UniTEX: Universal High Fidelity Generative Texturing for 3D Shapes

Supplementary Material

In this appendix, we first provide additional implementation details in Sec. A. We also clarify some evaluation details in Sec. A.1. Then, we give more results in Sec. B, and the data curation pipeline of our work is shown in Sec. C. We discuss the limitations and future work in Sec. D.

A. More Implementation

LTM: We implement our Large Texturing Model (LTM) using a transformer-based architecture with 16 attention heads, a model width of 1024, and 24 decoder layers. It contains about 400M parameters. The model is trained on 16 A100 GPUs for 300,000 iterations with a learning rate of 1e-5. For ablation studies, we conduct training for 40,000 iterations with a learning rate of 5e-5 with the AdamW optimizer.

Flux: We fine-tune the Flux backbone using LoRA with a rank of 16 and a scaling factor of 16. The training adopts the Prodigy optimizer ($lr = 1$) with mixed-precision training in bfloat16. The input resolution is set to 512×3072, with 1 row and 6 columns used for multi-view conditioning. Control images are provided as additional input guidance, and 50% of tokens are randomly dropped during training to improve robustness and regularization.

All models used for quantitative evaluation are trained for 2,000 iterations, which typically takes one day on our setup. This setting ensures a fair comparison with other models used in ablation studies. However, one thing should be noted is that, unlike conventional training regimes where longer training leads to better performance, we observe that Flux LoRA does not necessarily benefit from longer training.

Color in Invalid UV Areas The colors in the invalid UV areas (see Fig. 2 in the main text) are a deliberate result of a post-processing step designed to prevent rendering artifacts. After filling invisible UV regions via LTM, we dilate the valid UV colors outward to fill the entire map. This process, often known as texture padding or bleeding, mitigates potential seams caused by texture filtering (e.g., mipmapping) at the boundaries of UV islands.

A.1. Evaluation Details

Evaluation cases. As mentioned in the main paper, our method sets two benchmarks. For artist-created meshes, we randomly select 78 meshes from Objaverse [2] and the test list provided by MetaTexGEN [1]. For generative meshes, we select 30 images, which are used to generate 30 instances from two 3D generative models for texturing. This enables us to fully evaluate the texture generation ability in various real-world applications.

Implementation details of Image-Based Criteria. To evaluate the texture quality, we introduce an image-based evaluation protocol that operates in the visual space. Specifically, the object is normalized to the range $[-1, 1]$, and the camera trajectory is defined as a circular path with a fixed radius of 2.5. The camera elevation is linearly interpolated between -1.45 and 1.45, resulting in 24 evenly spaced views rendered per object. These multi-view images are then used to assess the visual fidelity and consistency of the generated textures.

User Study. To quantitatively assess the perceptual quality of our generated textures, we conducted a comprehensive user study. We developed a custom web-based interface (shown in Fig. 1) that presented participants with an input image alongside the textured 3D models produced by various methods. To ensure an unbiased assessment, the identities of all methods were concealed, and participants were instructed to select the result they considered to be the most visually appealing. We collected evaluations from 20 distinct individuals, each assessing 20 sets of results. For metric calculation, we adopted a "winner-takes-all" approach, tallying the proportion of times each method was chosen as the most preferred. As detailed in Table 1 in the main text, our method achieved a remarkable user preference score of 65.91%. This significantly outperforms the second-best method, Hunyuan3D-Paint, which scored 21.36%, representing a lead of over 40 percentage points. These findings robustly affirm the superior perceptual quality of our generated textures.

B. More Experiments

B.1. UV-based methods under different unfolding

We conducted an additional comparison to validate robustness against topological ambiguity. Specifically, we compared our method with TexGen using meshes from TexGen's repository under two UV parameterizations, xatlas and Blender. Our method yields consistent results under both settings (cmmd: 0.247, clip-score: 0.862), while TexGen (trained on xatlas-based image-UV pairs) shows a clear performance drop: with xatlas, it scores (cmmd: 0.262, clip-score: 0.880), and with Blender UVs, it scores (cmmd:

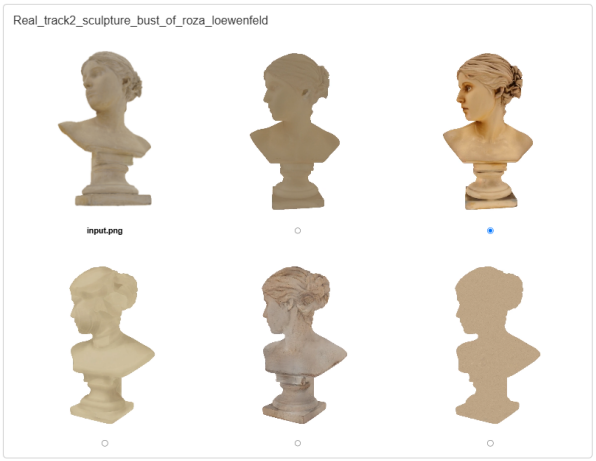


Figure 1. The example interface of our user-study experiments.

Table 1. Ablation studies on effectiveness of different uv parameterization.

Methods.	CMMD↓	clip-score↑
TexGen w xatlas	0.262	0.880
TexGen w blender	1.050	0.806
Ours w both	0.247	0.862

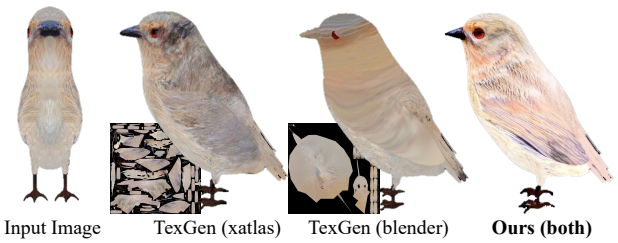


Figure 2. Generated textures under different UV-unfoldings. TexGen performs well with xatlas (used for training) but degrades with Blender UV-unfolding, whereas our method remains robust across various UV parameterizations.

1.050, clip-score: 0.806). This supports our claim that topological ambiguity affects UV-based methods, whereas ours is robust.

Moreover, we provide a visualization case illustrating the effect of UV-based methods under different unfolding strategies, as shown in Fig. 2. The mesh (bird) shares the same geometry, but when applying different UV-unfolding algorithms—xatlas (used during training) and Blender (unseen during training)—the performance of TexGen drops significantly. In contrast, our method remains robust across different UV parameterizations.

Table 2. Ablation studies of normal estimation using drop training strategy.

Metrics.	Hunyuan2.0-Paint		finetuned FLUX(Ours)	
	wo LTM	w LTM	wo LTM	w LTM
$PSNR_{UV}$	11.67	12.38 (+0.71)	13.75	14.54 (+0.79)

B.2. Texture Function Visualization

For better understanding, Fig. 3 provides a visual comparison between our proposed texture function and the traditional unsigned distance function (UDF). They are extracted in a similar way; the only difference between UDFs and TFs is as follows: after finding the closest point on the surface, the UDF calculates the distance from the query point to the closest point, while TFs, on the other hand, directly assign the color of the closest point to the query point.

B.3. LTM under different MVD

To evaluate whether our LTM can adapt to results generated by various diffusion models, we conduct experiments on the RealTrack dataset, which provides ground-truth UV maps. This allows us to compute $PSNR_{UV}$ between the ground-truth UV maps and those reconstructed through our full pipeline (using diffusion models with or without LTM) to assess the impact of LTM on texture completion.

Specifically, we test LTM with Hunyuan2.0-Paint and our FLUX model in Sec. 4.4.2 by replacing the ground-truth multi-view images with images generated by these diffusion models.

The results are shown in Tab. 2. These results demonstrate that our LTM consistently enhances the quality of the reconstructed texture maps across different multi-view diffusion models. Moreover, our first-stage model also achieves better overall performance compared with these baselines.

B.4. Drop training strategy in Nomral Estimation

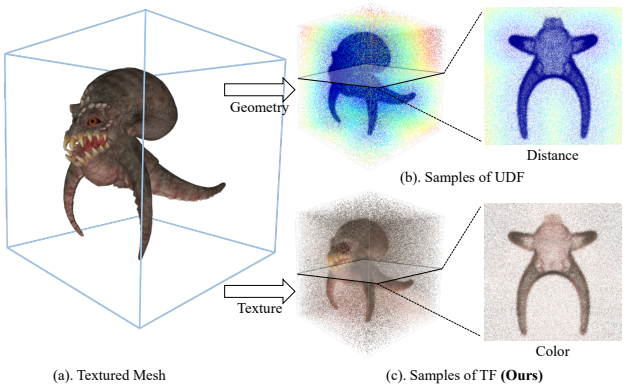
We further introduce the diffusion-based normal estimation task as a complementary evaluation to investigate advanced training strategies for large-scale diffusion transformers.

For the normal estimation task, we set the input condition as multi-view images (6 views) and output normal maps that are normalized in the coordinate system of the first input view. For training, we use the Objaverse LVIS set and simply split 100 cases from the training set as a validation set. We follow the evaluation protocol used by GeoWizard [3].

As shown in Tab. 3, our drop-training strategy attains comparable accuracy on the normal estimation task with significantly improved efficiency.

Table 3. Ablation studies of normal estimation using drop training strategy.

Methods.	Normal Estimation			
	mean↓	RMSE ↓	11.25° ↑	22.5° ↑
w/o Drop Training	**22.57**	**29.42**	44.85	**55.82**
w/ Drop Training	22.79	29.79	**45.13**	55.48



(a). Textured Mesh (b). Samples of UDF (c). Samples of TF **(Ours)**

Figure 3. Visualized example of the *Texture Functions* (TF) for represent the texture for the whole 3D space. (a) A textured mesh. (b) Unsigned Distance Function (UDF) samples representing 3D geometry. (c) Inspired by UDF, we define texture as a continuous function over 3D space, enabling volumetric texture representation.

B.5. More Comparisons

In addition to the comparison in the main paper, we further compare our method with the closed-source method by retexturing the generative models, as shown in Fig. 7. **All APIs of commercial models were queried in May 2025.**

B.6. Comparison with SoTA Multi-View Generation

We evaluated our first-stage method on the image-to-multi-view (I2MV) generation task, comparing it against leading state-of-the-art (SoTA) approaches. For quantitative assessment, we adopted the Alignment, Plausibility, and Texture Detail metrics from GPTeval3D [6], which specifically evaluate view consistency and texture quality. The remaining two metrics from the benchmark were omitted as they pertain to geometry, which is outside the scope of this stage. As shown in Tab. 4, our method suppresses existing SoTA techniques on Alignment, Plausibility, and gets a better overall score. Furthermore, the qualitative results in Fig. 5 visually confirm that our generated views exhibit superior consistency and fidelity.

Table 4. MV generation comparison. (test across 66 cases.)

	Alignment	Plausibility	Tex Detail	Overall
MVadapter(I2MV)	968.060	827.932	**1005.266**	933.752
Ours	**1031.940**	**1172.068**	994.734	**1066.247**

B.7. Choice of a Two-Stage versus End-to-End Framework

We compared a two-stage FLUX framework against a single end-to-end (E2E) model for our task. As shown in Fig. 4, we observed that directly training an E2E model (c) is difficult and results in suboptimal performance, especially in extreme visual cases. Conversely, the two-stage approach—comprising a generation stage (a) followed by a delighting stage (b)—demonstrates more robust and favorable outcomes. Since our work prioritizes final output quality over inference efficiency, we opted for the two-stage design.

B.8. More Results

Moreover, we further show additional generative texturing cases in Fig. 8, demonstrating that our method achieves better results.

C. Data Curation

Data Filtering To prepare high-quality training data for mesh texture generation, we filter out 3D assets sourced from Objaverse and Sketchfab. First, we perform basic geometry and texture filtering to eliminate low-polygon meshes and models lacking meaningful texture maps, ensuring a baseline level of visual and structural integrity. Next, we adopt an aesthetic scoring approach inspired by Microsoft/TRELLIS[7], using LAION's[5] aesthetic predictor to evaluate rendered views (front and four canonical angles) of each 3D asset, retaining only those with sufficiently high ratings. We further refine the dataset through metadata-based filtering, leveraging textual cues such as tags, categories, descriptions, and author information from Sketchfab to exclude irrelevant, spam-like, or poorly annotated models. Finally, we analyze the material node structures of glb files to verify adherence to physically based rendering (PBR) principles, ensuring material consistency across the dataset. This approach ensures a balanced dataset with robust geometric, aesthetic, and material properties suitable for training texture generation models.

Texture Function Extraction To generate supervised data used for training, we sample 3D points within a normalized bounding box of $[-1, 1]^3$ and associate each point with its corresponding RGB color value derived from the mesh texture. For each sampled point $\mathbf{p} \in \mathbb{R}^3$, we first compute its nearest projection point $\mathbf{q}$ on the mesh surface using a ray-mesh intersection method[4]. Once $\mathbf{q}$ is obtained, we determine its barycentric coordinates within the associated mesh triangle and interpolate its UV coordinates from the triangle's vertices. The interpolated UV coordinates are then used to query the texture map, yielding the

Figure 4. two flux(a.) then (b.), v.s. one flux (c.)

Figure 5. MV generation comparision.(zoom in for details)

color $\mathbf{c} \in \mathbb{R}^3$ at $\mathbf{q}$. This color is assigned to the original sampled point $\mathbf{p}$, forming the supervised pair $(\mathbf{p}, \mathbf{c})$

D. Limitation & Future work

One key limitation of our method is the increased inference time compared to existing texturing pipelines. Our three-stage texturing pipeline, which includes the large Flux backbone and the geometry-aware LTM refinement, introduces additional computational overhead during both feature extraction and texture generation. Furthermore, unlike UV-based approaches, our large texturing model cannot directly leverage powerful 2D image priors due to its UV-free design; the functional space does not have an explicit connection with the image modality. This limits its ability to exploit mature image-generation models trained in pixel space. In future work, we plan to explore more efficient architectures to reduce computation time and investigate hybrid approaches that integrate image priors without relying on explicit UV maps.

Also, the TF indeed has similar issues to SDF/UDF. However, such edge cases are generally uncommon, and our method is primarily targeted at AI-generated meshes that are almost watertight. So although it would be a potaintial drawback, but it not influence the main application of our method.

References

[1] Raphael Bensadoun, Yanir Kleiman, Idan Azuri, Omri Harosh, Andrea Vedaldi, Natalia Neverova, and Oran Gafni. Meta 3d texturegen: Fast and consistent texture generation for 3d objects. *arXiv preprint arXiv:2407.02430*, 2024. 1

[2] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 1

[3] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9492–9502, 2024. 2

[4] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. *ACM Trans. Graph.*, 21(3):703–712, 2002. 3

[5] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022. 3

[6] Tong Wu and et al. Gpt-4v (ision) is a human-aligned evaluator for text-to-3d generation. In *CVPR*, 2024. 3

[7] 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. *arXiv preprint arXiv:2412.01506*, 2024. 3

Figure 6. Examples of our training data.

Figure 7. More comparison result with close-source methods.

Figure 8. More examples for our identity-preserving 3D content generation.