LoG3D: Ultra-High-Resolution 3D Shape Modeling via Local-to-Global Partitioning

Supplementary Material

A. More Implementation Details

A.1. Architecture and Pre-processing

Our LoG-VAE model employs a symmetric encoder-decoder architecture. The pre-processing pipeline begins with the ground-truth meshes, which are canonically normalized into a unit cube of $[-0.5, 0.5]^3$. From these, we sample and compute Unsigned Distance Function (UDF) fields at resolutions of 1024^3 and 2048^3. To focus the model on the region near the surface, we truncate the UDF values: for the 1024^3 and 2048^3 fields, we clamp all values at a maximum threshold of $4/1024$ and $4/2048$, respectively. Any UDF value exceeding this threshold is set to the maximum value. Finally, the entire clamped UDF volume is min-max normalized to the range $[0, 1]$.

A.2. UBlock Setting and Sparsification

To enable a seamless fine-tuning process from the 1024^3 model to the 2048^3 model, we fix the initial spatial resolution of all UBlocks to $D = 8^3$. Consequently, the partition factor s varies with the input resolution: for a 1024^3 field, $s = 1024/8 = 128$, and for a 2048^3 field, $s = 2048/8 = 256$. After partitioning the entire UDF field, we perform a sparsification step. We retain only the "active" UBlocks—defined as those containing at least one inner voxel UDF value below the maximum threshold. This dramatically reduces the number of blocks from a potential s^3 to a much smaller set of L blocks. Finally, we add a padding of $\alpha = 2$ voxels around each of these L active UBlocks. The final resolution of a UBlock fed into the network is thus $D = (8 + 2 * 2)^3 = 12^3$.

A.3. Encoder

Due to GPU memory constraints, our network operates with a batch size of one. The input to the encoder is a tensor of shape $[1, L, 12^3]$, representing the set of L sparse UBlocks. For cases with a large L, we process the UBlocks in chunks to further mitigate memory consumption.

Local 3D Convolution. The encoder first processes each UBlock with a local 3D convolutional module. To achieve this efficiently, we treat the L dimension as a batch dimension, reshaping the input tensor to $[L, 1, 12, 12, 12]$. An initial MLP layer then projects these single-channel features into a higher-dimensional space, expanding the channel dimension to 32 and resulting in a tensor of shape $[L, 32, 12, 12, 12]$. Our local 3D convolution module consists of four layers, where each layer sequentially performs the following operations: a GroupNorm, an MLP for feature mapping, a 3x3x3 convolution with a ReLU activation, a second GroupNorm, another 3x3x3 convolution with a ReLU activation, and finally, a MaxPooling3D operation for downsampling. With each layer, the spatial resolution of the UBlock is halved ($12 \to 6 \to 3 \to 1$), while the number of feature channels doubles ($32 \to 64 \to 128 \to 256$). After passing through the four-layer local 3D convolution module, each of the L UBlocks is encoded into a feature vector, yielding a tensor of shape $[L, 256, 1, 1, 1]$.

Global Sparse Transformers. To facilitate information exchange between these UBlocks, we employ a global sparse transformer module. We treat each of the L feature vectors as a distinct token, reshaping the tensor to $[1, L, 256]$ to form a sequence. This sequence is then processed by a series of 8 sparse transformer blocks. Each block is based on the shift-window-attention mechanism with the following configuration: the input feature dimension is 256, the attention mechanism uses 8 heads, and the window size for the shifted-window attention is set to 8. Absolute positional encodings are added to the input sequence to retain spatial information. Furthermore, to enhance contextual interaction, an additional sparse voxel convolution layer that preserves the feature dimension is appended to the end of each transformer block. After passing through all 8 blocks, the output tensor retains its shape of $[1, L, 256]$. Finally, to produce the sparse latents required by the VAE, this global feature representation is projected down to a lower-dimensional space using a final MLP. This yields a final latent tensor of shape $[L, 16]$.

A.4. Decoder

The decoder mirrors the encoder's architecture, first applying a global sparse transformer followed by a local 3D deconvolution module. The process begins with the sparse latent vectors of shape $[L, 16]$. These are first projected back to a 256-dimensional space via an MLP and then reshaped into a sequence of shape $[1, L, 256]$. This sequence is processed by a series of 8 sparse transformer blocks, which share the same configuration as their counterparts in the encoder. The output is then reshaped back to $[L, 256, 1, 1, 1]$ to be fed into the local decoding module. The local module consists of five 3D convolutional blocks designed to upsample the feature maps. Each block sequentially performs: a 3D nearest-neighbor interpolation for upsampling, a GroupNorm, a 3x3x3 convolution with ReLU activation, a second GroupNorm, and another 3x3x3 convolution with ReLU ac-

tivation. Through these five blocks, the spatial resolution of each UBlock is progressively doubled ($1 \to 2 \to 4 \to 8 \to 16$), while the number of feature channels is halved ($256 \to 128 \to 64 \to 32 \to 16$). After the fifth block, each UBlock is represented by a tensor of shape $[L, 16, 16, 16, 16]$.

To produce the final UDF values, we first use an MLP to project the 16-channel feature map back to a single channel representing the UDF, resulting in a tensor of shape $[L, 1, 16, 16, 16]$. From this upsampled 16^3 volume, we crop out the central 12^3 region, which corresponds to the original padded UBlock size. This yields the decoder's final output, a tensor of shape $[L, 1, 12, 12, 12]$.

A.5. Mesh Reconstruction

In the final stage, we reconstruct the high-resolution mesh from the decoded UBlocks. First, we assemble the L decoded UBlocks, each of shape $[12, 12, 12]$, back into a full UDF field. This is done by placing each block at its original spatial location as determined during the initial partitioning stage. The UDF values are then de-normalized to their original value range. Due to the padding added before the encoder, adjacent UBlocks have overlapping voxel regions. To ensure a seamless and continuous surface, we apply our Pad-Average strategy: for any voxel located in an overlapping region, its final UDF value is computed as the average of the values from all contributing UBlocks.

Finally, we extract the mesh from this fully reconstructed UDF field using the Marching Cubes [4] algorithm. Since a UDF field represents unsigned distances, we cannot extract a zero-level set directly. Instead, we extract an isosurface at a small threshold ϵ. For our 1024^3 and 2048^3 resolution fields, we set ϵ to $1/1024$ and $1/2048$, respectively. Given the high resolution of the UDF field, this slight "inflation" of the surface has a negligible impact on the final mesh quality.

A.6. Training Details

Our framework is implemented in PyTorch [6]. For optimization, we use the AdamW [5] optimizer with a learning rate of $5e-5$ and a weight decay of 0.01. The models are trained using the Huber loss, with the delta threshold set to 0.2. The weight λ assigned to the KL-divergence loss is set to $1e-6$. To accelerate training, we leverage the Accelerate library for multi-GPU distributed training with fp16 mixed-precision. The batch size is set to 1 throughout all experiments due to GPU memory constraints.

Our 1024^3 LoG-VAE model was trained from scratch for approximately 3 days on 8 NVIDIA H20 GPUs. The 2048^3 resolution model was then fine-tuned from the pre-trained 1024^3 checkpoint for an additional 2 days on the same hardware setup. We will make our training and inference code publicly available to facilitate future research in the community.

B. Metrics

The Normal Mean Squared Error (NMSE) and Sharp Normal Error (SNE) metrics are adopted from Dora [1].

To compute the reconstruction metrics, we densely sample points from the outer surfaces of both the ground-truth meshes (denoted as $\mathbf{X}_{gt}$) and the predicted meshes (denoted as $\mathbf{X}_{pd}$), following the evaluation protocol of TRELLIS [9].

Chamfer Distance (CD). The Chamfer distance is computed using:

$$
\begin{aligned}
CD &= \frac{1}{2}(Comp. + Acc.), \\
Comp. &= \frac{1}{|\mathbf{X}_{gt}|} \sum_{x_{gt} \in \mathbf{X}_{gt}} \min_{x_{pd} \in \mathbf{X}_{pd}} ||x_{gt} - x_{pd}||, \\
Acc. &= \frac{1}{|\mathbf{X}_{pd}|} \sum_{x_{pd} \in \mathbf{X}_{pd}} \min_{x_{gt} \in \mathbf{X}_{gt}} ||x_{pd} - x_{gt}||
\end{aligned}
$$

F-Score (F1). The F-Score is defined as follows:

$$
FS = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall},
$$

where

$$
\begin{aligned}
Precision &= \frac{|\{x_{pd} \in \mathbf{X}_{pd} | \min_{x_{gt} \in \mathbf{X}_{gt}} ||x_{gt} - x_{pd}|| < \xi\}|}{|\mathbf{X}_{pd}|}, \\
Recall &= \frac{|\{x_{gt} \in \mathbf{X}_{gt} | \min_{x_{pd} \in \mathbf{X}_{pd}} ||x_{pd} - x_{gt}|| < \xi\}|}{|\mathbf{X}_{gt}|}
\end{aligned}
$$

We use $\xi = 0.01$ and $\xi = 0.001$ for the experiments.

C. Efficiency and Performance Comparison

Tab. 1 presents a quantitative comparison of our LoG-VAE against previous state-of-the-art methods, Direct3D-S2 [8] and TripoSF [2], in terms of inference time and the number of trainable parameters. All models were benchmarked on the 1024^3 resolution mesh reconstruction task. The results demonstrate our model's superior efficiency, as it requires both less inference time and fewer parameters than competing methods. Crucially, this efficiency is achieved without compromising reconstruction quality; in fact, our model delivers superior results (as shown in the main paper), highlighting the advantages of our architecture.

D. Additional Qualitative Comparisons

We provide additional qualitative comparisons against Hunyuan3D-2.1 [3] and TRELLIS [9] to further validate the performance of our model. Fig. 1 visually compares the reconstruction results from our models (at 1024^3 and 2048^3 resolutions) and these two methods. The results demonstrate that our approach consistently produces superior reconstructions across a diverse range of shapes. This holds

Table 1. **Comparison of Inference Time and Model Size.** All models were benchmarked on the 1024^3 resolution mesh reconstruction task using a single NVIDIA H20 GPU. Our model is both faster and more compact.

Method	Direct3D-S2 [8]	TripoSF [2]	Ours
Inference time per sample	18s	60s	15s
Parameters	84M	586M	77M

true for both common object categories, such as furniture, and for highly complex geometries. In all cases, our method yields results that are visually more faithful to the ground truth.

References

[1] Rui Chen, Jianfeng Zhang, Yixun Liang, Guan Luo, Weiyu Li, Jiarui Liu, Xiu Li, Xiaoxiao Long, Jiashi Feng, and Ping Tan. Dora: Sampling and benchmarking for 3d shape variational auto-encoders. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16251–16261, 2025. 2

[2] Xianglong He, Zi-Xin Zou, Chia-Hao Chen, Yuan-Chen Guo, Ding Liang, Chun Yuan, Wanli Ouyang, Yan-Pei Cao, and Yangguang Li. Sparseflex: High-resolution and arbitrary-topology 3d shape modeling. *arXiv preprint arXiv:2503.21732*, 2025. 2, 3

[3] Team Hunyuan3D, Shuhui Yang, Mingxin Yang, Yifei Feng, Xin Huang, Sheng Zhang, Zebin He, Di Luo, Haolin Liu, Yunfei Zhao, et al. Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material. *arXiv preprint arXiv:2506.15442*, 2025. 2, 4

[4] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998. 2

[5] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 2

[7] Stefan Stojanov, Anh Thai, and James M Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1798–1808, 2021. 4

[8] Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Yikang Yang, Yajie Bao, Jiachen Qian, Siyu Zhu, Xun Cao, Philip Torr, et al. Direct3d-s2: Gigascale 3d generation made easy with spatial sparse attention. *arXiv preprint arXiv:2505.17412*, 2025. 2, 3

[9] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21469–21480, 2025. 2, 4

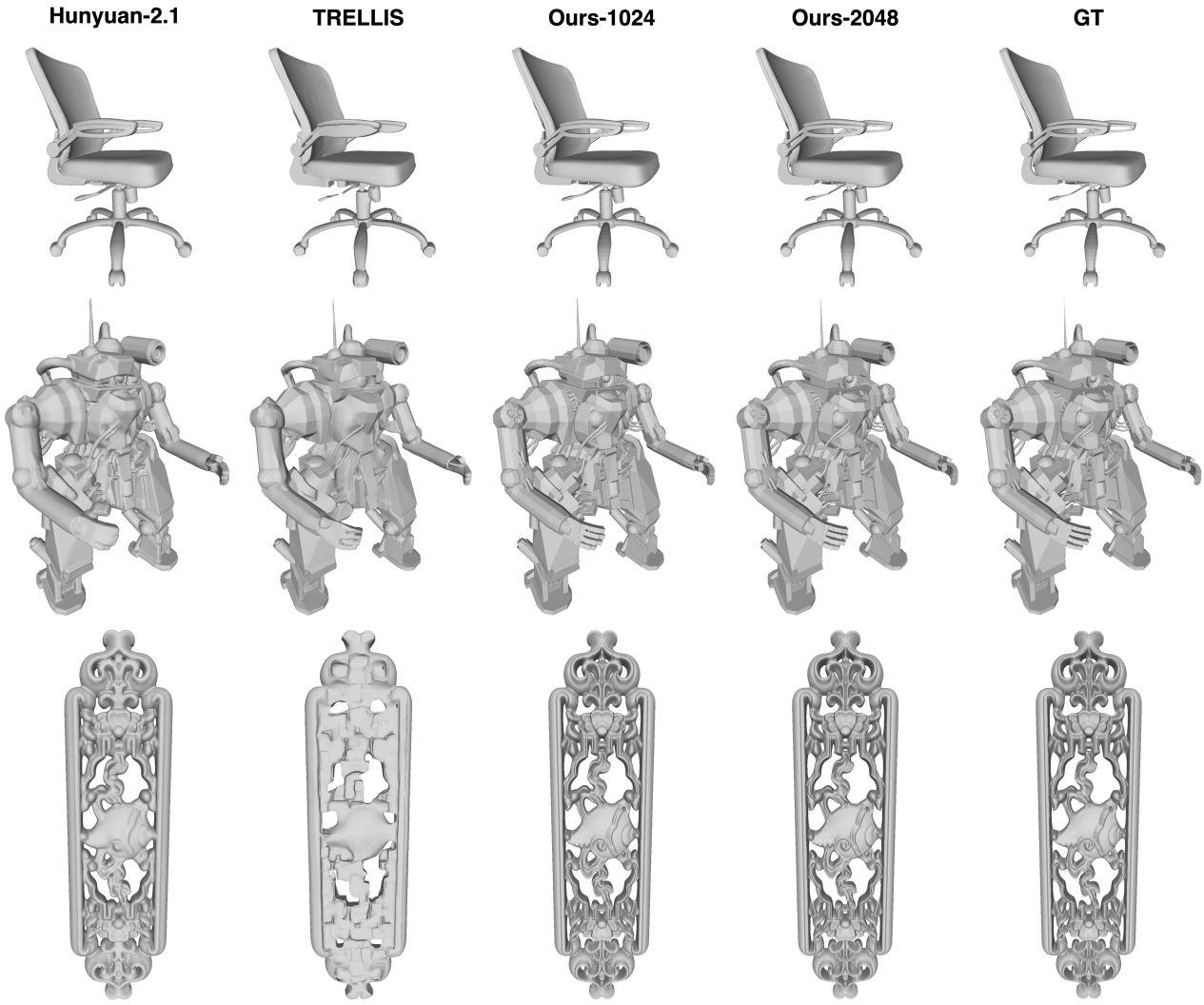

Figure 1. **Qualitative Comparison of VAE reconstruction with Hunyuan3D-2.1 [3] and TRELLIS [9].** The 3D models depicted in the figure are sourced from the Toys4k [7] and iHome dataset. Our method yields high-fidelity reconstructions for a diverse range of geometries, demonstrating its superior ability to simultaneously preserve surface smoothness and capture intricate topological details.