

APPENDIX

A IMPLEMENTATION DETAILS

A.1 PARAMETER ABLATION.

Fig. 1 demonstrates Tri-Vectors’ scalability. We sampled 400 shapes from ShapeNet and measured Chamfer Distance (CD) to assess reconstruction performance. Based on these results, we chose **rank=32, resolution=192** (corresponding to $\approx 18\text{K}$ parameters) to balance reconstruction detail and training cost for single-shape experiments. For experiments on Thingi10K and BuildingNet we use **rank=64, resolution=192** (corresponding to $\approx 37\text{K}$ parameters). These settings were selected because they provide a compact representation while still preserving fine geometric details, yielding a favorable trade-off between model simplicity and reconstruction quality.

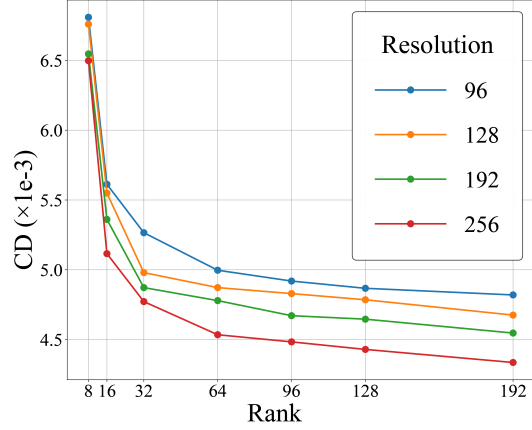


Figure 1: Ablation study of different parameter settings on ShapeNet dataset.

A.2 SHAPE LEARNING.

We optimize Tri-Vectors using 6M points sampled from the shape and their corresponding SDF values. For each optimization step, we randomly sample 16,384 points and use the Adam (Dereich et al., 2024) optimizer with a fixed learning rate of $2\text{e-}3$.

Linear interpolation. We project the query point onto different axes and interpolate the feature using the values of the two nearest points and the projection’s relative position between them. For example, to interpolate the value of a spatial point $p = (x, y, z)$ in the 3D domain, we decompose the process along each axis independently. The 3D space is normalized to $[-0.5, 0.5]^3$, and interpolation is performed on the discrete feature vectors x_r , y_r , and z_r . As illustrated in Fig. 2, for a given axis, e.g., x , the scaled coordinate is computed as $x_{\text{scaled}} = (x + 0.5) \cdot (W - 1)$, where W is the resolution of the vector. The left ($\lfloor x_{\text{scaled}} \rfloor$) and right ($\lceil x_{\text{scaled}} \rceil$) indices and the interpolation weight $\alpha = x_{\text{scaled}} - \lfloor x_{\text{scaled}} \rfloor$ are used to compute $f(x) = (1 - \alpha)f_l + \alpha f_r$, where f_l and f_r are feature values at the respective indices.

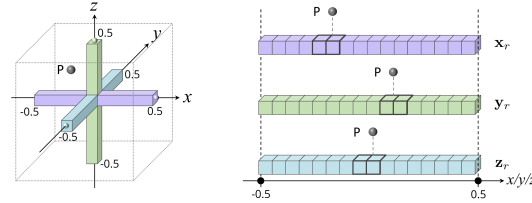


Figure 2: Illustration of the linear interpolation process.

Baselines. We first compared our Tri-Vectors with DenseGrid, Triplane (Shue et al., 2023), NeuralWavelet (Hu et al., 2024), and MSDF (Yariv et al., 2024), as these methods are commonly used in 3D generative applications. We randomly selected 200 models each from the Thingi10K and the ShapeNet datasets. The main distinction between these datasets is model complexity: Thingi10K generally contains more intricate shapes. (1) For DenseGrid, we set the spatial resolution to 128, sample the coordinates of the center points of each grid, and compute the corresponding SDF values. Finally, we reconstruct the surface using the Marching Cubes algorithm. (2) For Triplane, the resolution of each plane is given by $\mathbf{f}_{xy}, \mathbf{f}_{xz}, \mathbf{f}_{yz} \in \mathbb{R}^{128 \times 128 \times 32}$, with a separate MLP assigned to each shape for decoding. (3) For NeuralWavelet, we follow its original wavelet decomposition on the TSDF, applying a multi-scale decomposition up to $J = 3$. For reconstruction, we retain only a coarse grid coefficient with a spatial resolution of 46 and a detail grid coefficient with a spatial resolution of 76, as described in the paper. The shape is then reconstructed using the inverse wavelet transform. (4) For MSDF, we uniformly sample 1024 points on the shape’s surface, representing the MSDF tensor as $X \in \mathbb{R}^{1024 \times (3+1+7^3)}$. The representation is then optimized following the algorithm outlined in the paper.

Then, we compared our Tri-Vectors with StriVec (Gao et al., 2023), InstantNGP (Müller et al., 2022), and DictionaryFields (Chen et al., 2023). We randomly selected 200 models from the Thingi10K dataset. (1) For StriVec, we implement the method following the original hierarchical sparse multi-scale tensor representation and adapt it for SDF compression. We place three levels of local Tri-Vectors (rank = 48) at occupied voxel centers, maintaining the original structure. The voxel resolutions at each level are set to 2, 4, and 8, with corresponding Tri-Vector dimensions of 32, 64, and 128. (**Model size: $\approx 4\text{M}$**). (2) For Instant-NGP, we follow the original SDF compression settings, utilizing a multi-resolution hash grid encoding with 16 levels. Each level stores feature vectors of dimension 2, with a hash table size of 2^{19} and a base resolution of 16. The decoder consists of a density MLP with 2 hidden layers of 64 units. (**Model size: $\approx 12\text{M}$**). (3) For DictionaryFields, we follow the original coefficient-basis factorization settings, representing the scene with multiple resolution levels of basis and coefficient grids. The basis grids employ a periodic coordinate transformation, while the coefficient grids are fixed at a resolution of 32 across all levels. The model uses bilinear interpolation for both basis and coefficient lookup, with a single-layer MLP of 64 hidden units. (**Model size $\approx 5\text{M}$**).

Metrics. To evaluate reconstruction quality, we used Chamfer Distance (CD), Intersection-over-Union (IoU), and F-score. IoU was calculated from the occupancy predictions of 50K query points sampled in normalized 3D space, while CD and F-score were assessed using two point clouds of 50K points sampled from the reconstructed and ground-truth surfaces. We also assessed the parameter efficiency of each method by counting the required parameters for representing a shape. Additionally, MSDF, TP, and Tri-Vectors involve an optimization process, we measured their computation times on the same device and on a single RTX 4090 GPU to ensure a fair comparison.

A.3 SHAPE GENERATION.

For ShapeNet dataset, we set the Tri-Vectors parameters to **18K (rank=32, resolution=192)**, the optimization runs for 5000 iterations, taking approximately 30 seconds for one shape. For Thingi10K and BuildingNet datasets: we set the Tri-Vectors parameters to **37K (rank=64, resolution=192)**, the optimization runs for 8000 iterations, taking approximately 60 seconds for one shape.

For ShapeNet dataset, we follow the dataset split protocol from (Chen & Zhang, 2019) and train a separate model for each category to match TP and NW. Specifically, TP is trained on the *car*, *chair*, and *airplane* categories, while NW is trained on the *table*, *chair*, and *airplane* categories. We choose SiT-B (Ma et al., 2024) as our backbone, training each model on 8 RTX 4090 GPUs with a global batch size of 2048 for 800K iterations, with each model taking approximately three days to train. During the sampling process, we employ an SDE-based method with 1000 steps, where generating Tri-Vectors for a single shape takes around 10 seconds on a single RTX 4090 GPU. We use the AdamW (Loshchilov & Hutter, 2019) optimizer with a fixed learning rate of $1e-4$ and weight decay of 0 and incorporate an EMA (Exponential Moving Average) optimization strategy with a weight decay of 0.9999. For reconstructing triangle mesh from Tri-Vectors, we use a spatial resolution of 256 for querying and reconstruction, with each shape requiring approximately 0.2 seconds to process on a single RTX 4090 GPU. The model weights updated through EMA are used for the data sampling.

Metrics. We use standard metrics to assess our Tri-Vectors models, comparing generated shapes S_g to reference shapes S_r . We compute Chamfer Distance (CD) and Earth Mover’s Distance (EMD) for shape comparison, and from these, derive Maximum Mean Discrepancy (MMD), Coverage (COV), and 1-Nearest Neighbor Accuracy (1-NNA) to evaluate fidelity, diversity, and distributional similarity. For the evaluation, we generate 2000 shapes and compare them to the validation set.

B NOVELTY ANALYSIS

We generate 500 tables using our method, for each table, we retrieve the most similar shape in the training set by Chamfer Distance (CD), then, we plot the distribution of CDs for all retrievals. Fig. 20 illustrates the details of the distribution and some samples of generated and retrieved shapes.

C MORE APPLICATIONS

Tri-Vectors for textured shapes. We extend the Tri-Vectors framework to encode textured shapes by jointly representing geometry and vertex colors. Specifically, vertex colors are modeled as separate fields for the R, G, and B channels, compressed using additional tri-vector sets. A textured shape (Collins et al., 2022) is represented as $\mathcal{S}_{\text{TVC}} = \{\mathcal{S}_{\text{TV}}, \mathcal{S}_{\text{R}}, \mathcal{S}_{\text{G}}, \mathcal{S}_{\text{B}}\}$, where \mathcal{S}_{TV} encodes the geometry, and $\mathcal{S}_{\text{R}}, \mathcal{S}_{\text{G}},$ and \mathcal{S}_{B} represent the corresponding R, G, and B color fields. The texture is treated as a continuous field aligned with the geometry, and each vertex’s color is reconstructed by querying these fields at the corresponding spatial coordinates. For the color of any point in space, we determine its nearest triangle on the ground truth shape and compute the projection of the point onto that triangle. Knowing the colors of the triangle’s vertices from the ground truth, we interpolate the color of the projection point using barycentric coordinates. This interpolated color is considered the color of the corresponding point.

During optimization, the RGB color $c(p_j)$ at point p_j is incorporated into the Tri-Vectors framework, and the combined loss for both geometry and texture reconstruction is formulated as:

$$\mathcal{L}_{\text{TVC}} = \sum_{j=1}^J \|\hat{s}(p_j) - s(p_j)\|_2^2 + \sum_{j=1}^J \|\hat{c}(p_j) - c(p_j)\|_2^2. \quad (1)$$

Our Tri-Vectors compresses both geometry and texture into a compact, efficient format, enabling high-fidelity texture reconstruction alongside accurate geometry recovery. Fig. 21 and Fig. 22 demonstrate more results about textured shape reconstructions using our Tri-Vectors.

In our experiments, we set the rank of the Tri-Vectors representation for geometry to 64 with a resolution of 192, ensuring high-fidelity shape encoding. For each color channel (R, G, and B), we use separate Tri-Vectors representations with a rank of 32 and a resolution of 192. The total number of parameters required to represent a single textured shape is approximately 92K.

Tri-Vectors for deformable shapes. To handle deformable shapes, we extend Tri-Vectors into 4D space, incorporating time as the fourth dimension. This extension enables the efficient encoding of dynamic shapes by capturing both spatial and temporal deformations. The temporal component is represented as an additional 1D vector corresponding to the time parameter t . Each point $p_j(t)$ is thus characterized by its spatial coordinates and the temporal parameter t . The reconstruction of deformable shapes is optimized by minimizing the following loss function:

$$\mathcal{L}_{\text{deform}} = \sum_{j=1}^J \sum_{t=1}^T \|\hat{s}(p_j(t)) - s(p_j(t))\|_2^2. \quad (2)$$

The $\hat{s}(p_j(t))$ represents the predicted SDF value at time t and $s(p_j(t))$ is the ground truth SDF value at time t . Fig. 23 shows the results for deformable shapes using Tri-Vectors.

In our experiments, we extend the original Tri-Vectors representation by incorporating an additional temporal dimension into the geometry encoding. Specifically, we set the rank of the Tri-Vectors representation to 128 with a resolution of 192. The total number of parameters required to represent a single deformable shape in our framework is approximately 98K.

D MORE SHAPE LEARNING RESULTS

Fig. 3 and Fig. 4 show more results about shape reconstruction using Tri-Vectors with different configurations. Fig. 5 demonstrates the reconstruction comparison between our Tri-Vectors and other representations.

E MORE GENERATED RESULTS

Galleries of generated samples. Fig. 8, Fig. 9, Fig. 10, and Fig. 11 gallery the generated results based on our Tri-Vectors on *airplane*, *car*, *chair* and *table* categories of ShapeNet dataset. Fig. 12, Fig. 6, and Fig. 7 showcase the generative model results based on our Tri-Vectors on BuildingNet dataset, highlighting the intricate internal and external structures of the generated buildings.

Comparisons of generated samples with CD query. We present cross-method visual comparisons of generated samples using Chamfer Distance (CD) queries, for each shape, we uniform sample 2048 points on the surface. In Fig. 13, Fig. 15, Fig. 14, and Fig. 16, we use our generated samples to query the closest samples generated by other methods on different categories. In Fig. 17, Fig. 18, and Fig. 19, we use the generated samples by other methods to query the closest samples generated by our method.

This analysis highlights the samples’ differences in geometric fidelity and structural detail between our approach and other methods, demonstrating the effectiveness of Tri-Vectors for the shape generative task.

F LIMITATION AND FUTURE WORK

Limitation. While Tri-Vectors provides a compact and efficient 3D representation, its global CP decomposition prioritizes large-scale structure over fine-grained spatial locality, limiting precise local control in generative tasks. Imposing strong local constraints or detailed feature manipulation remains challenging. A potential solution is adaptive basis refinement, allowing localized adjustments while maintaining model compactness. Additionally, while Tri-Vectors is resolution-independent, optimizing its expressiveness across different scales remains an open challenge.

Feature work. For future work, we seek to regularize and interpret tri-vector components for more explicit shape control. Improving Tri-Vectors’ integration with deep generative models could enable dynamic decomposition refinement during training. Further, extending Tri-Vectors to higher-dimensional tasks, such as multi-object interactions, 4D dynamic scenes, and controllable physics simulations, could expand its applicability while retaining efficiency.

G LLM USAGE STATEMENT

We used large language models only at the paragraph level to correct grammar, wording, and fluency; they were not used to generate substantive scientific content, analyses, experimental design, or interpretations.

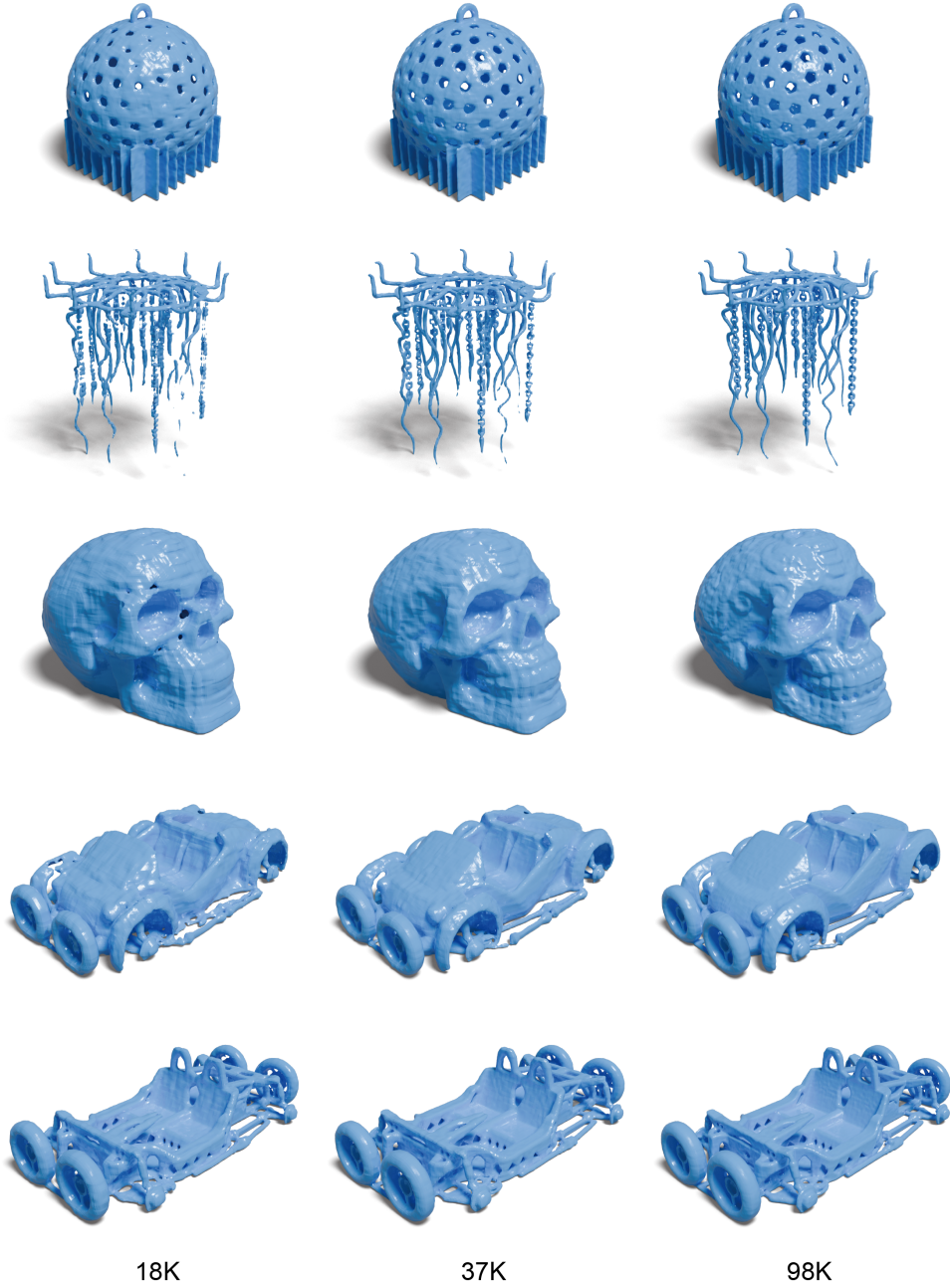


Figure 3: Reconstruction results from Tri-Vectors with different parameter counts (18K: rank=32, resolution=192; 37K: rank=64, resolution=192; 98K: rank=128, resolution=256).

270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323

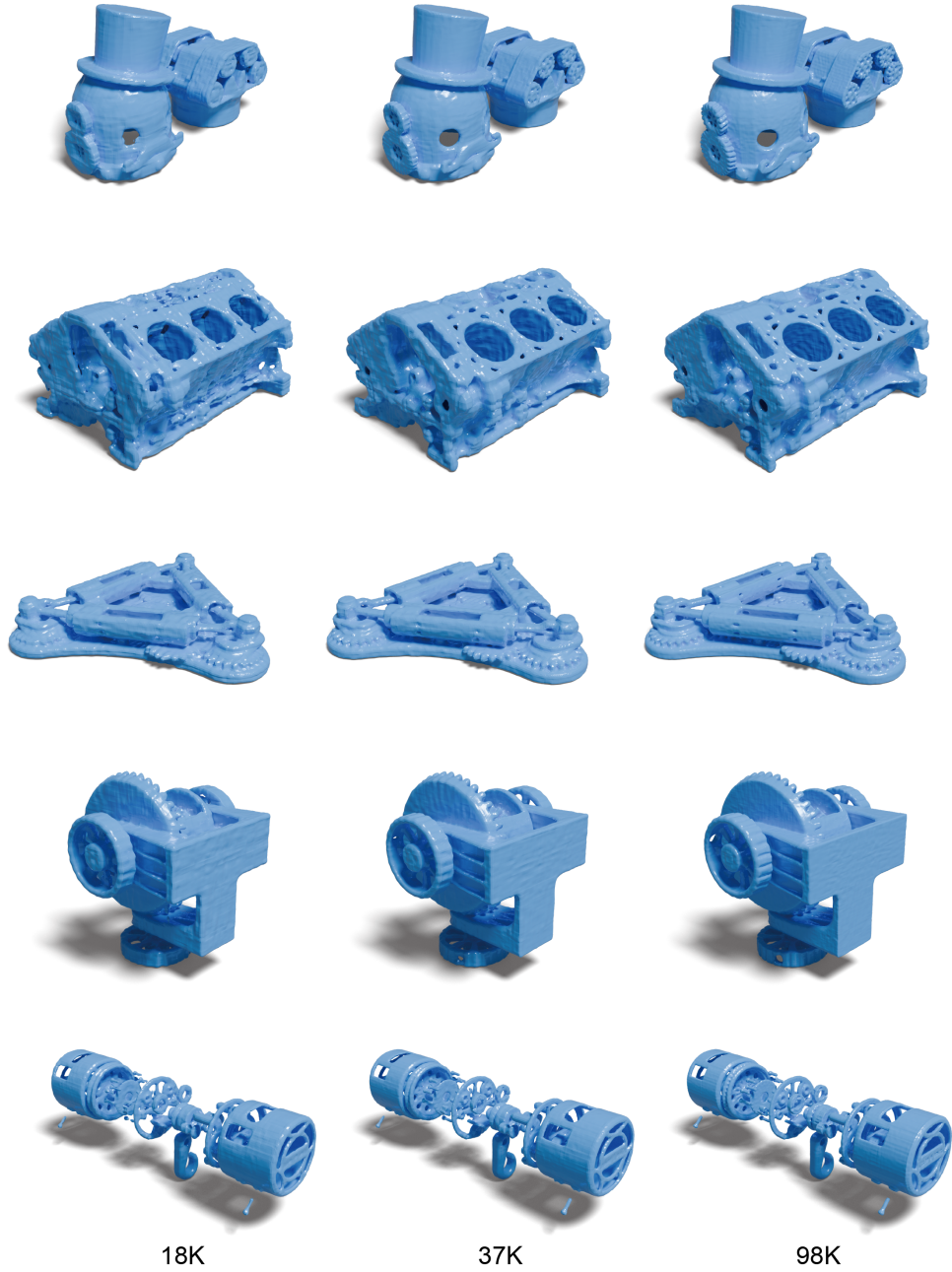


Figure 4: Reconstruction results from Tri-Vectors with different parameter counts (18K: rank=32, resolution=192; 37K: rank=64, resolution=192; 98K: rank=128, resolution=256).

324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377

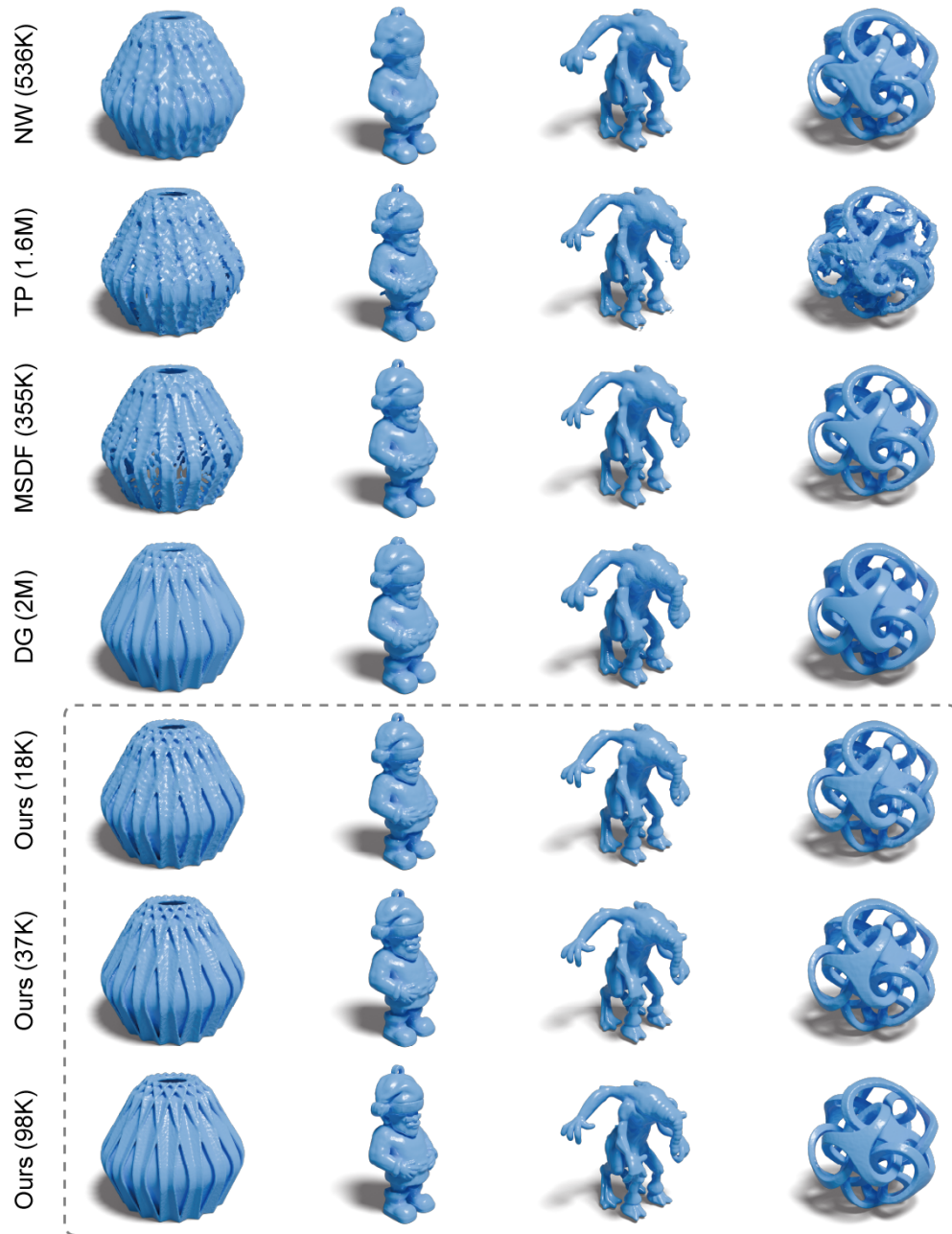


Figure 5: Reconstruction comparison between Tri-Vectors and other methods.

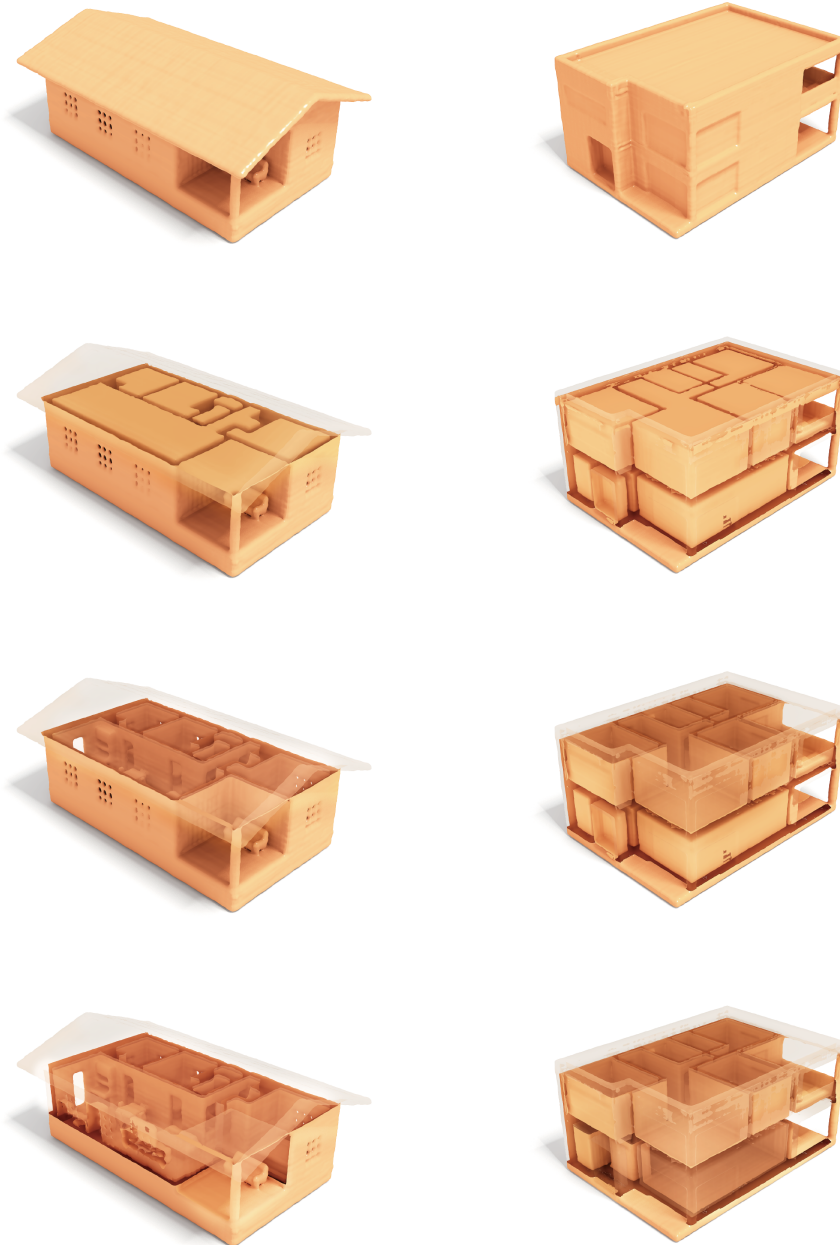


Figure 6: Shapes generated by our method on BuildingNet Selvaraju et al. (2021) dataset, where each shape is represented using 37K parameters (rank=64, resolution=192) with our Tri-Vectors.

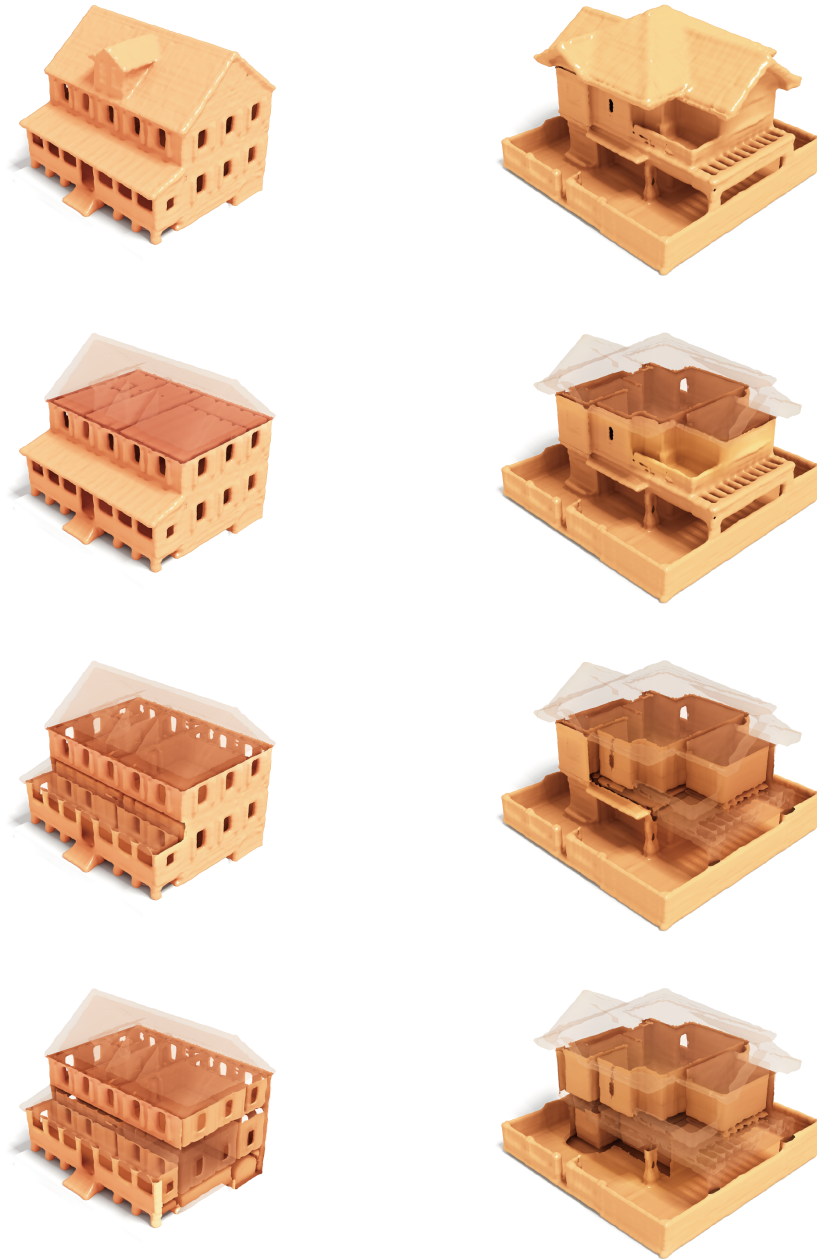


Figure 7: Shapes generated by our method on BuildingNet Selvaraju et al. (2021) dataset, where each shape is represented using 37K parameters (rank=64, resolution=192) with our Tri-Vectors.

486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539

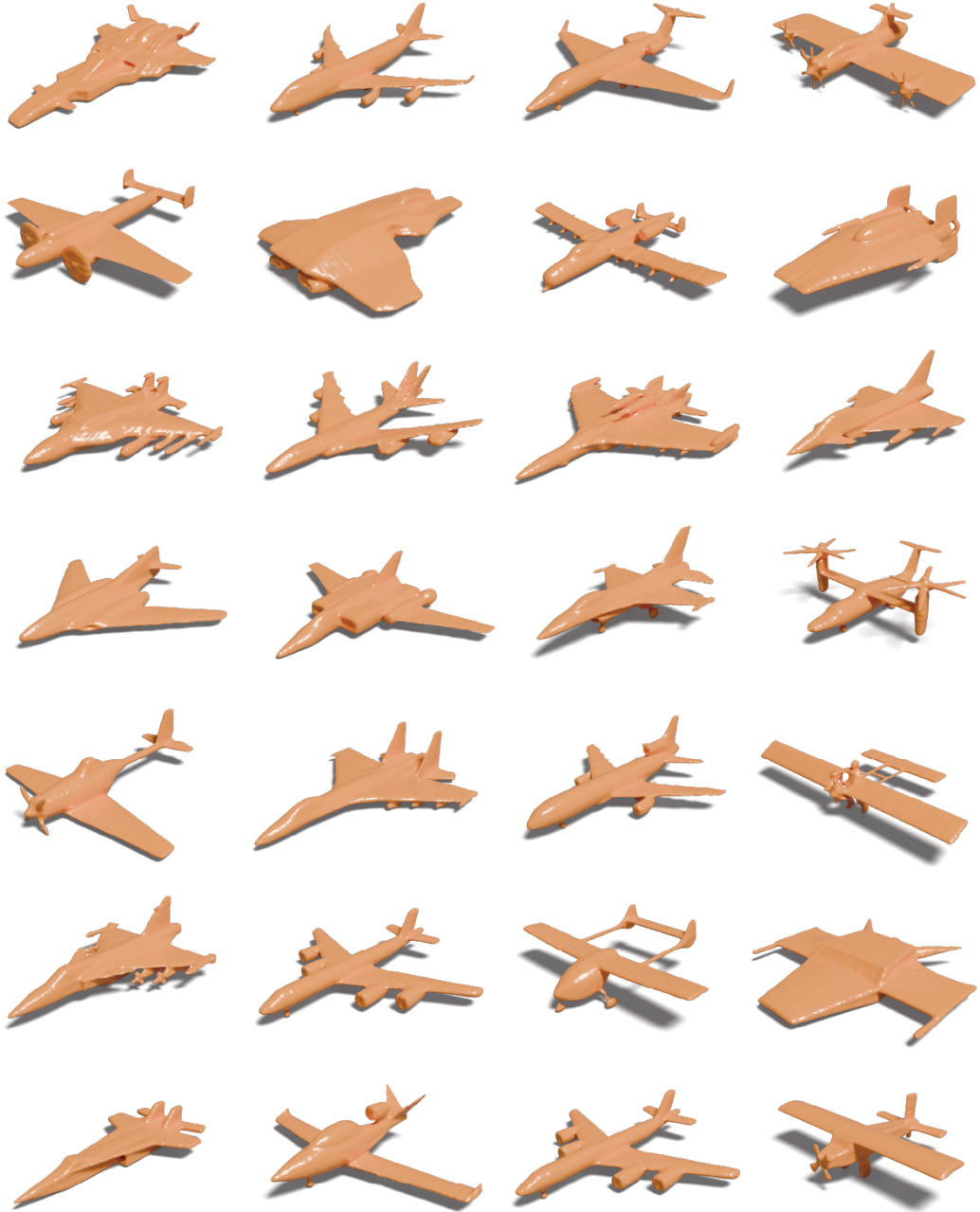


Figure 8: Shapes generated by our method on *airplane* category, where each shape is represented using 18K parameters (rank=32, resolution=192) with our Tri-Vectors.



Figure 9: Shapes generated by our method on *car* category, where each shape is represented using 18K parameters (rank=32, resolution=192) with our Tri-Vectors.

594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647



Figure 10: Shapes generated by our method on *chair* category, where each shape is represented using 18K parameters (rank=32, resolution=192) with our Tri-Vectors.



Figure 11: Shapes generated by our method on *table* category, where each shape is represented using 18K parameters (rank=32, resolution=192) with our Tri-Vectors.

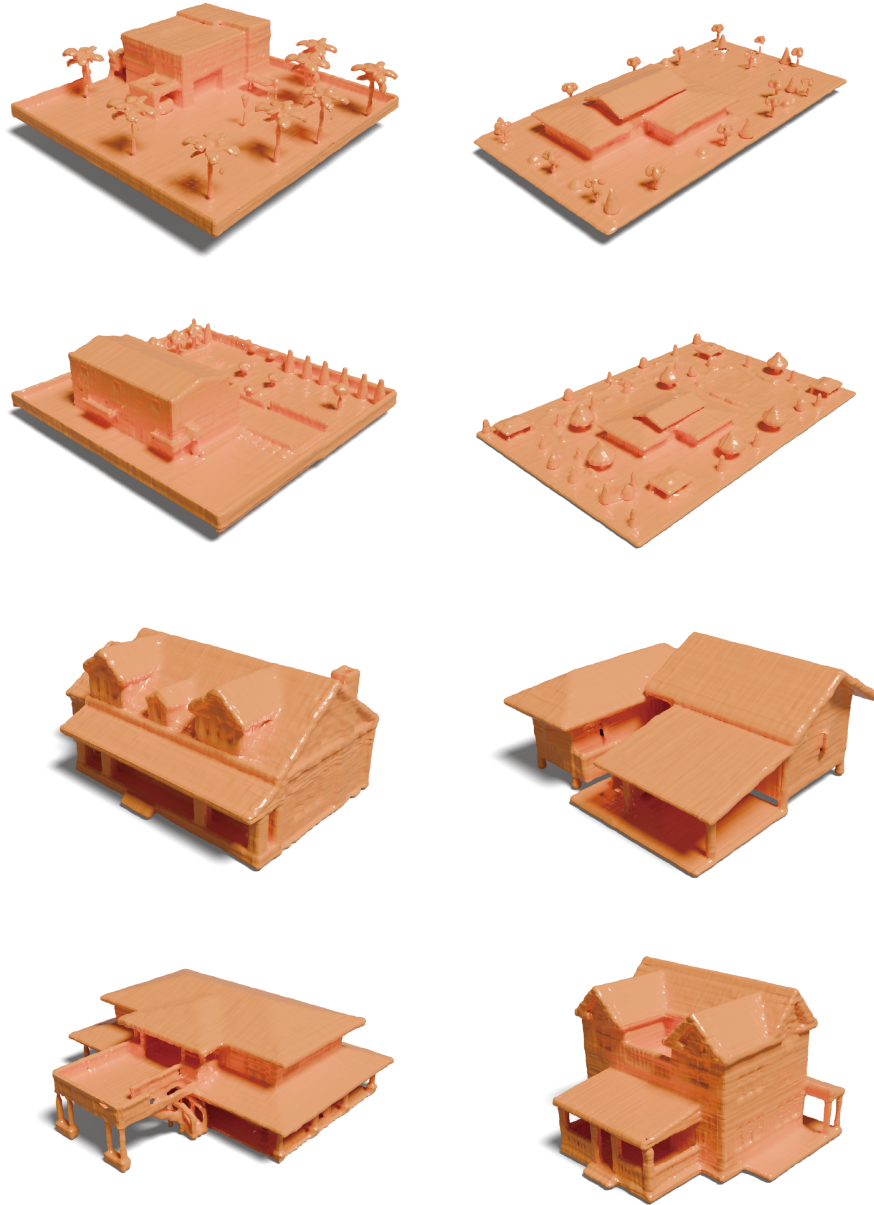


Figure 12: Shapes generated by our method on BuildingNet Selvaraju et al. (2021) dataset, where each shape is represented using 37K parameters (rank=64, resolution=192) with our Tri-Vectors.

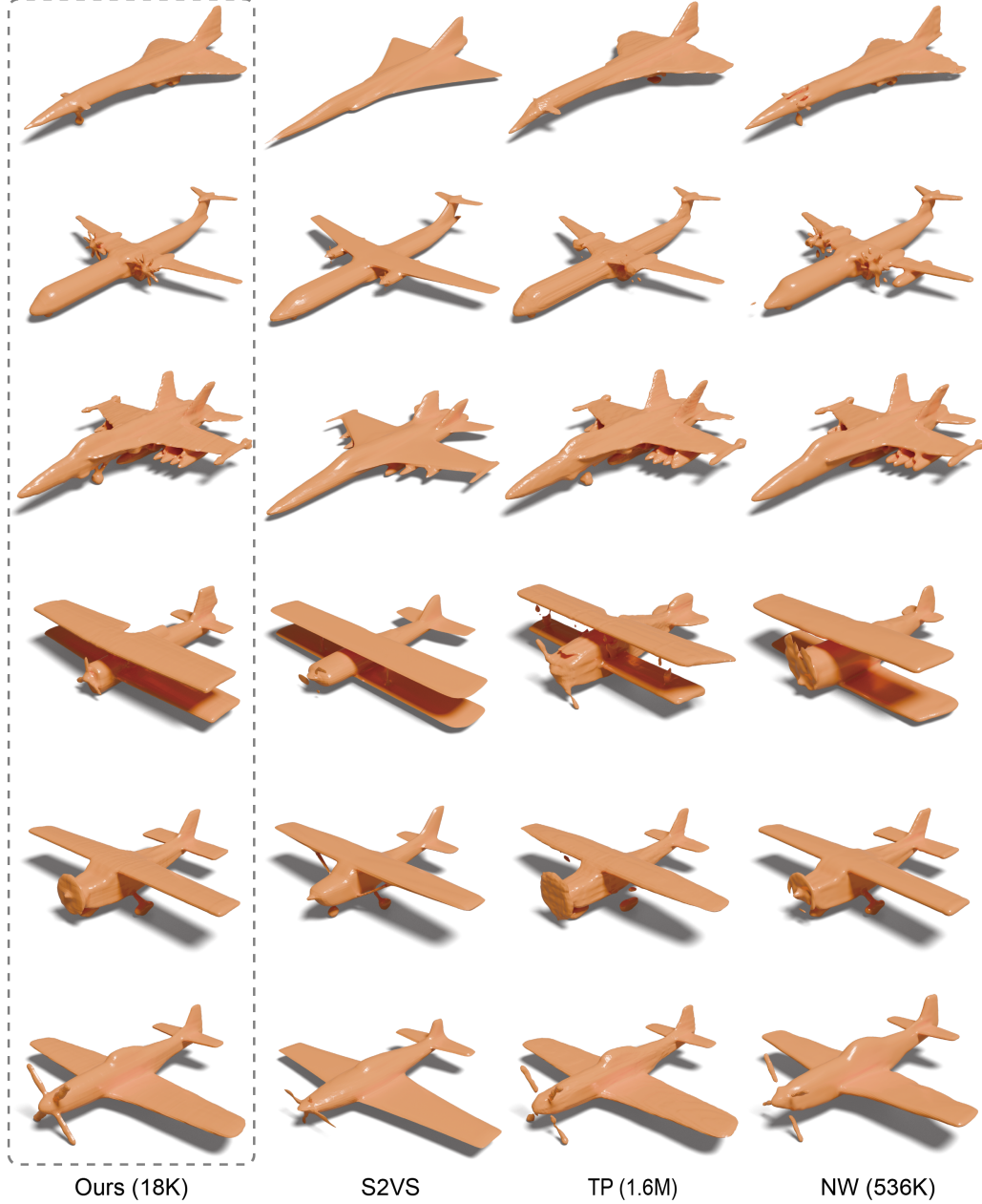


Figure 13: Shapes generated by our method and shapes retrieved (CD query) from S2VS Zhang et al. (2023), TP Shue et al. (2023), and NW Hui et al. (2022) on *airplane* category. The parameter count used for representing each shape is indicated while S2VS is not specified due to its use of a network-based compression mechanism.

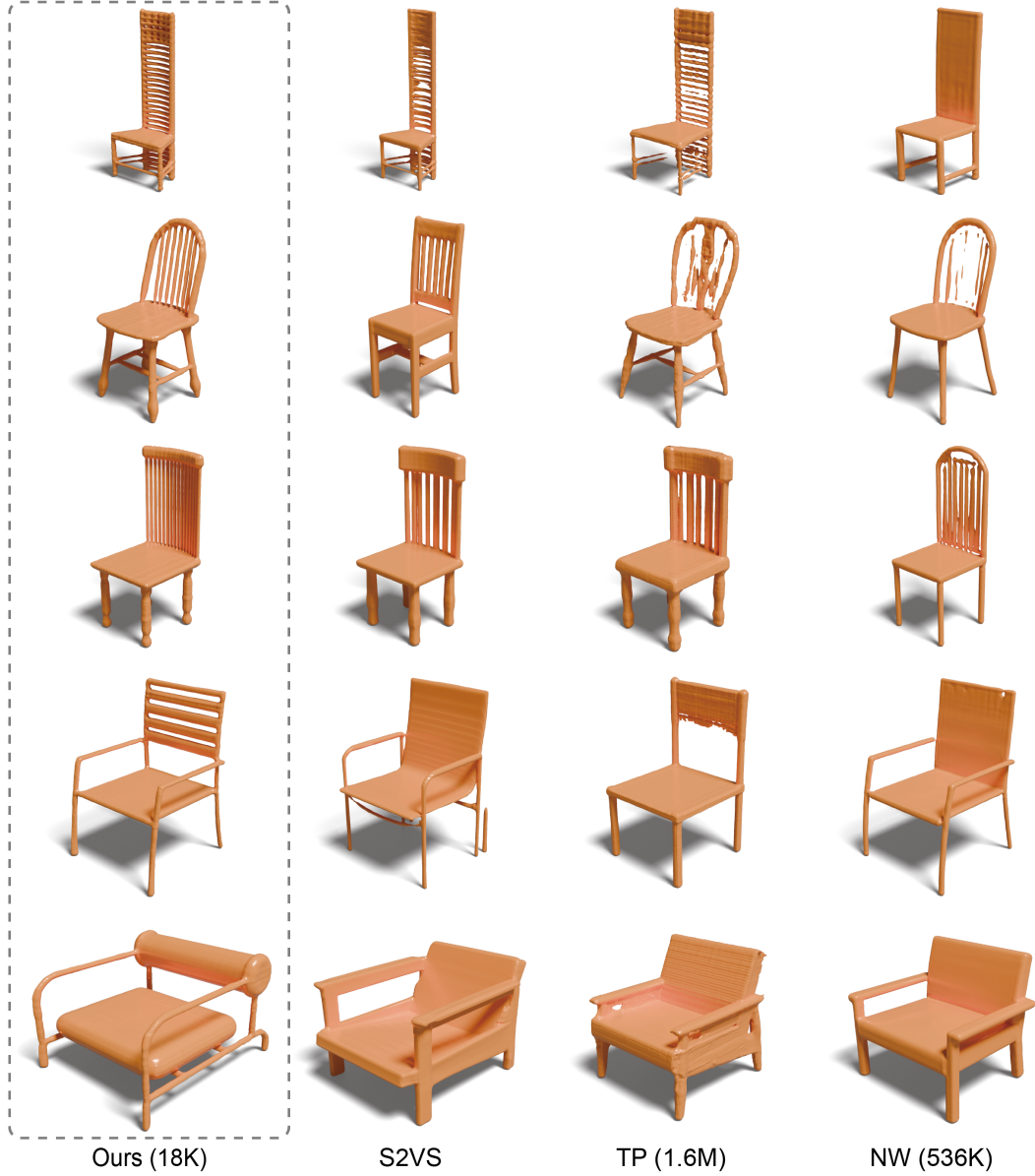


Figure 14: Shapes generated by our method and shapes retrieved (CD query) from S2VS Zhang et al. (2023), TP Shue et al. (2023), and NW Hui et al. (2022) on *chair* category. The parameter count used for representing each shape is indicated while S2VS is not specified due to its use of a network-based compression mechanism.

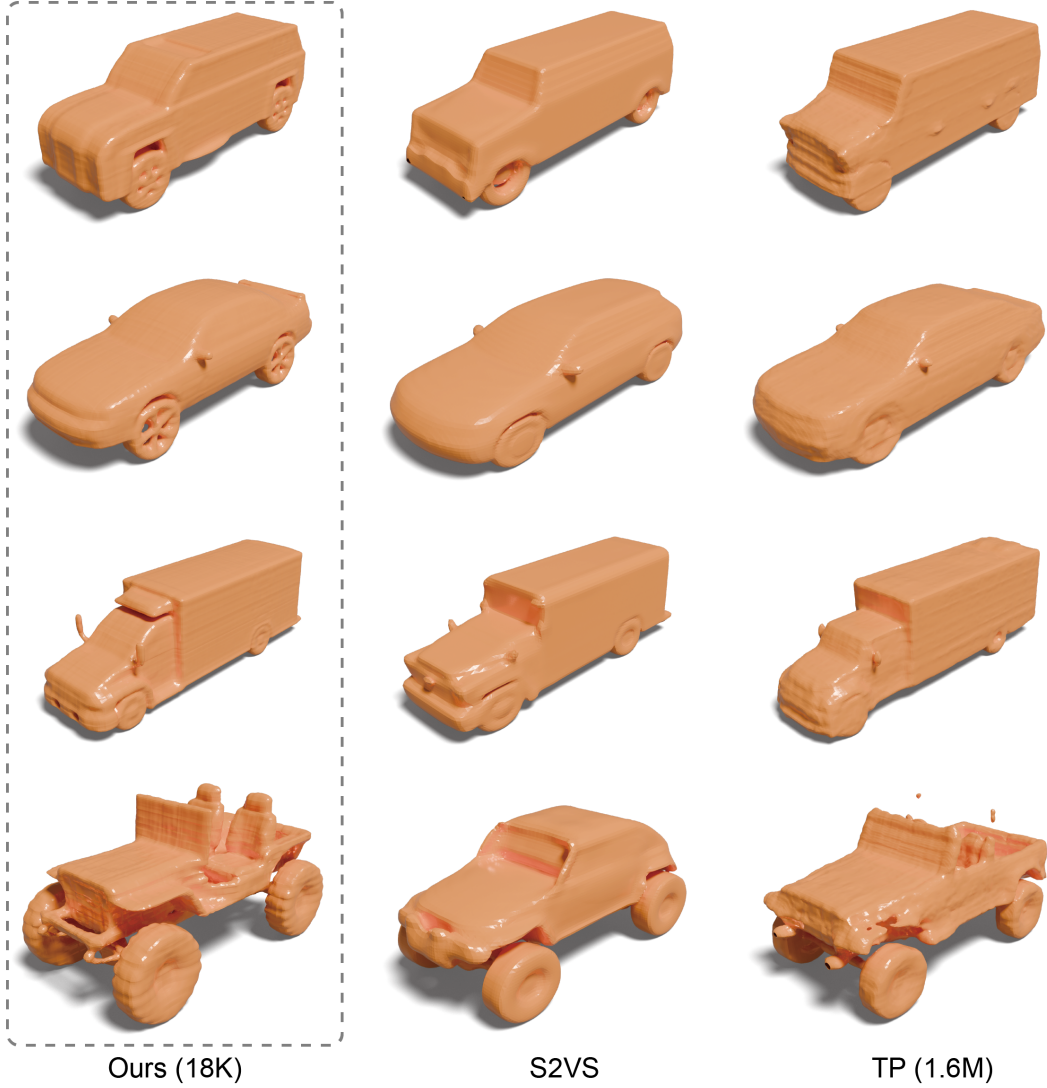


Figure 15: Shapes generated by our method and shapes retrieved (CD query) from S2VS Zhang et al. (2023) and TP Shue et al. (2023) on *car* category. The parameter count used for representing each shape is indicated while S2VS is not specified due to its use of a network-based compression mechanism.

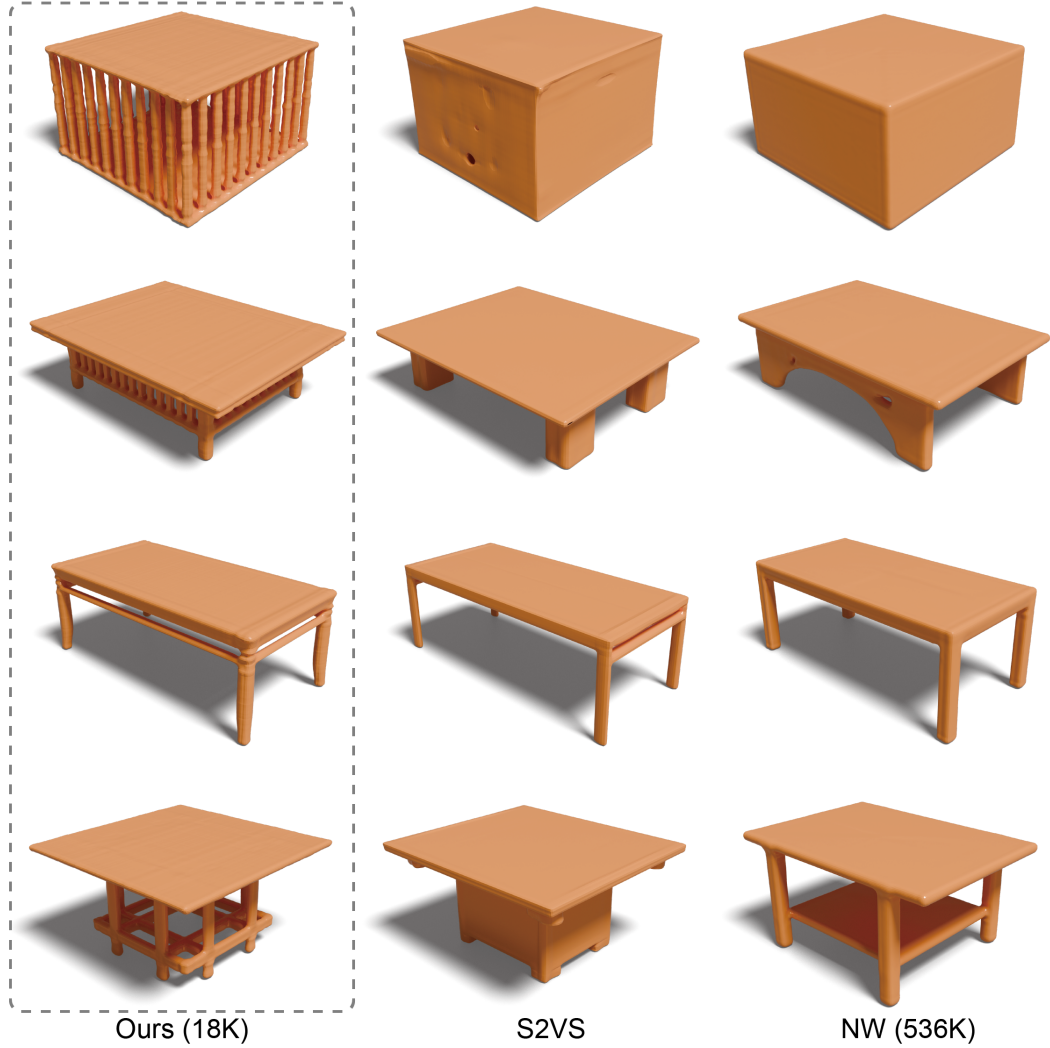


Figure 16: Shapes generated by our method and shapes retrieved (CD query) from S2VS Zhang et al. (2023) and NW Hui et al. (2022) on *table* category. The parameter count used for representing each shape is indicated while S2VS is not specified due to its use of a network-based compression mechanism.

972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

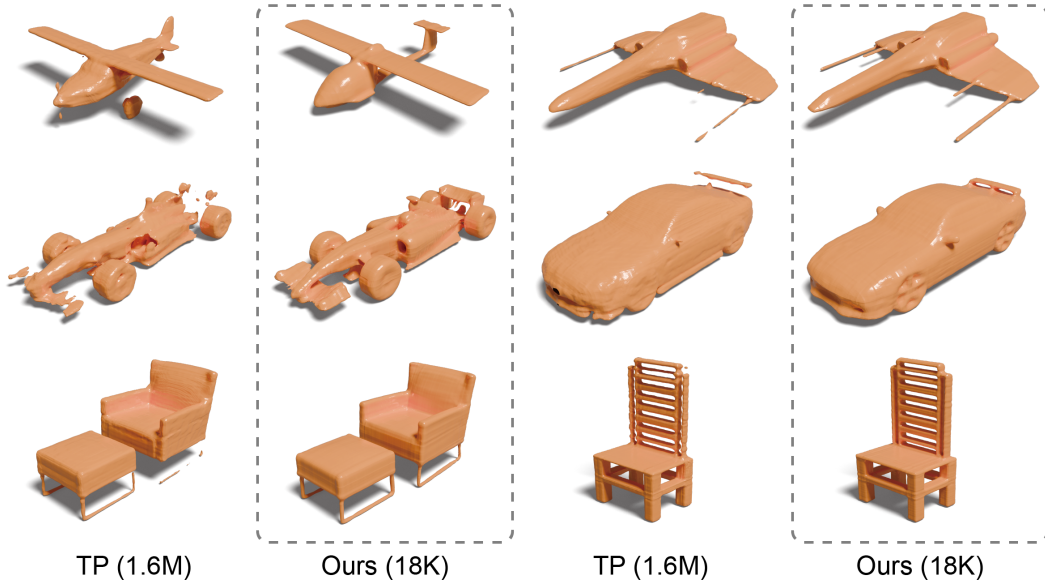


Figure 17: Shapes generated by TP Shue et al. (2023) and shapes retrieved (CD query) from our generated samples on *airplane*, *car*, and *chair* categories. The parameter count used for representing each shape is indicated.

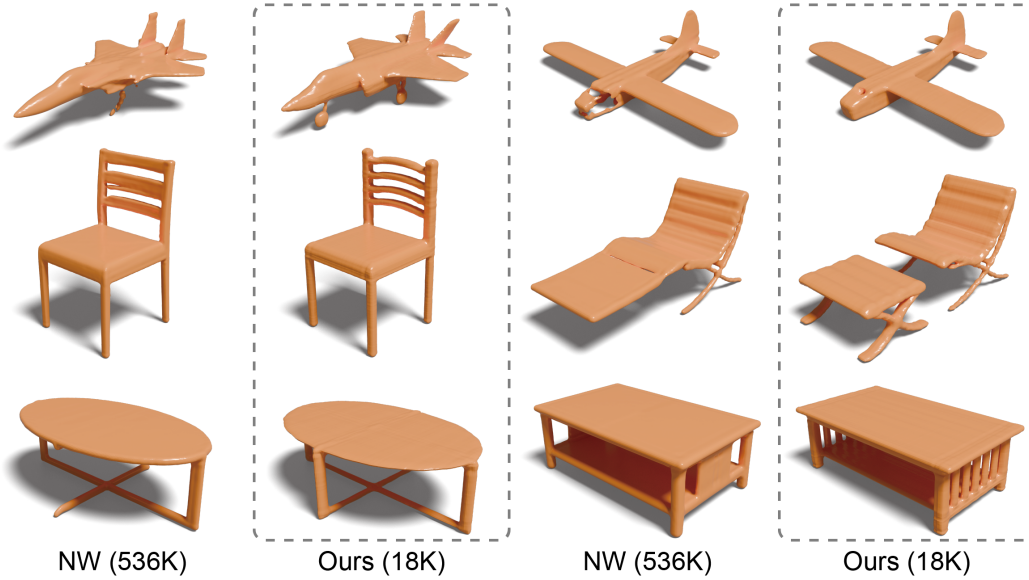


Figure 18: Shapes generated by NW Hui et al. (2022) and shapes retrieved (CD query) from our generated samples on *airplane*, *chair*, and *chair* categories. The parameter count used for representing each shape is indicated.

1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079

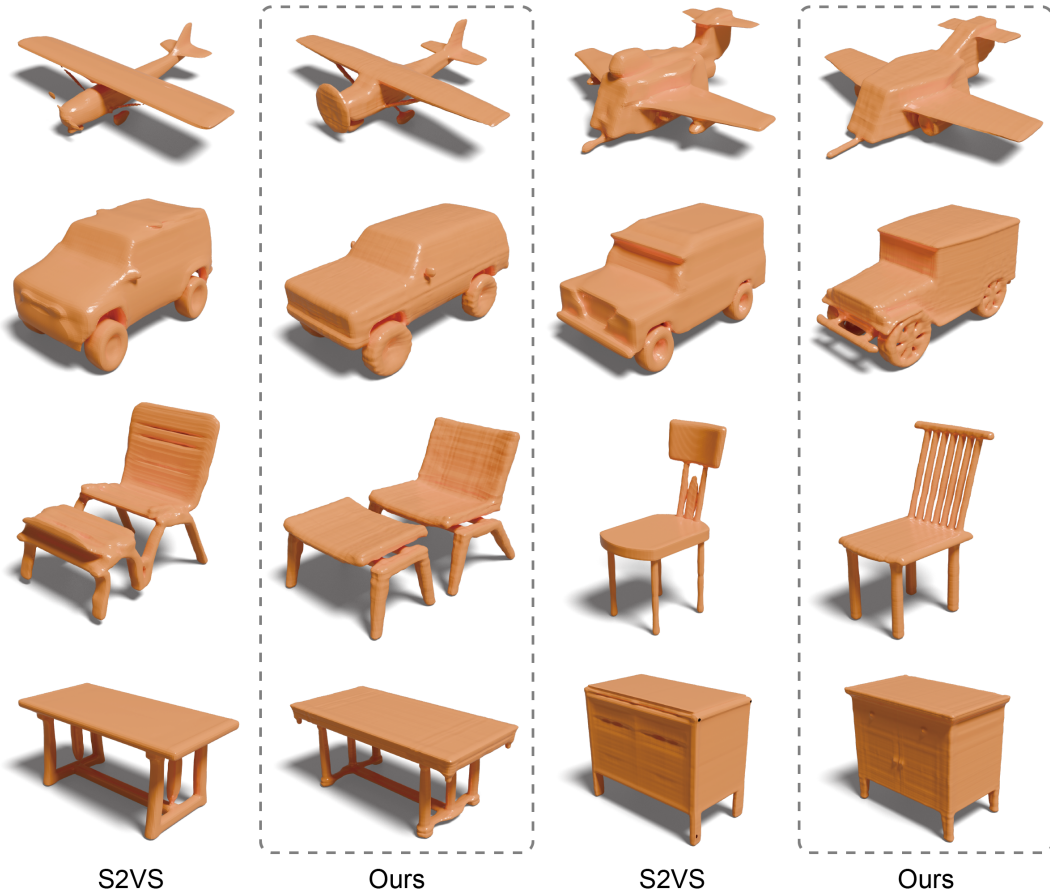
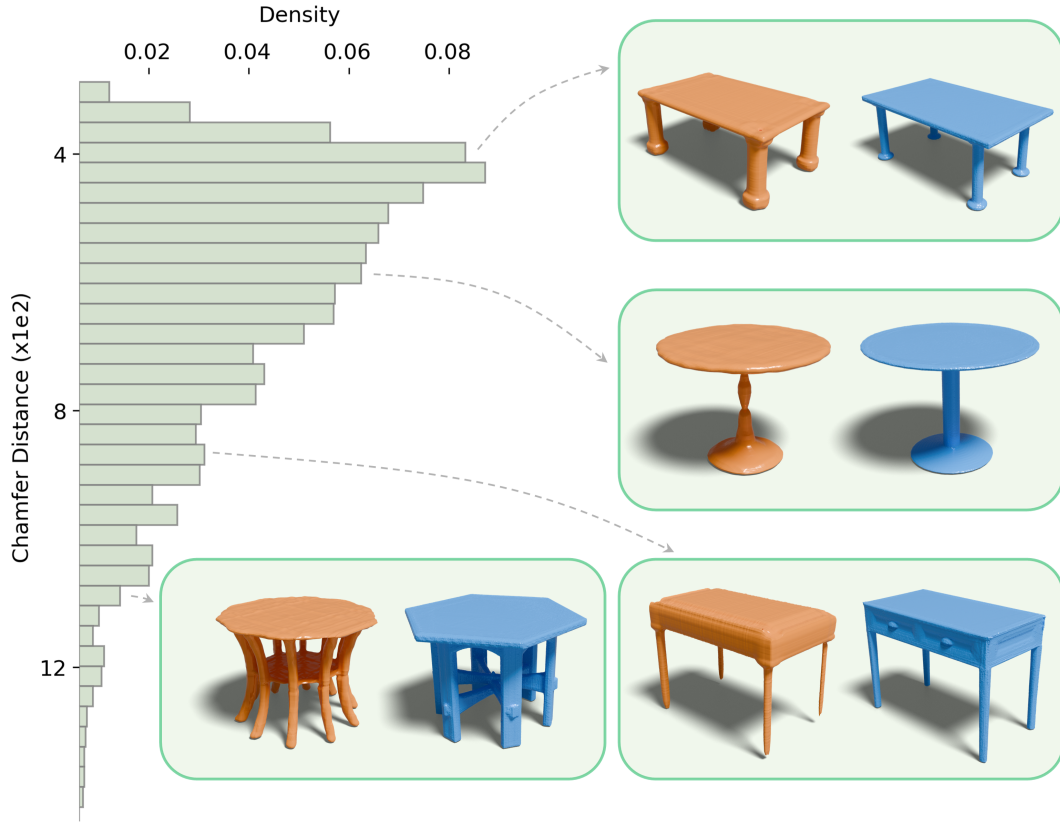


Figure 19: Shapes generated by S2VS Zhang et al. (2023) and shapes retrieved (CD query) form our generated samples on *airplane*, *car*, *chair*, and *table* categories.



The Chamfer Distance (CD) Distribution of Our Generated Shapes

Figure 20: We generate 500 tables using our method; for each table, we retrieve the most similar shape in the training set by Chamfer Distance (CD); then, we plot the distribution of CDs for all retrievals. *Orange* means the shape generated by our method. *Blue* means the most similar shape retrieved from training set.

1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187



Figure 21: Reconstruction of textured shapes based on our Tri-Vectors, where each shape is represented with only 92K parameters.

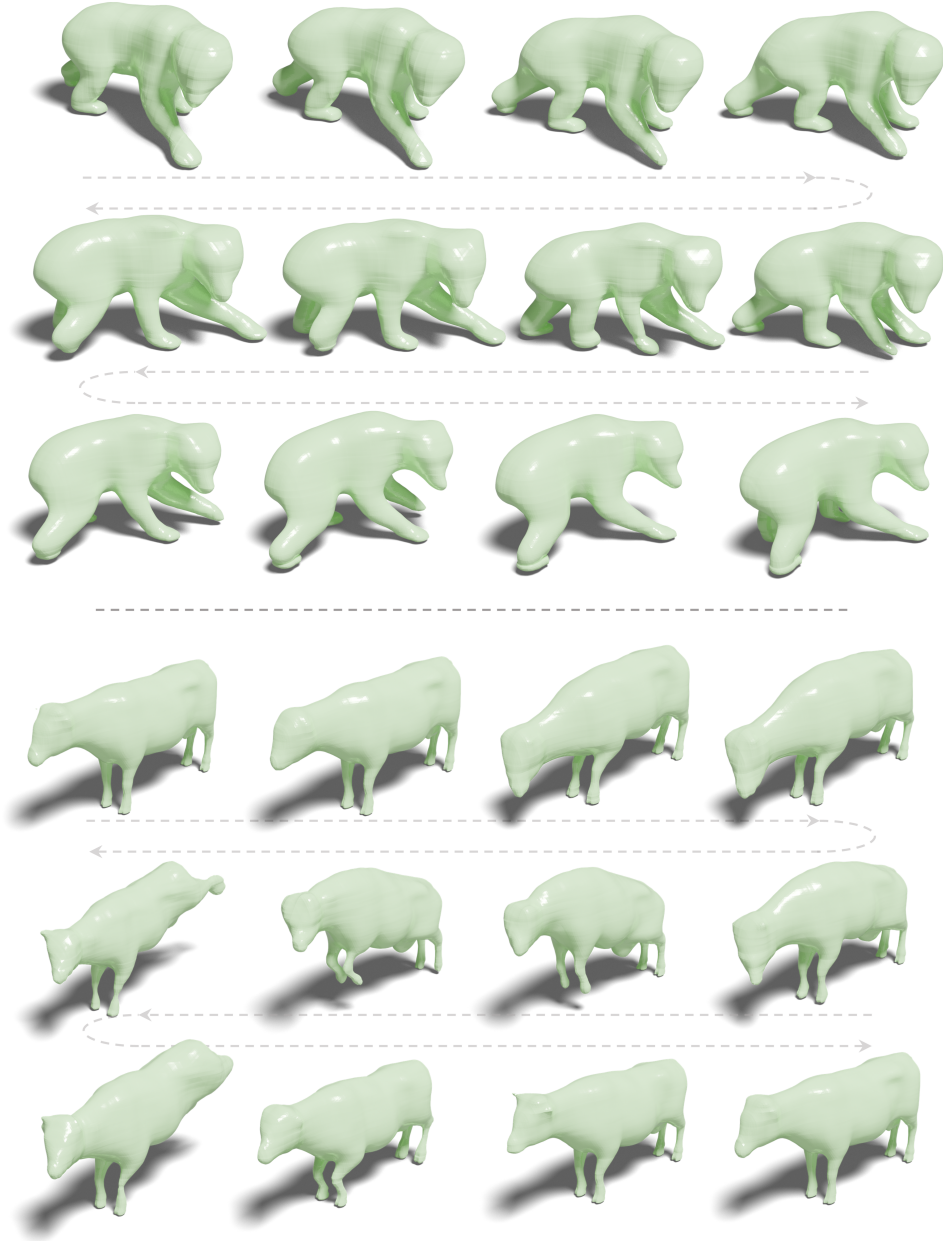


Figure 23: Reconstruction of deformable shapes based on our Tri-Vectors, where each shape is represented with only 98K parameters.

REFERENCES

- Anpei Chen, Zexiang Xu, Xinyue Wei, Siyu Tang, Hao Su, and Andreas Geiger. Dictionary Fields: Learning a neural basis decomposition. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 42(4): 156:1–156:12, 2023.
- Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, pp. 5939–5948, 2019.
- Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F. Yago Vicente, Thomas Dideriksen, Himanshu Arora, Matthieu Guillaumin, and Jitendra Malik. ABO: Dataset and benchmarks for real-world 3D object understanding. In *CVPR*, pp. 21094–21104, 2022.
- Steffen Dereich, Robin Graeber, and Arnulf Jentzen. Non-convergence of Adam and other adaptive stochastic gradient descent optimization methods for non-vanishing learning rates. *CoRR*, abs/2407.08100, 2024.
- Quankai Gao, Qiangeng Xu, Hao Su, Ulrich Neumann, and Zexiang Xu. Strivec: Sparse tri-vector radiance fields. In *ICCV*, pp. 17569–17579, 2023.
- Jingyu Hu, Ka-Hei Hui, Zhengzhe Liu, Ruihui Li, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3D shape generation, inversion, and manipulation. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 43(2):16:1–16:18, 2024.
- Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3D shape generation. In *Proc. SIGGRAPH ASIA*, pp. 24:1–24:9, 2022.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.
- Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. SiT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *ECCV*, volume 15135, pp. 23–40, 2024.
- Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 41(4): 102:1–102:15, 2022.
- Pratheba Selvaraju, Mohamed Nabail, Marios Loizou, Maria Maslioukova, Melinos Averkiou, Andreas Andreou, Siddhartha Chaudhuri, and Evangelos Kalogerakis. BuildingNet: Learning to label 3D buildings. In *ICCV*, pp. 10377–10387, 2021.
- J. Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3D neural field generation using triplane diffusion. In *CVPR*, pp. 20875–20886, 2023.
- Lior Yariv, Omri Puny, Oran Gafni, and Yaron Lipman. Mosaic-SDF for 3D generative models. In *CVPR*, pp. 4630–4639, 2024.
- Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3DShape2VecSet: A 3D shape representation for neural fields and generative diffusion models. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 42(4):92:1–92:16, 2023.