse coefficients

(c) Coarse coefficients

(d) Coarse coefficients after 3-DWT

Figure 4.3: Effect of Wavelet Decomposition levels and the Dropping of High-Frequency Coefficients on Plane Mesh Reconstruction. (a) Original plane mesh; (b) Reconstructed plane after applying wavelet decomposition and reconstruction using all coefficients (both coarse coefficients and fine coefficients); (c) Reconstruction after one single-level wavelet decomposition level, keeping only low-frequency coefficients (coarse coefficients) and setting others to zero; (d) Reconstruction after **three** levels of wavelet decomposition, keeping only low-frequency coefficients (coarse coefficients) and setting others to zero.

Dataset decomposition with SVD We propose to encode the DWT coefficients with SVD, which can be conducted through the following steps: (1) for n meshes, flatten their DWT coefficients, denoted by $C_i \in \mathbb{R}^m, m = 130^3, i = 1, \ldots, n$, (2) stack

the coefficients, resulting in a matrix $X = [C_1, C_2, ..., C_n] \in \mathbb{R}^{n \times m}$, and (3) perform singular value decomposition (SVD) on X. Let $r \leq \min(m, n)$ denote the rank of X, the compact SVD is $X = U\Sigma V^\top$, where $U \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{m \times r}$ are semi-unitary matrices and $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ contains the singular values on it diagonal. We arrange the singular values in descending order, i.e., $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, for later truncation. Here, we address:

1. For the geometric information of each mesh (stored in rows of X), the rows of U compress it drastically when $m \gg n \geq r$, e.g., high-resolution meshes are used on a small 3D shape dataset with sample number n.

2. SVD can be used to obtain a low-dimensional rank approximation of X by keeping only the $d < r$ largest singular values, where the approximation error depends on the spectrum of X. Let $\widehat{X}_d = U_d \Sigma_d V_d^\top$, where $\Sigma_d = \mathrm{diag}(\sigma_1, \ldots, \sigma_d)$ and $U_d \in \mathbb{R}^{n \times d}, V_d \in \mathbb{R}^{m \times d}$ obtained from U and V by only keeping the first d columns, respectively. We have the approximation error: $\|X - \widehat{X}_d\|_{\mathrm{F}}^2 = \sum_{i=d+1}^{r} \sigma_i^2$. SVD achieves the optimal approximation error by the Eckart–Young–Mirsky theorem [103]. If the spectrum of X decays rapidly, then we can safely truncate off a large fraction of singular values and keep the error small.

In practice, we maximize computational efficiency by decreasing the approximation rank d to the lowest value, where the reconstructed mesh shows no visually recognizable error, and measured infinity error should be acceptable. See Figure 4.5 for an example of selecting d for the airplane dataset in ShapeNet [26]. Intuitively, each row of U_d is the low-rank representation of a mesh shape, and its corresponding singular value reflects its frequency in the entire data set. Thereby, we decide to define the *spectral features* of the mesh shape by scaling each row of U_d with its singular value, i.e., rows of matrix $U_d\Sigma_d$. We shall denote by α a row of $U_d\Sigma_d$. Also, the column space of V_d is a subspace of the original wavelet coefficients, serving as a "dictionary" or "basis" for representing the DWT data. Thus, we define V_d as *DWT basis* in this chapter. In this setup, each (flattened) DWT coefficient is approximated by

$$\widehat{C}_i = \alpha V_d^\top,$$

where $\widehat{C}_i = C_i$ iff. $d = r$. In the sequel, we shall train a generative model on the spectral feature α, and a new shape can be created by sampling a new α from the model and reconstructing the DWT coefficients with the matrix V_d. Note that the space where α lies maintains the same smoothness of the SDF space, as only

continuous functions are applied to those.

4.2.2 Spectral Domain Diffusion

Diffusion model architecture For the diffusion model, we adapt the denoising diffusion probabilistic model (DDPM) architecture proposed by [61]. However, we replace the 2D convolutional layers with fully connected dense layers to better handle the 1D sequence nature of the spectral features α. This modification is motivated by the fact that the values in α are ordered according to the weights of the corresponding eigenvectors, and dense layers are better suited to capture global patterns in such structured data. For training and inference, we utilize a score-matching generative model, specifically the elucidating diffusion model (EDM) [71, 167], due to its fast sampling and efficient training capabilities. The diffusion model is trained to predict the spectral features α from noisy inputs, enabling the generation of new α values that can be used to reconstruct novel meshes.

Training objective The spectral features α are normalized to the scale $[-3, 3]$ and used to train the diffusion model. The diffusion model is trained using a composite loss function designed to ensure accurate prediction of the spectral features α while preserving the structural integrity of the generated shapes. The loss function is defined as

$$L = (1 - \lambda)L_\alpha + \lambda L_C, \tag{4.3}$$

where L_α ensures the model accurately predicts the spectral features α, and L_C acts as a regularization term to incorporate the precomputed DWT basis $V^\top$. The individual loss terms are defined as

$$L_\alpha = \mathbb{E}_{\sigma,\alpha,\eta} \| D_\theta(\alpha + \eta; \sigma) - \alpha \|_2^2, \tag{4.4}$$

$$L_C = \mathbb{E}_{\sigma,\alpha,\eta} \| D_\theta((\alpha + \eta) \cdot V^\top; \sigma) - \alpha \cdot V^\top \|_2^2, \tag{4.5}$$

where, $\ln(\sigma) \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$, $\eta \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$, D_θ is the implemented neural denoiser and $V^\top$ is the DWT basis obtained from SVD. While $V^\top$ does not participate directly in the training process, it serves as a critical multiplication factor for shape reconstruction. The regularization term L_C ensures that the generated α values, when combined with $V^\top$, produce coherent and structurally valid low-frequency coarse coefficients, as demonstrated in Section 4.3.3.

4.2.3 Mesh Generation

Our proposed pipeline yields a set of basis $V^\top$ for storing shape elements, and features α that serve as "weights" that can be combined with the basis and form new shapes. Thus, at the beginning of our pipeline, we introduce clustering to maximize the shape elements obtained in the basis $V^\top$, ensuring that when the model generates new spectral features, they can be leveraged to explore the shape space of the larger dataset.

During generation, the trained diffusion model produces new α values from random noisy inputs. These generated α values are denormalized with pre-stored scaling parameters ($\alpha_{min} \in \mathbb{R}^d$ and $\alpha_{max} \in \mathbb{R}^d$ estimated from source α among all training samples) and multiplied with the pre-stored $V^\top$ to reconstruct the low-frequency coarse coefficients C_i. Finally, the inverse DWT and marching cube algorithms are applied to C_i to generate a new mesh M_i.

4.3 Experiments

4.3.1 Experimental Dataset and Setup

Evaluation is conducted on ShapeNet [26] to compare with previous SOTA models. We focus on the airplane and chair categories from ShapeNet. These categories are chosen due to their geometric complexity and relevance in benchmarking generative models for 3D surface reconstruction. For mesh decomposition, we consistently select a sample size of $n = 1k$ for each dataset and set the reduced dimensionality to $d = 512$ as the default configuration. An experiment on tuning d is presented in Section 4.3.3. Training is conducted on a single NVIDIA A10G GPU with a batch size of 32 and a learning rate of 5×10^{-4} for $100k$ steps. Using the EDM training pipeline [71], we retain the original hyperparameters: $P_{\text{mean}} = -1.2$ and $P_{\text{std}} = 1.2$. For inference, we set $\sigma_{\text{min}} = 0.002$, $\sigma_{\text{max}} = 80$, $\rho = 5$, and $T = 64$.

4.3.2 Evaluation Metrics

Evaluating the unconditional synthesis of 3D shapes is a different challenge different than the one we introduced in Chapter 3 due to the absence of direct ground truth correspondence. To address this, we use established metrics consistent with previous works which include:

Chamfer Distance (CD) : This metric measures the similarity between two point clouds. It computes the average distance between each point in one point cloud and its closest point in the other point cloud. The Chamfer Distance for two point clouds P and Q, respectively, is defined as:

$$\mathrm{CD}(P, Q) = \frac{1}{|P|} \sum_{p \in P} \min_{q \in Q} \|p - q\|^2 + \frac{1}{|Q|} \sum_{q \in Q} \min_{p \in P} \|q - p\|^2, \tag{4.6}$$

where p and q are points in the point clouds P and Q, respectively, and $\|\cdot\|$ denotes the Euclidean distance between two points.

Minimum Matching Distance (MMD) : This metric measures the mean Chamfer Distance between a sample in the test dataset and its closest sample in the generated dataset. Lower values indicate better performance. The MMD is given by:

$$\mathrm{MMD}(P, Q) = \frac{1}{|P|} \sum_{p \in P} \min_{q \in Q} \mathrm{CD}(p, q), \tag{4.7}$$

where P and Q represent the point clouds in the test and generated datasets, respectively.

Coverage (COV) : Coverage measures the percentage of test data that has at least one corresponding match in the generated data. After assigning every generated sample to its closest test data based on Chamfer Distance, the Coverage is computed as:

$$\mathrm{COV}(P, Q) = \frac{1}{|Q|} \sum_{q \in Q} \mathbb{I} \left(\min_{p \in P} \mathrm{CD}(q, p) \leq \epsilon \right), \tag{4.8}$$

where $\mathbb{I}$ is the indicator function, which is 1 if the condition holds, and 0 otherwise. ϵ is a predefined threshold for matching.

1-Nearest-Neighbor Accuracy (1-NNA) : This metric computes the accuracy of the nearest neighbor search by measuring the percentage of generated point clouds that match the nearest ground truth point cloud. Ideally, the accuracy should be around 50%, indicating that the generated data is similar to the test data. The accuracy is computed as:

$$\text{1-NNA}(P, Q) = \frac{1}{|Q|} \sum_{q \in Q} \mathbb{I} \left(\min_{p \in P} \mathrm{CD}(q, p) \leq \min_{q' \in Q} \mathrm{CD}(q', p) \right). \tag{4.9}$$

Jensen-Shannon Divergence (JSD) : JSD measures the divergence between the probability distributions of two datasets. In our context, we convert the point clouds into discrete voxel grids and compute the divergence between the test and generated data distributions. The JSD between two distributions P and Q is given by:

$$\mathrm{JSD}(P, Q) = \frac{1}{2} \left(D_{\mathrm{KL}}(P \| M) + D_{\mathrm{KL}}(Q \| M) \right), \tag{4.10}$$

where $M = \frac{1}{2}(P + Q)$, and $D_{\mathrm{KL}}(P \| Q)$ is the Kullback-Leibler divergence between distributions P and Q, defined as:

$$D_{\mathrm{KL}}(P \| Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)}. \tag{4.11}$$

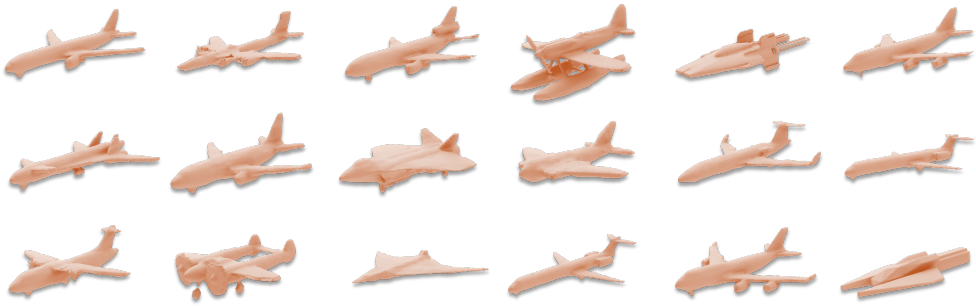
These metrics ensure a comprehensive evaluation of the generated meshes in both the 3D space and visual quality, addressing various aspects of geometric accuracy, distributional similarity, and perceptual quality.

4.3.3 Ablation Study

Dimension of the spectral space Reducing the dimensionality of the spectral space d directly contributes to reducing data size, model size, and computational costs at the expense of reconstruction accuracy. To investigate this trade-off, we conducted a hyperparameter tuning experiment, varying d across several values. The goal is to identify an optimal dimension that balances these competing factors while achieving strong overall performance. Based on the results, we select $d = 512$ as the most suitable configuration for our method. Figure 4.5 illustrates the effect of different truncation levels on various evaluation metrics, while Table 4.1 provides a quantitative comparison of performance across various dimensions ($d \leq n = 1\,000$). Metrics used for evaluation include minimum matching distance(MMD), Jensen-Shannon divergence(JSD), coverage(COV), and the L_2-norm reconstruction error.

At $d = 512$, our method demonstrates a balanced performance across metrics, achieving the best trade-off between reconstruction accuracy and generative diversity. Specifically, the L_2-norm reconstruction error improves significantly compared to $d = 256$, dropping from 1.54×10^{-6} to 5.82×10^{-7}. While increasing d to 786 or 1 000 further reduces reconstruction error, this comes at the cost of higher computational demands and a marginal decrease in generative diversity metrics such as JSD and COV. Dimension $d = 512$ strikes an effective balance, delivering strong coverage

(a) Examples of 3D chairs.



(b) Examples of 3D airplanes.

Figure 4.4: Examples of 3D meshes generated by our SpoDify for qualitative evaluation.

(64.71%) and reconstruction quality without unnecessarily increasing model size or computational overhead.

Training loss Several techniques have been introduced in our method, i.e., the loss regularization L_C (Equation (4.5)) and the limiting function $g(\cdot)$ (Equation (4.2)). Thus, we evaluate the impact of each feature through an ablation study. Here, we consider the following ablated models:

(i) Ablation (w/o L_C): Removing the loss regularization of wavelet coefficients L_C;

(ii) Ablation (w/o L_α): Removing the loss term of spectral feature L_α;

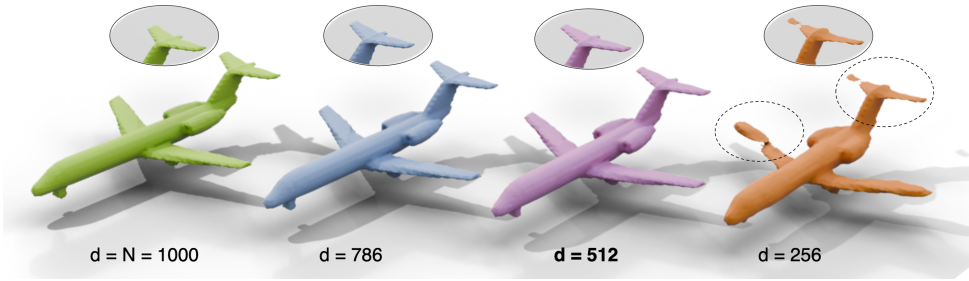


Figure 4.5: Truncation level. Changing the reduced length d of rows in α can impact the visual quality of final results and the computational power required to train the generative model. We notice that by truncating the row length until $d = 512$, no significant visual artifacts are brought to the reconstructed meshes, whereas with $d = 256$, reconstructed meshes show structural errors.

Table 4.1: Ablation study on the dimension of the spectral space out of $n = 1\,000$ possible.

d	MMD↓	JSD↓	COV(%) ↑	L2 Recons↓
256	1.68	3.28	50.06	1.54E-6
512	1.7	3.1	64.71	5.82E-7
786	1.81	3.05	69.55	1.49E-7
1 000	1.76	2.9	61.05	0

(iii) Ablation (w/o $g(\cdot), L_C$): Deactivating the limiting function on SDF $g(\cdot)$ and removing the loss regularization of wavelet coefficients L_C;

From the results presented in Table 4.2, we have the following observations: (1) The full model, which includes all three components (L_C, L_α, and $g(\cdot)$), achieves in average the best performance; (2) Removing the wavelet coefficient regularization (L_C) results in a slight increase in MMD and JSD, as well as a drop in COV, indicating that L_C helps in improving coverage and reduces discrepancy; (3) Removing the spectral-

Table 4.2: Ablation study on training and regularization losses with the airplane Dataset from ShapeNet. Configurations are formed by different combinations of parameters: $V^\top$ indicates the inclusion of regularization loss with respect to the wavelet domain, g denotes the use of a limiting function on SDF values, and α specifies the inclusion of loss with respect to the spectral space.

	$g(\cdot)$	L_C	L_α	MMD↓	JSD↓	COV(%) ↑
SpoDify	✓	✓	✓	1.7	**3.1**	**64.71**
Ablation (i)	✓		✓	1.776	3.27	61.05
Ablation (ii)	✓	✓		1.73	3.47	45.17
Ablation (iii)			✓	**1.64**	3.15	56.1

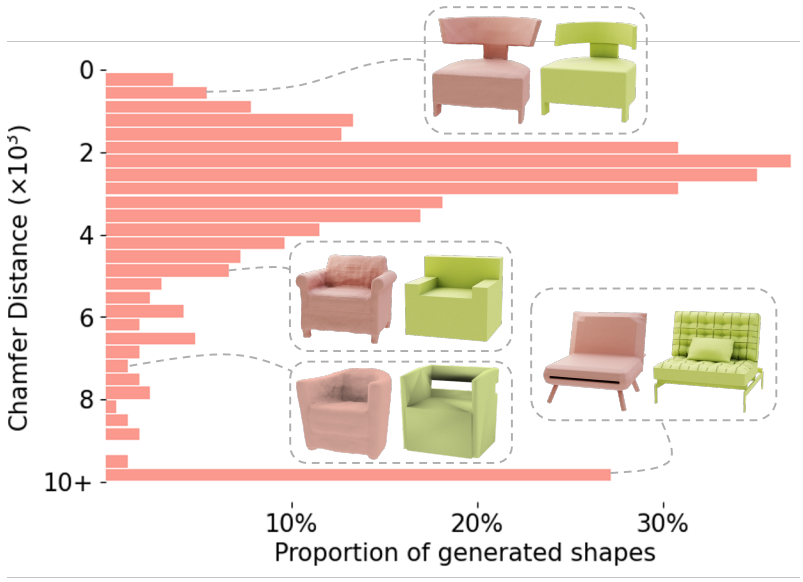


Figure 4.6: Shape novelty analysis on ShapeNet [26] chair category. We plot the distribution of 500 chair samples generated by our method and their closeness to the training dataset. Additionally, we display samples to visualize the similarity of various CD values, where *green chairs* are from the training dataset and *red chairs* are generated.

feature regularization loss (L_α) leads to a notable increase in JSD and a significant decrease in COV, confirming that L_α is key to maintaining the diversity and quality of the generated shapes; (4) When both $g(\cdot)$ and L_C are removed, the model performs better in terms of MMD compared to removing L_α alone, but it still lagged behind the full model in JSD and COV. This suggests that while the limiting function $g(\cdot)$ and the wavelet regularization term help in coverage, they do not fully compensate for the loss of spectral regularization.

In conclusion, our findings emphasize the importance of all three components in achieving the best performance. The wavelet coefficient regularization L_C and the limitation function $g(\cdot)$ contribute to model stability and coverage, while the spectral regularization loss L_α is essential for maintaining high-quality outputs and preventing overfitting. The full model, with all components, strikes the optimal balance between MMD, JSD, and COV, demonstrating the effectiveness of our design choices.

Table 4.3: Quantitative evaluation of our proposed pipeline and current 3D shape generators. Metrics are computed over ShapeNet classes airplane and chair using the Chamfer distance (CD). NWD provided generated meshes for evaluation, while UDIff did not release its ShapeNet meshes. Due to computational constraints, we did not retrain UDiFF to compute JSD, resulting in missing values.

Category	Method	1-NNA ↓	MMD $\times 10^{-3}$ ↓	COV (%) ↑	JSD ↓
Airplane	NWD [64]	97.24	1.69	71.2	2.9
	UDiFF [200]	**74.48**	**0.315**	**64.77**	-
	Ours	97.98	1.7	64.71	3.1
Chair	NWD [64]	53.4	1.180	45.19	0.027
	UDiFF [200]	65.96	1.167	52.58	-
	Ours	**46.69**	**1.114**	**53.5**	**0.022**

Table 4.4: Efficiency comparison. Our approach can use less than 90% GPU compared to TetraDiffusion [69] and be trained in a few hours compared to days needed for NWD [64] and UDiff [200]. For NWD and UDiFF, the (+12) indicates additional GPU memory used for training the detail predictor network, separate from the main network responsible for global coefficient training.

Method	**Representation**		**Training**			**Inference**
	Dimension	Compression rate	GPU (GB)	Speed (it/s)	Duration (h)	Speed (s/obj)
NWD [64]	256^3	$(46/256)^3 = 5.8‰$	5.3 (+12)	15.3	84	3.6
TetraDiffusion [69]	192^3	$(192/192)^3 = 100\%$	78.2	0.3	-	33.3
UDiFF [200]	256^3	$(46/256)^3 = 5.8‰$	4.5 (+12)	4.89	84	3.4
Our SpoDify	256^3	$512/256^3 = 0.03‰$	7.2	100	3	3.3

4.3.4 Results

Qualitative comparison To evaluate the performance of our method, SpoDify, we first conduct a qualitative evaluation against several leading 3D mesh generation models while focusing later on the trade-off between performance and model complexity. We display the qualitative comparison in Figure 4.7. The evaluation is performed on the ShapeNet categories airplane and chair, where we compare the results of SpoDify to those of NWD [64], UDiFF [200], and 3DShape2VecSet [195]. Our primary objective is not to surpass the SOTA generation models but to demonstrate that our method can achieve comparable or near-SOTA performance with a significantly less complex architecture and less computational power. While being able to fully capture the diversity of the dataset, our model maintains the structural properties of the underlying dataset. We notice that generated samples rarely present floating material or other structural artifacts that are physically incoherent in practice.

(a) 3DShape2VecSet [195]

(b) NWD [64]

(c) Our SpoDify

Figure 4.7: Qualitative comparison of chairs generated by different methods: (a) 3DShape2VecSet [195], (b) NWD [64], and (c) our SpoDify. The results from 3DShape2VecSet show a significant amount of artifacts, e.g., floating material and incomplete component; the results from NWD perform better in results plausibility but tend to be simple designs; meanwhile, our SpoDify is able to generate complicated designs and meantaining a high level of plausibility.

Quantitative comparison In the further quantitative comparison with SOTA methods, as results shown in Table 4.3, our SpoDify performs competitively across various metrics, including 1-nearest-neighbor accuracy (1-NNA), minimum matching distance (MMD), coverage (COV), and Jensen-Shannon divergence (JSD), indicating that it can generate high-quality 3D shapes similar to those produced by more complex models. In particular, our model outperforms the state of the art on the chair dataset. Moreover, COV values are consistently large, as the implicit representation adopted ensures the capability of reconstructing training samples.

Shape novelty analysis This study examines the capacity of our proposed method to generate novel shapes with respect to the ones in the training dataset. To do so, we build on the work by [152] and synthesize 500 chairs using SpoDify. All shapes (generated and from the training set) are preprocessed through normalization into a unit cube, ensuring a standardized and equitable framework for comparison. We employ Chamfer Distance (CD) as the metric to quantify the similarity between shapes. We use CD to determine the sample in the training dataset that is the most similar to the

generated chair. From the CD distribution shown in Figure 4.6, we observe that our model generates not only shapes that closely match those in the training set (low CD) but also produces realistic shapes that differ significantly from the training set shapes (high CD).

Efficiency comparison Table 4.4 highlights the comparative efficiency of various methods regarding training and inference requirements. Additionally, it reports the compression ratio between the input mesh and its corresponding representation used as input to the training model. Our proposed method demonstrates outstanding improvements in training speed and duration due to the impressive compression rate while requiring minimum GPU usage, while achieving comparable state-of-the-art results. Specifically, our method achieves the fastest training speed at 100 iterations per second (it/s), thanks to its streamlined model backbone, which significantly outpaces competing methods such as NWD [64] (15.3 it/s) and UDiFF [200] (4.89 it/s). This acceleration reduces the time required for large-scale training scenarios, making it feasible to handle extensive datasets without excessive computational cost. For inference, our method achieves a speed of 3.3 seconds per object, marginally surpassing UDiFF (3.4 s/obj) and vastly outperforming TetraDiffusion [69] (33.3 s/obj).

Moreover, our model leverages an efficient data compression strategy, achieving a compression rate of $512/256^3 = 0.03‰$, which contributes to reduced memory overhead during processing. Meanwhile, 3DShape2VecSet [195], while effective in certain scenarios, is hindered by the exceptionally large size of its data representation—approximately 300 GB—making it challenging to download and impractical to evaluate comprehensively under typical resource constraints.

4.4 Conclusion

Unlike prior works that rely on training to store shape information within the autoencoder parameters, our approach introduces a novel application of singular value decomposition to extract two components from the source data: (1) spectral features, a compact latent representation utilized to train the generative model, and (2) the basis, which provides foundational information for data reconstruction and is preserved during decomposition for reuse during the generation process. We are the first work to use the outputs of singular value decomposition as inputs for training generative models, we demonstrate that its outputs (spectral features and basis) can be effectively leveraged for generative modeling, enabling the synthesis of novel shapes.

Furthermore, our SpoDify not only maintains high-quality reconstruction but also achieves significant improvements in scalability, reducing training times from days to hours and compressing data dimensionality from gigabytes to megabytes. These advancements mark a crucial step toward making 3D generative models practical for handling high-dimensional data, especially when the amount of data is limited, and open up new avenues for research and applications.