Unleashing Vecset Diffusion Model for Fast Shape Generation Supplementary Material

Zeqiang Lai[1,2⋆] , Yunfei Zhao[2,3⋆] , Zibo Zhao[2,4] , Haolin Liu[2] , Fuyun Wang[1]
Huiwen Shi[2] , Xianghui Yang[2] , Qingxiang Lin[2] , Jingwei Huang[2]
Yuhong Liu[2] , Jie Jiang[2] , Chunchao Guo[2†] , Xiangyu Yue[1†]
[1]MMLab, CUHK [2]Tencent Hunyuan [3]VISG, NJU [4]ShanghaiTech
https://github.com/Tencent-Hunyuan/FlashVDM

A. Implementation Details

Decoder Finetuning. The efficient vecset decoder illustrated in Section 3.2 is fine-tuned by freezing the vecset encoder. In [6], the decoder is made up of 8 self-attention layers and 1 cross-attention layer. Since our design only alters the cross-attention layer, we initialize self-attention layers as before. Both self- and cross-attention layers are trained during the finetuning. We use a constant learning rate of 1×10^{-4} and a batch size of 256. The decoder could quickly converge to a pretty good one with 300k steps, but we find longer training to 800k steps converges better, leading to nearly identical performance to the original one.

Diffusion Distillation. The batch size is always 256 for different stages in progressive flow distillation. Following [2, 3], the guidance distilled model is conditioned on the guidance strength w, which is injected into the diffusion backbone with a similar approach as timestep. During training, w is randomly select from $w \sim U[2, 8]$. The learning rate is set to 1×10^{-6}. The model is trained with 20k steps. For step distillation, we set λ of huber loss to 1×10^{-3}, and the guidance strength is set to a constant of 5.0. Following [2], we also use the skipping-step technique with $k = 10$. We utilize multiphase [4] techniques to train the model for 20k steps with 5 phases and a learning rate of 1×10^{-6} and then finetune the model for 8k steps with a learning rate of 1×10^{-7}. The EMA decay rate is set to 0.999. For adversarial fine-tuning, we keep the distillation loss of the previous stage and set the adversarial loss weight to 0.1. The learning rate is set to 1×10^{-7} for generator and 1×10^{-6} for discriminator. We train 5k steps for this stage.

B. Generalization Capability

Generalization to other architectures. The proposed acceleration techniques are extendable. The hierarchical volume decoding aims to speed up the implicit function queries, which is general to most methods that apply implicit representations (including SDF, NeRF, and UDF). The adaptive KV selection aims to accelerate the querying operation in the perceiver-style architecture. Therefore, this method could speed up 2D, 3D, and video VAE built with a cross-attention decoder. The distillation approach accelerates the diffusion (flow) model that learns with vector set representation, potentially extending to other token set-based generative models.

Figure 1. Illustration of FlashVDM on different VDMs.

Generalization to other VDM. Thank you for the suggestion. At the time of FlashVDM's submission, Hunyuan3D-2 was the best-performing open-source VDM, significantly outperforming other available models. Therefore, we chose it to validate our algorithms. As 3D generation has rapidly progressed, more open-source VDMs such as TripoSG, Step1X-3D, HoloPart, and DetailGen3D have become available, and the VAE component of FlashVDM has also been integrated into some of these models. Here, we include distillation results using the architectures of TripoSG and Michelangelo—the former representing a different DiT structure, and the latter a different VAE structure, as shown in Figure 1.

C. Details of Hierarchical Volume Decoding

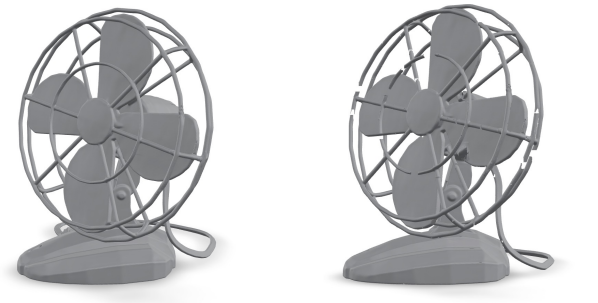
Effect of Dilate and tSDF. Fig. 2 compares the reconstruction with and without dilate and tSDF strategy. It can be

⋆ Equal contribution. † Corresponding authors.

Algorithm 1 Hierarchical Volume Decoding.

Input: An implicit function $f(\mathbf{p})$ that evaluates the SDF at position $\mathbf{p}$. Target resolution $\mathbf{r}$. Shape latents Z, tSDF threshold η, isosurface threshold γ.
Output: A signed distance field (SDF) $S \in \mathcal{R}^{r \times r \times r}$.

$R = \text{GetResolutions}(r)$ ▷ List of cascade resolution.
$P_{R[0]} = \text{GenGridPoints}(R[0])$
$S_{R[0]} = \text{QueryField}(P_{R[0]}, Z)$
for $i = 1$ **to** $\text{len}(R)$ **do**
  $\hat{P}_{R[i]} = \text{FindIntersect}(S_{R[i-1]}, \gamma)$
  $\hat{P}_{R[i]}$ += $\text{FindNear}(S_{R[i-1]}, \eta)$
  $\hat{P}_{R[i]} = \text{Dilate}(\hat{P}_{R[i]})$
  $P_{R[i]} = \text{Expand}(\hat{P}_{R[i]})$
  $S_{R[i]} = \text{QueryField}(P_{R[i]}, Z)$
end for
return $S_{R[-1]}$



(a) Octree Volume Decoding (b) Octree Volume Decoding w/o Dilate+tSDF

Figure 2. Comparison of reconstruction results with and without dilate and tSDF strategy for hierarchical volume decoding.

seen that dilate+tSDF is mandatory for reconstructing complete mesh without holes.

Implementation and Practical Consideration. The overall pseudocode for hierarchical volume decoding is shown in Algorithm 1. In practice, we set the tSDF threshold $\eta = 0.95$ and the isosurface threshold $\gamma = 0.0$. The dilate operation is implemented using a 3D convolution with a kernel size of 3. At the final resolution, the total number of points increases significantly, thus the `FindNear` operation would introduce many redundant points. To address this, we omit the `FindNear` operation while `Expand` twice, striking a balance between speed and quality. Practically, we find this strategy has minimal impact on the overall quality while speeding up slightly.

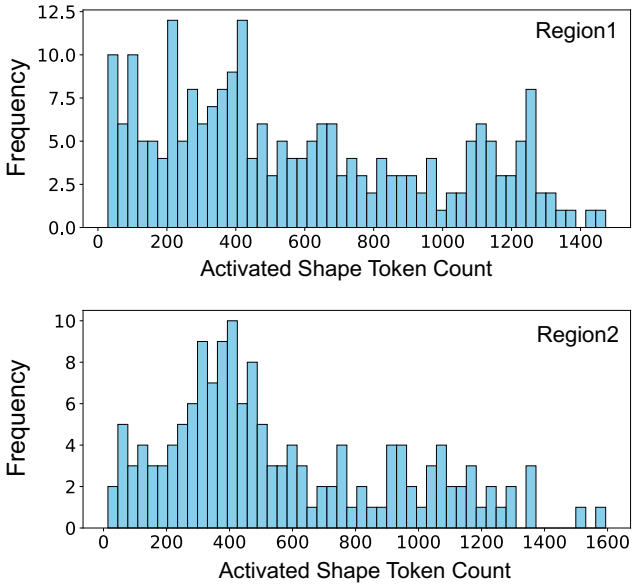


Figure 3. Histogram of activated token counts within different regions, measured with 300 cases.

D. Details of Adative KV Selection

D.1. Analysis of Locality.

Activated Tokens Across Different Cases in the Same Region. Fig. 3 presents a histogram of the activated shape token count across 300 different test cases. As observed, different regions and cases activate different sets of tokens. This suggests that locality is case-dependent rather than region-dependent. In other words, the same region in different cases does not consistently share a similar set of activated tokens.

Distribution of Activated Tokens Within a Case. Fig. 4 shows the histogram of the total number of activated tokens within a case, based on 200 cases. It can be observed that most cases contain over 3000 tokens, with a maximum of 3072 tokens. This further confirms that the phenomenon of having fewer activated tokens per region is due to token locality, rather than token redundancy.

IoU Changes with Respect to TopK Tokens. Fig. 5 shows the relationship between volume IoU and the number of TopK tokens, with all methods utilizing hierarchical volume decoding. The results for Original and FlashVDM differ due to the use of the efficient decoder. It can be observed that FlashVDM(r4) closely matches the curve of Original(r4), suggesting that our efficient decoder design preserves most of the reconstruction ability. Additionally, we notice that r16 performs significantly better than r4, highlighting the strong locality of attention between queries and shape tokens. Higher resolution corresponds to smaller subvolumes, resulting in improved locality. Interestingly, r16 maintains a similar IoU even with just 16% (512/3072)

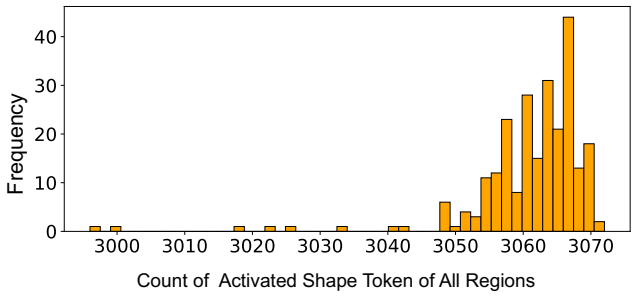


Figure 4. Histogram of the number of total activated token within all regions, measured with 200 cases.

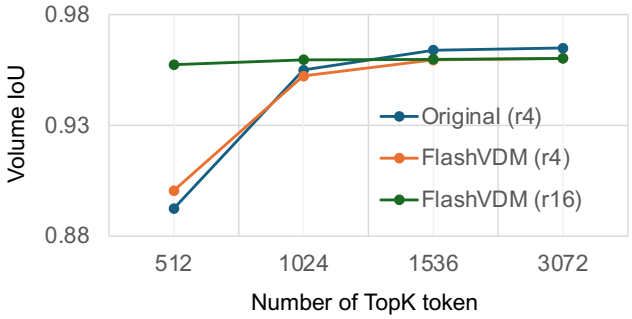


Figure 5. The graph shows the relationship between volume IoU and the number of TopK tokens. r4 denotes the volume is divided into 4^3 subvolumes, and r16 denotes 16^3 subvolumes.

of the tokens.

D.2. Implementation

Combination with Hierarchical Volume Decoding. In Section 3.2, we briefly introduce the combination of Adaptive KV Selection (AKVS) and hierarchical volume decoding. Here, we provide a more detailed explanation of the implementation and background. AKVS can be naively implemented as shown in Algorithm 2. The algorithm consists of four main steps: sampling queries, computing the mean attention score, selecting Top-K, and performing attention. Since the attention score is computed from the sampled queries, we need to feed the queries subvolume by subvolume, with queries being spatially close to one another, to keep locality.

For the original volume decoding, this can be easily achieved by changing the chunk-splitting method to a subvolume-splitting method, as the original method also uses chunk-splitting to reduce memory requirements. However, to maintain locality, the chunk size must be much smaller, which may be too small for efficient GPU acceleration. Additionally, with hierarchical volume decoding, the number of queries in each subvolume can be even smaller. To address this, we propose to pre-divide the subvolume and concatenate all queries of each subvolume. Instead of processing each subvolume sequentially, we concatenate multiple subvolumes and process them in parallel until cross-attention is reached. This approach helps reduce the running time for MLP and other linear layers in the decoder.

Algorithm 2 Adaptive KV Selection.

Input: Query $Q \in \mathcal{R}^{N\times D}$, Key $K \in \mathcal{R}^{M\times D}$, and Value $V \in \mathcal{R}^{M\times D}$ of cross attention, the number of queries $n \ll N$ for estimating TopK correlated KV.
Output: Attention result $O \in \mathcal{R}^{N\times D}$.

$\hat{Q} = \text{Sample}(Q), \hat{Q} \in \mathcal{R}^{n\times D}$
$\text{M} = \text{Mean}(\hat{Q} \times \hat{K}^T), M \in \mathcal{R}^{n\times M}$
$\hat{K}, \hat{V} = \text{TopK}(M, K, V), \hat{K}, \hat{V} \in \mathcal{R}^{k\times D}$
$O = \text{Attention}(Q, \hat{K}, \hat{V})$
return O

E. Details of Diffusion Distillation

In Section 4.3, we provide a brief ablation study of the proposed progressive flow distillation with a case study. Here, we present a more detailed comparison with additional test cases and also include ablations of Huber loss and Phase 1 fine-tuning.

Guidance Distillation Warmup. Fig. 6 shows the results without guidance distillation warmup. We observe significant degradation in results when guidance distillation is omitted, confirming the effectiveness of our strategy.

Huber Loss vs L2 Loss. Fig. 7 shows a visual comparison between models trained with L2 and Huber loss. While L2 loss generates reasonable results, the quality is noticeably inferior to that of the model trained with Huber loss. For example, certain structures, like the radio and several houses, are broken in the L2 model. We hypothesize that it is because Huber loss is less sensitive to outliers, thus stabilizing the training and improving the results.

EMA of Target Network. Fig. 8 compares models trained with and without EMA. Both models were fine-tuned from a guidance-distilled model using consistency flow distillation, with no Phase 1 or adversarial fine-tuning. It can be seen that the meshes are broken without EMA, highlighting the importance of EMA for stability.

Phase 1 Fine-tuning. During consistency flow distillation, we follow PCM [4] to divide the total trajectory into 5 phases and force the model to predict different targets at each phase. However, there is a training-test gap as the model needs to predict final target during the inference. To address this, we propose Phase 1 fine-tuning after Phase 5 pretraining. We empirically find that this strategy slightly improves performance, as shown in Fig. 9.

Adversarial Fine-tuning. The comparison between models with and without adversarial fine-tuning is shown in Fig. 10. It is evident that adversarial fine-tuning helps

improve surface smoothness, corrects detail generation, and fixes mesh holes.

Effect of Sampling Steps. As shown in Fig. 11, our method demonstrates the ability to generate rough results with just 2 steps, and simple objects can be effectively generated within 3 steps.

F. More Results

Shape Generation Results. Fig. 12 presents a set of shape generation results from Hunyuan3D-2 Turbo, which has been distilled using the proposed FlashVDM framework. Our model achieves fast generation with only 5 diffusion sampling steps and ultra-fast decoding, while maintaining high-quality meshes across a variety of categories.

Compatibility with Texture Generation. Fig. 13 showcases texture generation results for meshes produced by Hunyuan3D-2 Turbo, distilled with FlashVDM. It is evident that the meshes generated by our method are fully compatible with texture generation, demonstrating its versatility.

Comparison with Other Methods. Fig. 14 compares FlashVDM with other fast 3D generation methods. The results highlight that our method consistently outperforms existing approaches across a broad range of input types.

G. Limitations and Future Works.

In this work, we have significantly accelerated both VAE decoding and diffusion sampling. Despite these improvements, there are still areas that could be further enhanced. For instance, our PyTorch implementation contains several indexing operations, which can slow down the GPU pipeline. Operator fusion and more efficient memory access strategies could be promising directions for optimization. Additionally, exploration of locality of vecset would also be an interesting direction. Regarding diffusion sampling, single-stage distillation may be preferable, as the current multi-stage approach is complex and introduces cascade errors, which limit its performance potential. Furthermore, while our investigation of adversarial finetuning shows promising results, further research could focus on continuously utilizing real 3D data with adversarial finetuning or even reinforcement learning, a direction we believe holds significant promise. Lastly, as VAE inference time is reduced, the proportion of time spent on diffusion sampling increases. This suggests that exploring one-step distillation could be a valuable avenue for future research.

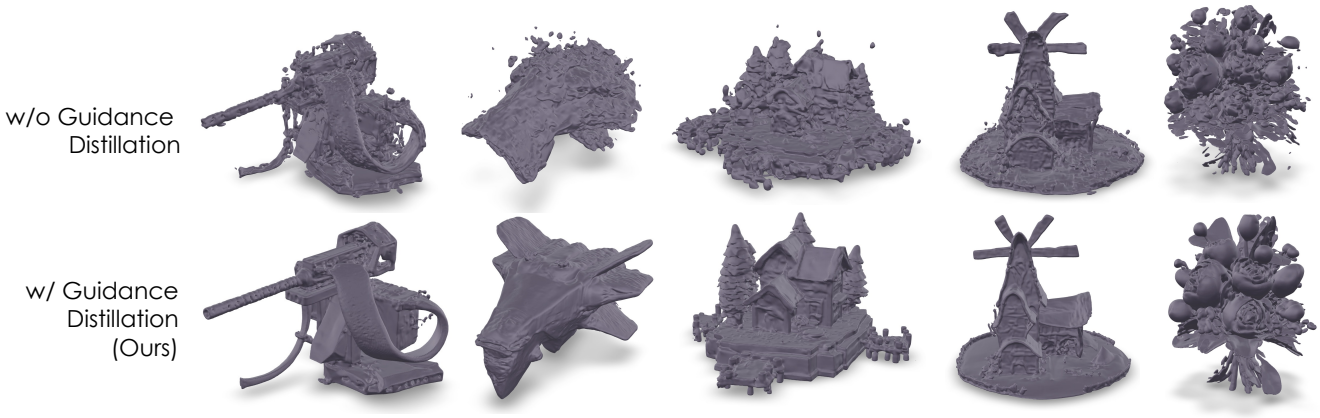


Figure 6. Visual comparison of models **with and without guidance distillation warmup**. The adversarial fine-tuning and Phase1 fine-tuning are not adopted. It demonstrates that the guidance distillation warmup is essential for successful distillation.

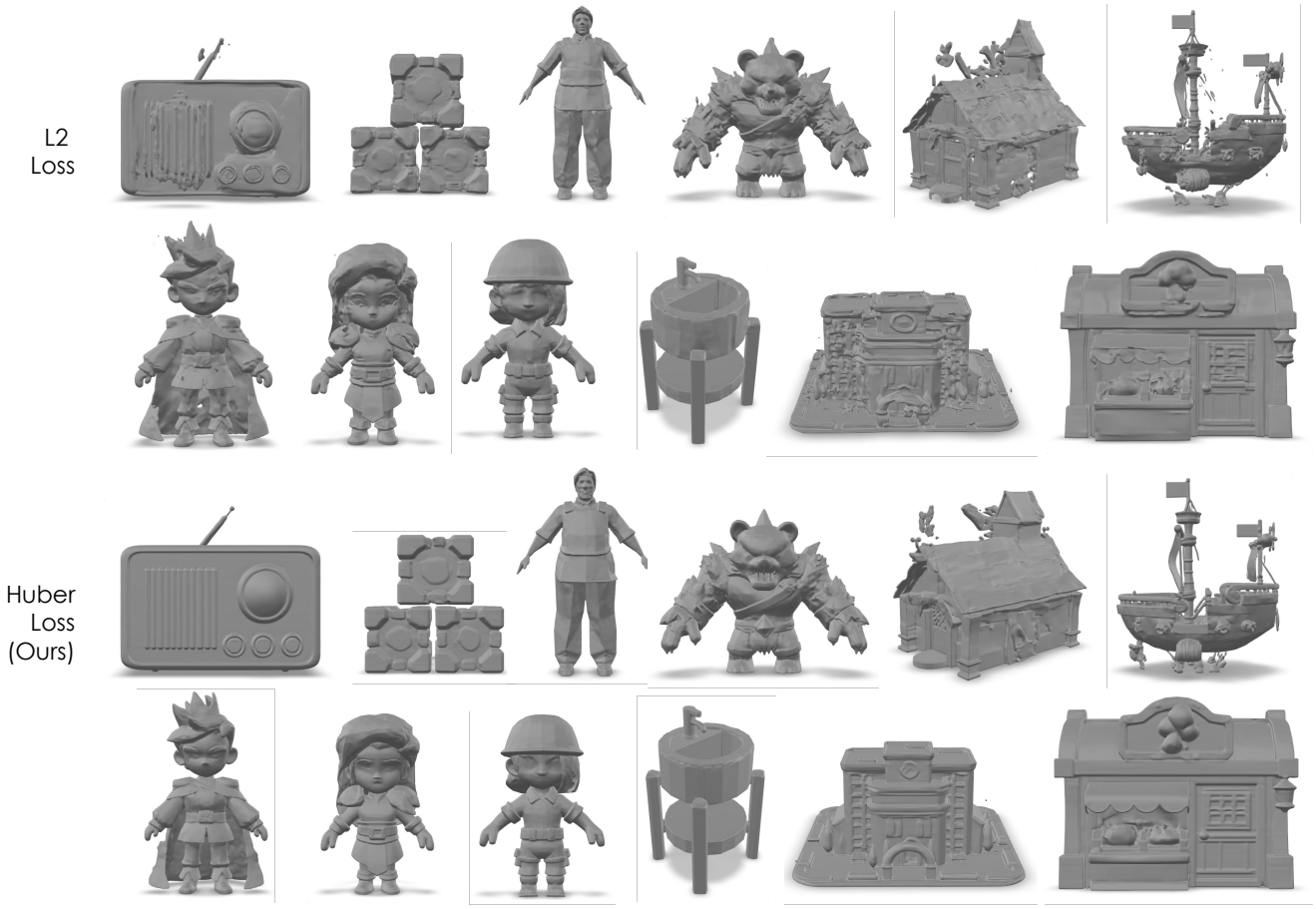


Figure 7. Visual comparison of models trained with **L2 and huber loss**. The adversarial fine-tuning and Phase1 fine-tuning are not adopted. It demonstrates that the huber loss is significantly better than l2 loss, which we hypothesis that is due to huber loss is less sensitive to outliers so that stablizes the training and makes results better.

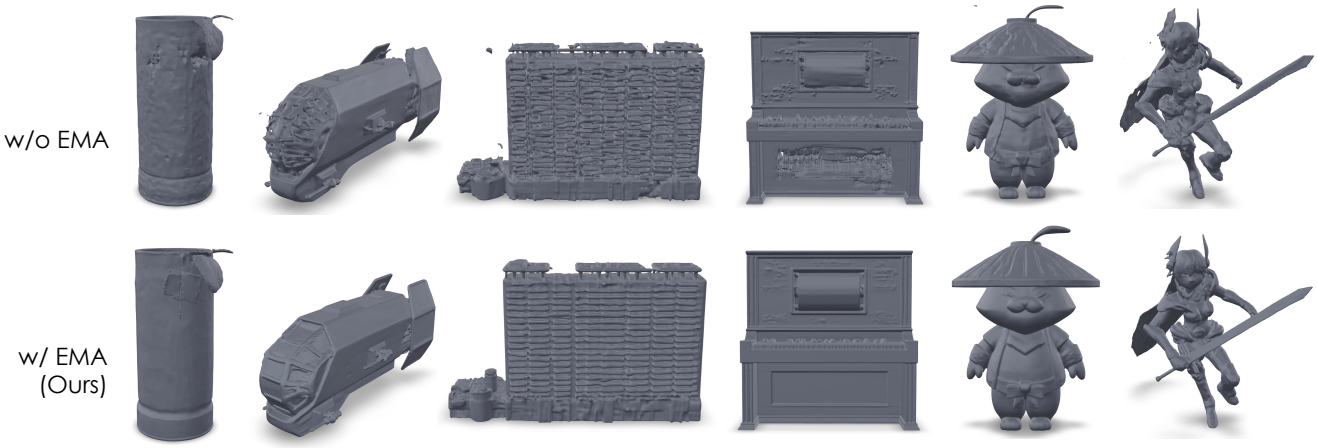


Figure 8. Visual comparison of models trained **with and without EMA**. The adversarial fine-tuning and Phase1 fine-tuning are not adopted. It demonstrates that the meshes tend to be broken without EMA.

Figure 9. Visual comparison of models **with and without guidance distillation warmup**. The adversarial fine-tuning and Phase1 fine-tuning are not adopted. It demonstrates that the guidance distillation warmup is essential for successful distillation.

Figure 10. Visual comparison of models **with and without adversarial finetuning**. All other distillation stages are used. It demonstrates that the predicted meshes are more accurate and smooth after adversarial finetuning.

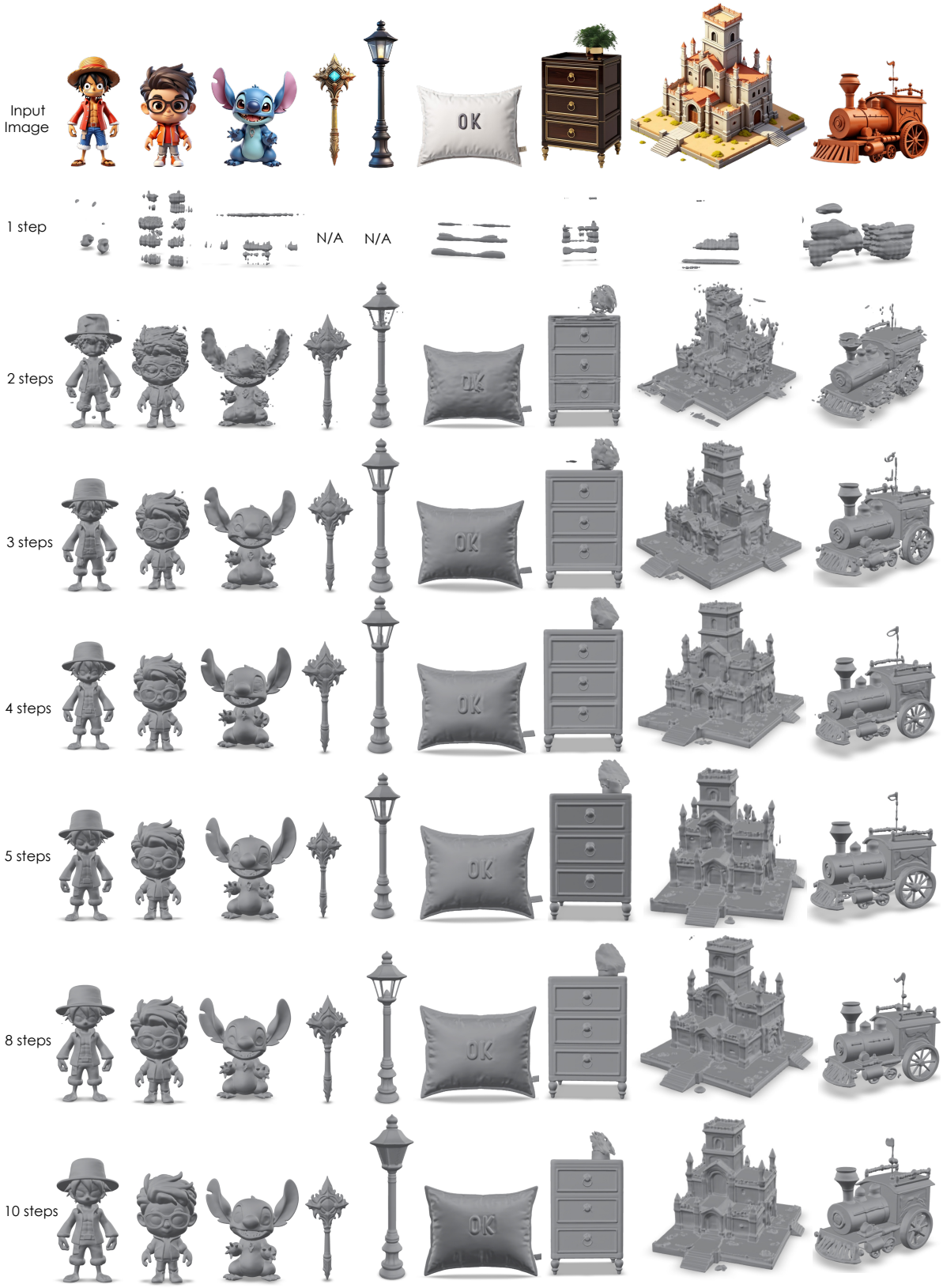


Figure 11. Visual comparison of FlashVDM generation results with different sampling steps.

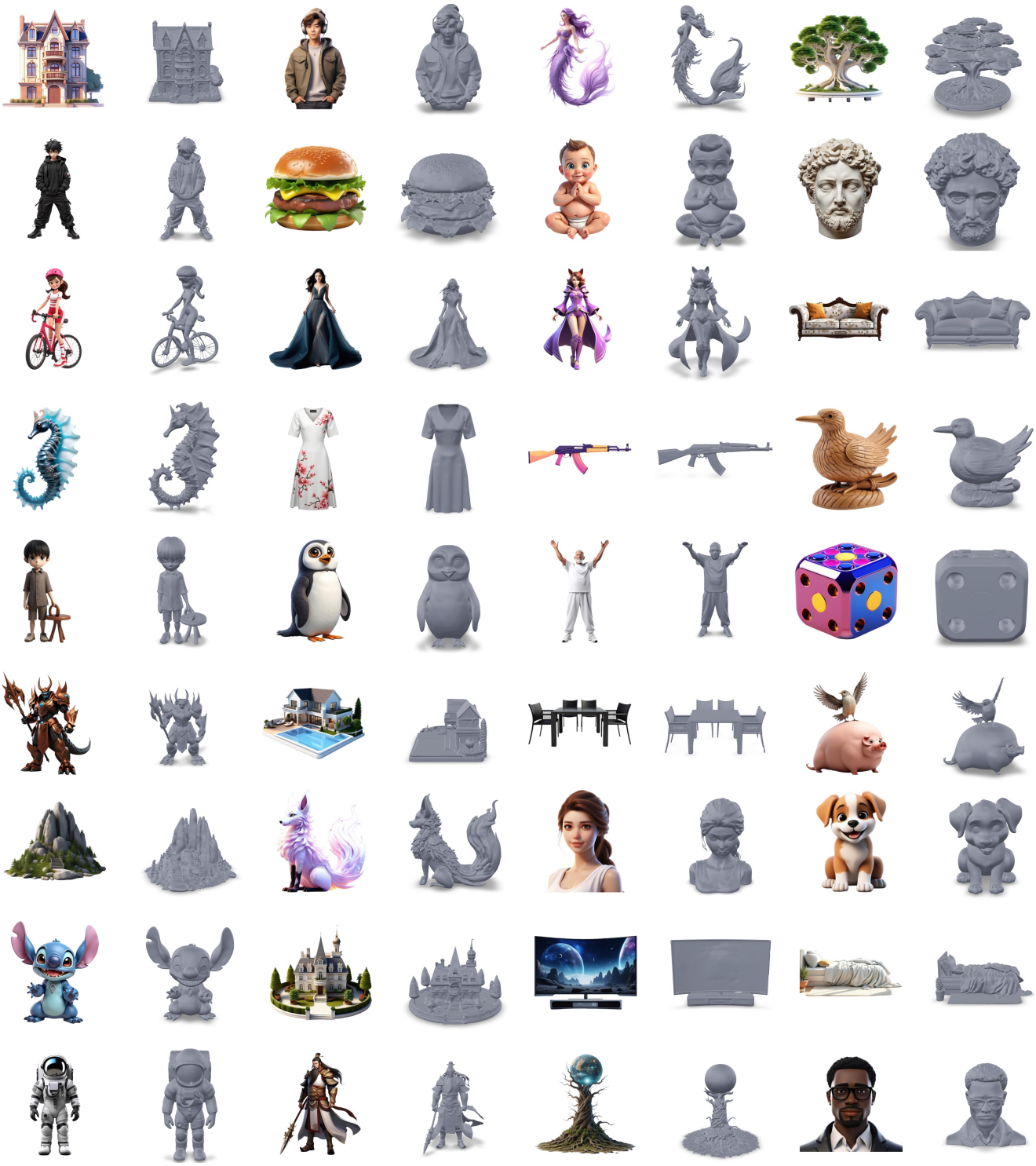


Figure 12. Shape generation results of Hunyuan3D-2 Turbo distilled with the proposed FlashVDM. Image prompts are generated by HunyuanDiT [1]. The number of inference steps is 5.

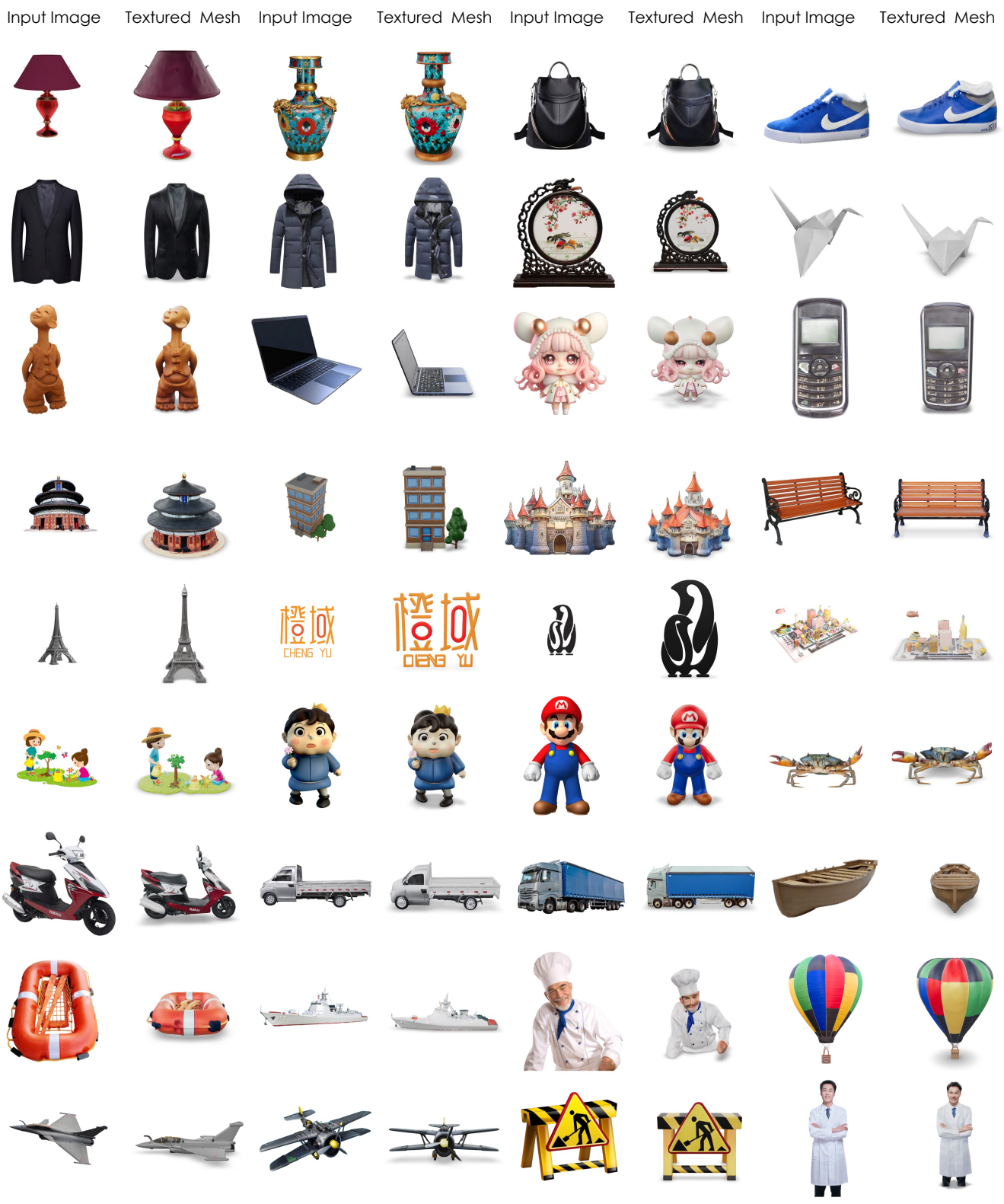


Figure 13. Texture generation results of Hunyuan3D-2 Turbo distilled with the proposed FlashVDM and Hunyuan3D-Paint-2 [5]. Image prompts are generated by HunyuanDiT [1]. The number of inference steps is 5.

Figure 14. Comparison between FlashVDM (Hunyuan3D-2 Turbo) 5 steps and other fast 3D generation methods.

References

[1] Zhimin Li, Jianwei Zhang, Qin Lin, Jiangfeng Xiong, Yanxin Long, Xinchi Deng, Yingfang Zhang, Xingchao Liu, Minbin Huang, Zedong Xiao, Dayou Chen, Jiajun He, Jiahao Li, Wenyue Li, Chen Zhang, Rongwei Quan, Jianxiang Lu, Jiabin Huang, Xiaoyan Yuan, Xiaoxiao Zheng, Yixuan Li, Jihong Zhang, Chao Zhang, Meng Chen, Jie Liu, Zheng Fang, Weiyan Wang, Jinbao Xue, Yangyu Tao, Jianchen Zhu, Kai Liu, Sihuan Lin, Yifu Sun, Yun Li, Dongdong Wang, Mingtao Chen, Zhichao Hu, Xiao Xiao, Yan Chen, Yuhong Liu, Wei Liu, Di Wang, Yong Yang, Jie Jiang, and Qinglin Lu. Hunyuan-dit: A powerful multi-resolution diffusion transformer with fine-grained chinese understanding, 2024. 9, 10

[2] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023. 1

[3] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*, 2023. 1

[4] Fu-Yun Wang, Zhaoyang Huang, Alexander Bergman, Dazhong Shen, Peng Gao, Michael Lingelbach, Keqiang Sun, Weikang Bian, Guanglu Song, Yu Liu, et al. Phased consistency models. *Advances in Neural Information Processing Systems*, 37:83951–84009, 2025. 1, 3

[5] Shengyu Zhao, Jonathan Cui, Yilun Sheng, Yue Dong, Xiao Liang, Eric I Chang, and Yan Xu. Large scale image completion via co-modulated generative adversarial networks. In *ICLR*, 2021. 10

[6] Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025. 1