

Appendix - Sparc3D: Sparse Representation and Construction for High-Resolution 3D Shapes Modeling

Zhihao Li^{1,2,*}, Yufei Wang¹, Heliang Zheng^{2,‡}, Yihao Luo^{2,3,†}, Bihan Wen^{1,†}

¹Department of EEE, Nanyang Technological University, Singapore

²Math Magic ³Imperial-X, Imperial College London, UK

zhihao005@e.ntu.edu.sg, yufei001@e.ntu.edu.sg, zhenghllj@gmail.com

y.luo23@imperial.ac.uk, bihan.wen@ntu.edu.sg

<https://lizhihao6.github.io/Sparc3D>

A Sparconv-VAE Implementation Details

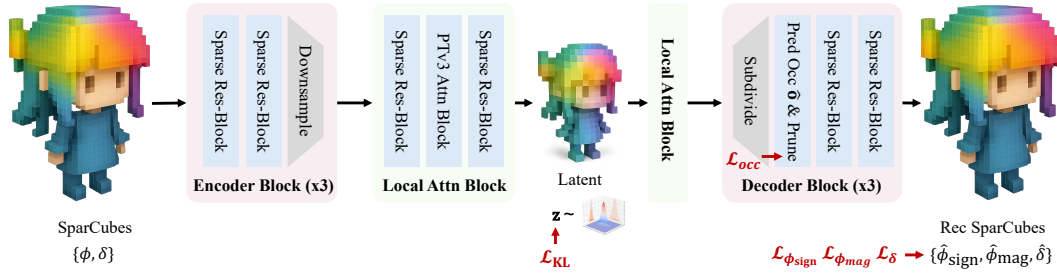


Figure A1: **SparConv-VAE architecture.** Our SparConv-VAE comprises a sequence of sparse convolutional blocks, integrated with a lightweight local attention module adapted from PTv3 [6].

Similar to XCube [5] and Flux [4], we employ a variational autoencoder to learn a compact latent representation for our SparCubes as shown in Fig. A1. The encoder consists of three downsampling blocks, each comprising two sparse residual convolutional blocks and a downsampling layer. Following Flux [4], we apply group normalization [7] with 32 groups and the Swish activation function [2]. In each downsampling layer, we first perform voxel shuffling to reduce spatial resolution, then use a linear layer to decrease channel dimensionality. After the final downsampling block, we insert a local attention module—adapted from PTv3 [6]—to further fuse neighborhood features.

The decoder mirrors this design with three upsampling blocks. Each block begins with a linear layer to expand channel width, followed by shuffle-based upsampling to increase spatial resolution, and two sparse residual blocks. We then apply a convolutional layer to predict an occupancy mask, which is pruned before feature propagation. At the final stage, a tanh activation predicts the SparCubes parameters. Channel dimensions grow progressively from 128 to 256 to 512 across the scales. Empirically, we set the loss weights to $\mathcal{L}_{occ} = 0.02$, $\mathcal{L}_{sign} = 0.05$, $\mathcal{L}_{mag} = 0.1$, $\mathcal{L}_{\delta} = 0.1$, and $\mathcal{L}_{KL} = 1 \times 10^{-5}$.

B Comparison with SparseFlex

Concurrently, SparseFlex [3] adapts Trellis’s VAE [8] for high-resolution 3D reconstruction by replacing the original DINOv2 features with surface-sampled points (with normals) and employing

*This work was conducted during Zhihao Li’s research internship at Math Magic.

†Corresponding authors; ‡project lead.

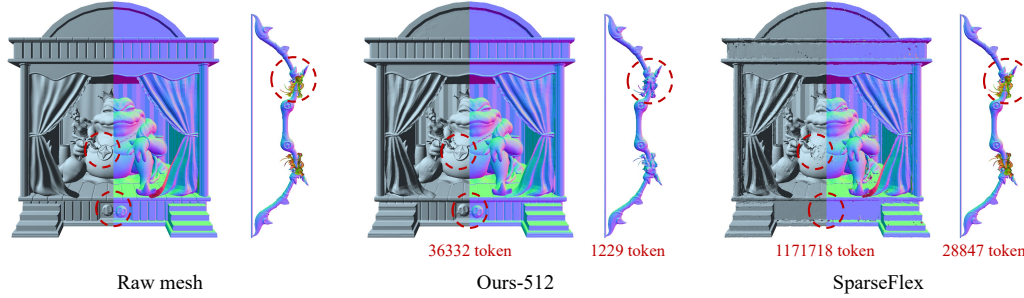


Figure A2: **Qualitative comparison of VAE reconstructions from our method and SparseFlex [3].** SparseFlex exhibits token-count explosion (1,171,718 tokens for complex objects), loss of fine geometric details (*e.g.*, the house’s prismatic pillar), incorrect face normals (*e.g.*, the dragon head on the bow), and numerous holes. *Best viewed with zoom-in.*

a 256^3 latent voxel grid. Nonetheless, in addition to the issues inherited from Trellis [8] (modality conversion and non-watertight outputs), SparseFlex exhibits four further critical limitations compared to our method, as illustrated in Fig. A2:

- **Token-count explosion.** Using a 256^3 grid with redundant sampling inflates their token count to 1,171,718 for a complex object—roughly the length of *Les Misérables* and $32\times$ larger than ours—making high-resolution generative training impractical.
- **Loss of fine details** Uniformly sampling points within each voxel cannot capture sharp edges or intricate geometry, leading to blurred or missing features. By contrast, our progressively downsampled, 3D-supervised pipeline faithfully recovers all geometric nuances.
- **Dependence on raw mesh normals.** Their pipeline relies on the input mesh’s face normals and must automatically recompute them when they’re incorrect—an operation that can fail and introduce erroneous normals into the result. In contrast, our method has no such dependency, ensuring consistently accurate surface orientations.
- **Numerous holes.** The output mesh may contain holes, and because it has many open surfaces with half-edge boundaries, these holes are difficult to detect and repair.

C Comparison with Trellis

To disentangle representation quality from training scale, we align the training setup at a comparable resolution and report efficiency under identical hardware. Both models encode a latent tensor of spatial size 64^3 (with the same sparsity ratio). The key differences are: (i) TRELLIS employs sparse transformer blocks (Swin-style attention), while our VAE is built entirely on sparse convolutions with explicit sparse down/up-sampling; (ii) our decoder targets higher output resolution and channel capacity. Concretely, our VAE uses 16 latent channels and decodes to 512^3 , whereas TRELLIS uses 8 channels and decodes to 256^3 . Under the conventional compression definition (total voxels per latent scalar), this yields a $4\times$ *higher* compression ratio for ours. For an apples-to-apples efficiency comparison at 256^3 , we adopt a two-stage pipeline ($256^3 \rightarrow 64^3 \rightarrow 256^3$), while TRELLIS keeps its original single-stage ($64^3 \rightarrow 256^3$).

D Remeshing Cost Comparison

We compare the remeshing latency of our method against Dora [1] across different resolutions. At low resolutions (below 512^3), our remeshing pipeline delivers performance comparable to Dora’s. However, at high resolutions (above 512^3), our approach is significantly faster, thanks to its efficient custom CUDA kernel design. Moreover, Dora [1] requires additional surface sampling and near-surface point refinement to train its VAE, which incurs extra computational overhead—an expense our method entirely avoids.

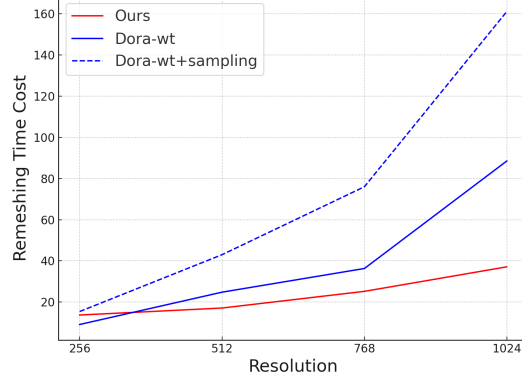


Figure A3: Comparison of remeshing time costs between our method and Dora [1].

E Additional Generation Results

In Fig. A4, we present additional comparisons of 3D assets generated by our method, as detailed in the Experiments section of the main paper.

References

- [1] R. Chen, J. Zhang, Y. Liang, G. Luo, W. Li, J. Liu, X. Li, X. Long, J. Feng, and P. Tan. Dora: Sampling and benchmarking for 3d shape variational auto-encoders. *arXiv preprint arXiv:2412.17808*, 2024.
- [2] A. Fatima and A. Pethe. Nvm device-based deep inference architecture using self-gated activation functions (swish). In *Machine Vision and Augmented Intelligence—Theory and Applications: Select Proceedings of MAI 2021*, pages 33–44. Springer, 2021.
- [3] X. He, Z.-X. Zou, C.-H. Chen, Y.-C. Guo, D. Liang, C. Yuan, W. Ouyang, Y.-P. Cao, and Y. Li. Sparseflex: High-resolution and arbitrary-topology 3d shape modeling. *arXiv preprint arXiv:2503.21732*, 2025.
- [4] B. F. Labs. Flux. <https://github.com/black-forest-labs/flux>, 2024.
- [5] X. Ren, J. Huang, X. Zeng, K. Museth, S. Fidler, and F. Williams. Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4209–4219, 2024.
- [6] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4840–4851, 2024.
- [7] Y. Wu and K. He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.
- [8] J. Xiang, Z. Lv, S. Xu, Y. Deng, R. Wang, B. Zhang, D. Chen, X. Tong, and J. Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024.



Figure A4: More generation results. *Best viewed with zoom-in.*