MVPaint: Synchronized Multi-View Diffusion for Painting Anything 3D

Supplementary Material

A. Detailed Discussion on Related Works

A.1. Baseline Methods

In this section, we review the most representative baseline that we compared during the evaluation and discuss its strengths and weaknesses. Other great close-source methods, such as Meta 3D TextureGen [3], are removed from the comparison scoop. We omit the SOTA method Text2Tex [5] in this report due to the extensive evaluation in previous literature [16, 25].

TEXTure. TEXTure [18] presents a method for generating 3D textures from textual descriptions using a pretrained depth-to-image diffusion model. TEXTure employs an iterative approach to ensure consistent texturing from multiple viewpoints by dividing rendered images into "keep", "refine", and "generate" regions. It supports texture transfer and editing through both text prompts and user input. However, the method can produce global inconsistencies when handling complex geometries or viewpoints that do not fully capture the model, which the authors identify as areas for future improvement.

Paint3D. Paint3D [25] is a two-stage generative method. In the first stage, it utilizes ControlNet to generate textures from individual viewpoints. In the second stage, it innovatively proposes direct texture generation in UV space. Paint3D [25] employs a UV position map as the control signal to train a ControlNet [27], leveraging the generative capabilities of existing diffusion models to produce corresponding textures. However, due to significant differences between texture map samples and image samples, the diffusion network inherently lacks robust texture generation capabilities. Moreover, during the control generation process using the UV position map, the diffusion model tends to assign similar colors to atlas textures that are closer in 2D mapping space, rather than to the 3D space represented by the control channels. As a result, Paint3D [25] performs poorly on most automatically unwrapped complex texture maps, leading to numerous instances of misaligned textures.

SyncMVD. SyncMVD [16] proposes a zero-shot texture generation method. It addresses the limitations of asynchronous diffusion that plague traditional project-and-inpaint techniques. While previous approaches generate textures from individual views without adequate synchronization, leading to inconsistencies, SyncMVD synchronizes the diffusion processes for multi-view generation. This innovative method facilitates early consensus in texture generation by sharing denoised content across overlapping views during each denoising step. As a result, SyncMVD [16] achieves consistent textures that exhibit remarkable details and coherence across various perspectives. We highlight that SyncMVD does not utilize a multi-view diffusion model for generating multi-view images. Instead, SyncMVD retains the iterative camera pose and subject framework generated by single-view diffusion, which often leads to multi-faces problems due to the lack of constraints between multiple views.

A.2. Border Related Methods

In this section, we discuss the other related works that share similar designs that are out of our evaluation's scope.

Meta 3D TextureGen. Meta 3D TextureGen [3] proposes a two-stage texture generation method. In the first stage, normal and position controls are used to generate aesthetically pleasing multi-view images, which are then weighted and projected based on the mesh surface normals and view directions to obtain an initial UV texture map. In the second stage, the process continues in the UV space, where normal and position controls are again employed for inpainting. Finally, texture enhancement is applied to perform super-resolution and enrich texture details. Through this innovative approach, Meta 3D TextureGen can generate high-definition textures that are rich in detail and aesthetically pleasing with Emu [6] base model. However, due to the relatively independent multi-view generation in the first stage, there may be a lack of consistency between the textures across different views and may be potentially vulnerable to the Janus problem. Additionally, the limited number of viewpoints in the first stage can lead to significant unobserved areas for objects with complex occlusions. This poses challenges for the subsequent texture inpainting and enhancement processes, which are performed entirely in UV space. Since the code for Meta 3D TextureGen has not been open-sourced, we do not include it in our experimental comparisons.

Unique3D. Unique3D [21] is an innovative image-to-3D framework capable of efficiently generating high-quality 3D meshes from single-view images, demonstrating both high fidelity and robust generalization capabilities. It integrates multi-view diffusion models, multi-scale upsampling strategies, and the proposed ISOMER algorithm, enabling the generation of detail-rich textured meshes in 30 seconds. Similar to our method, Unique3D uses a coarse-to-fine manner to generate multi-view high-resolution images. Differently, Unique3D targets simultaneous geometry and texture generation, and 3D consistency is encouraged but not strictly aligned. For example, it upscales 256×256 im-

ages with a multi-view ControlNet [27] and further boosts the resolution to 2k with a view-separate upscaling model without any synchronization.

CLAY. CLAY [28] is a large-scale 3D generation model that can produce high-quality 3D geometry and materials from text or image inputs. It employs a multi-resolution VAE and a minimal latent diffusion transformer, supporting various control modalities such as multi-view images and voxels to facilitate precise 3D asset creation. The texturing module of CLAY [28] is also built upon MVDream [19], they modify it by adding additional channels and modalities to support physical-based rendering (PBR), ControlNet [27] to achieve view control, and uses LoRA [11]-based fine-tuning. CLAY [28] generates the 4 orthogonal views of images, and similar to Paint3D [25] inpaint and upscale the PBR images in UV space. We argued inpainting directly on UV space is vulnerable to complicated UV unwrapping, especially the coarse UV is generated from a low coverage of 4 views at a resolution of 256×256 with great loss in detail. Note that CLAY focuses on PBR generation, and it has the image-prompt ability, whereas we more focus on text-to-texture generation. As the implementation is not public available, we exclude CLAY in our evaluation.

B. Implementation Details

B.1. Network

We utilize the MVDream [19] as the base model of T2MV $\mathcal{D}_{\text{MV}}$ of SMG, and we add the control module τ_p and train it with the same training scheme of ControlNet [27]. Different from other controlled MVDream methods [14] which only controls a single view, we densely control multi-view for better shape alignment. For synchronized refinement, we choose the SDXL [17] as the base model for I2I refinement $\mathcal{D}_{\text{I2I}}$, where two pre-trained ControlNets [8, 22], τ_t and τ_g are deployed. During refinement, the per-view latents $z^{\mathbf{I}}$ are with 128×128 resolution, and they are synchronized on $\mathbf{T}_{\text{sync}}$ with 512×512 resolution. In all SMG processes, models are worked on $N = 8$ views with evenly distributed azimuth angle and interleaved elevation of $\pm 30°$.

B.2. Control Strength of I2I Finer Painting

In the Synchronized Multi-view Generation (SMG) process, we employed a refinement model composed of two control modules. The first module, denoted as τ_t, is designed to provide low-resolution multi-view (MV) images that are free of Janus artifacts and maintain multi-view consistency for the refinement stage. The second module, τ_g, offers geometric guidance, ensuring that the multi-view refinement process is fully aligned with the underlying mesh structure. These two components are controlled by the parameters s_t and s_g, which allow for flexible user-defined configurations. If the user is satisfied with the generated MV images, a relatively high s_t can be set to maintain consistency, while s_g can be reduced to minimize divergence, as demonstrated in the case of the 'Blue Hippo' in the Fig. 1. Conversely, if the user is dissatisfied with the MV images or desires more creative diversity, a lower s_t and higher s_g can be used, as shown in the 'Unicorn' case. It is important to note that without any s_t, our refinement model may face Janus issues similar to those observed in SyncMVD [16] and TEXTure [18] illustrated by the 'Blue Hippo' case in Fig. 1.

B.3. Spatial-aware 3D Inpainting

In Sec. 3.2 of the main paper, we briefly described the execution flow of the algorithm. Here, we provide a more detailed explanation of the algorithm. **Spatial-aware 3D Inpainting (S3I)** is designed to inpaint unobserved regions after multi-view projection. We sample the mesh into a dense point cloud, enabling us to apply coloring at the point cloud level to capture structural information. S3I is a learning-free method that propagates color from observed regions to unobserved regions.

Since the propagation is based on the k-nearest neighbors (KNN) algorithm, there can be errors in propagation across planes, particularly in seam areas. To address this, we calculate the normal vector for each point, incorporating it into the weight calculation to prevent color diffusion across non-coplanar surfaces. The pseudocode for the detailed algorithm execution is as follows:

B.4. Spatial-Aware Seam-Smoothing Algorithm.

In Sec. 3.3 of the main paper, we introduce the implementation concept of the **Spatial-Aware Seam-Smoothing** algorithm. It is used to correct color discontinuities at seams after super-resolution in the UV space. Similar to S3I, it employs k-nearest neighbor (KNN) search to perform weighted color averaging. The detailed pseudocode for the algorithm is as follows.

B.5. Unprojection Reduction Algorithm

After generating consistent multi-view images, we apply weighted projection to obtain the texture UV map. However, due to occlusion in 3D objects, this can lead to projection errors and artifacts in the UV map, as shown in Figure 3. To address this, we determine occlusion relationships and reduced the affected projection areas. The specific algorithm workflow is as follows: we first extract the 3D coordinates for each valid pixel in the UV map, generating a 3D point cloud. Then, using the occlusion detection algorithm proposed in [12], we mark occluded points for each view. Finally, these occlusion marks are mapped back onto the UV map, and regions marked as "occluded" are excluded from projection.

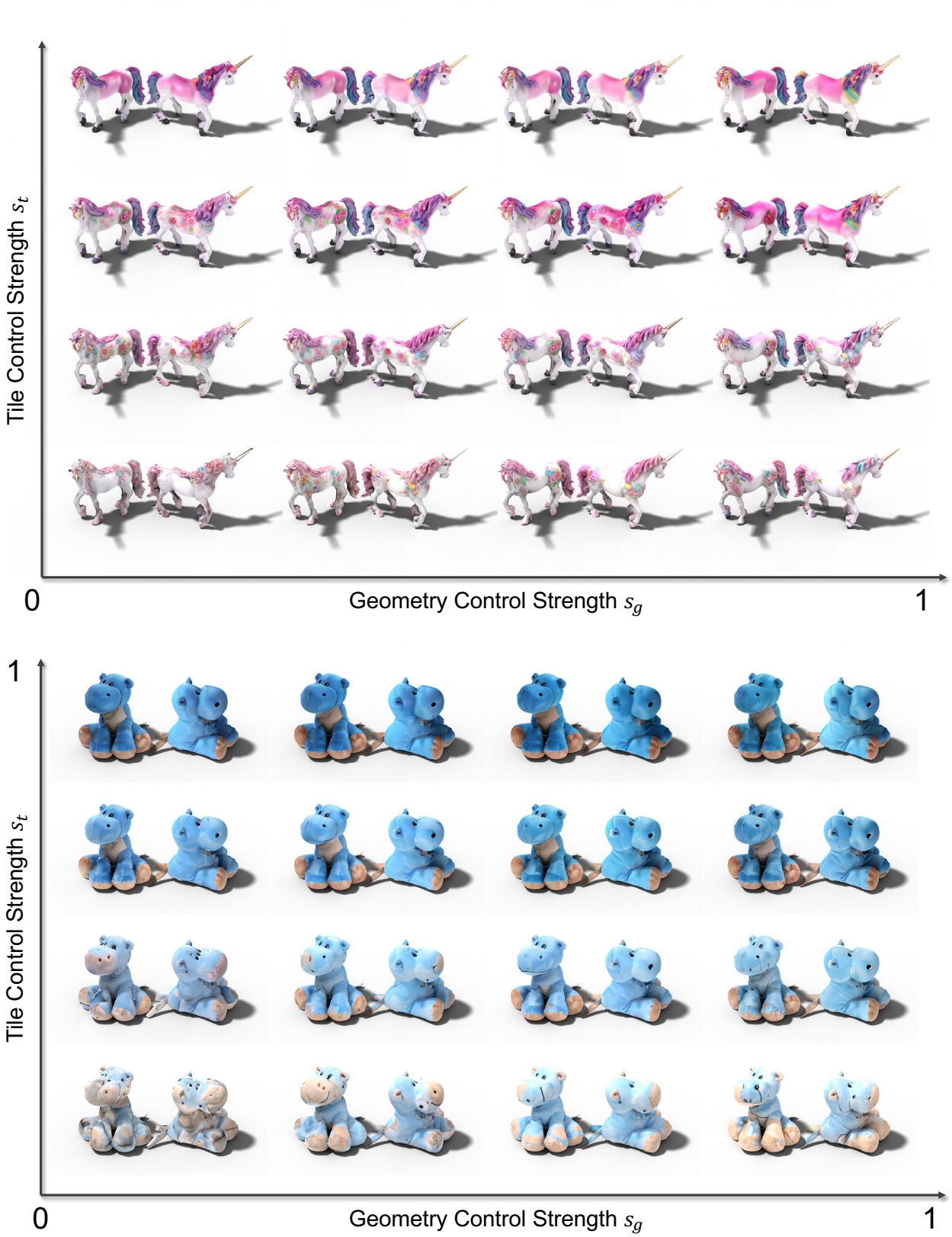

Figure 1. **High Flexibility in Proposed SMG Design.** With two refinement modules τ_g, τ_t and their corresponding control strength s_g, s_t, MVPaint can provide user versatile choices of texture generation. With a larger strength of s_t, the generated results will have more alignment with coarse MV images. With a larger strength of s_g, the generated results will have more creativity (see the 'Unicorn' case) while with a higher risk of Janus problem (see the 'Blue Hippo' case).

Algorithm 1: Spatial-aware 3D Inpainting

Input: *colored_points*: A set of points with color information
color_mask: boolean array where *true* indicates points with valid color
n: number of nearest neighbors for KNN search
Output: *updated_colored_points*: A set of points with updated color information
Function `update_colored_points` (*colored_points, colored_mask*) **:**

```
points ← colored_points.points
colors ← colored_points[color_mask].colors
normals ← calculate surface normals of points
unknown_points ← points[color_mask]
unknown_normals ← normals[color_mask]
tree ← KDTree(points, n)
distances, indices ← tree(unknown_points)
neighbors_normals ← Index_Select(normals, indices)
cos ← Cosine_Similarity(unknown_normals, neighbors_normals)
distance_score ← Normalize(1 / distances)
weight ← cos * distance_score
coloring round ← 0
while stage == "uncolored" or coloring_round > 0 do
    for point in unknown_points do
        neighbors ← KNN(points, n)
        new_color ← Weighted-Average(weight, neighbors)
        colored_points.assign_color(point, new_color)
    Calculate total number of colored points
    if coloring progress then
        Increment coloring round
    else
        Decrease coloring round or exit loop if no further progress
return colored_points
```

Algorithm 2: KNN Seam Smoothing Algorithm

Input: *colored_points*: A set of points with color information, where each row is $[xyzrgb]$
seam_mask: boolean array where true indicates seam points
n: number of nearest neighbors for KNN search
Output: *new_color*: Smoothed color for seam points
Function `knn_seam_smooth` (*colored_points, seam_mask, n_neighbors*) **:**

```
non_seam_points ← colored_points[¬seam_mask] normals ← calculate surface normals of colored_points
seam_normals ← normals[seam_mask]
non_seam_normals ← normals[¬seam_mask]
colors ← non_seam_points.colors
tree ← KDTree(non_seam_points, n)
distances, indices ← tree(seam_points)
seam_neighbors_normals ← Index_Select(non_seam_normals, indices)
cos ← Cosine_Similarity(seam_normals, seam_neighbors_normals)
distance_score ← Normalize(1 / distances)
weight ← cos * distance_score
for point in seam_points do
    neighbors ← KNN(colored_points, n)
    new_color ← Weighted-Average(weight, neighbors.colors)
    colored_points.assign_color(point, new_color)
return colored_points.colors
```

B.6. Discussion on UV Space Tiling

Depending on the specific generation requirements, our UV upscale model $\mathcal{D}_{\text{UP}}$ can be replaced by a refinement network $\mathcal{D}_{\text{TILE}}$. Similar to the tiling network in the second stages in Paint3D [25] and Meta 3D TextureGen [3], $\mathcal{D}_{\text{TILE}}$ is a diffusion model controlled by a tiling control module τ_{tile} and a position map control module τ_{pos}. The position map is a UV map $\mathbf{T}_{\text{pose}}$ where the 3D position of the corresponding mesh surface replaces the channel values. Formally, this optional module can be written as

$$\mathbf{T}_{\text{TILE}} = \mathcal{D}_{\text{TILE}}(z^{\text{UV}}, \mathbf{T}_{\text{c}}, \mathbf{T}_{\text{pose}}; \tau_{\text{tile}}, \tau_{\text{pos}}). \quad (1)$$

Different from previous literature [3, 25], we find the UV tiling process is very vulnerable to continuity or the UV wrapping, leading to obvious seams on 3D mesh when UV atlas is packed randomly like Xatlas [24]. Thus, we only treat this module as an opt for $\mathcal{D}_{\text{UP}}$ if users want to pursue extreme details. We give an example of a comparison of UV upscaling and tiling in Fig. 2, where tiling can increase the upper bound of generation quality, while still suffering from stability. Thus, we use $\mathcal{D}_{\text{UP}}$ as our default super-resolution model, and it is very important to point out that all the results in this work are generated by $\mathcal{D}_{\text{UP}}$.

C. Detailed Evaluations

C.1. Detailed Evaluation Settings

Evaluation Elevation Selection. During the comparative experiments, we observed that each method predefined its

optimal elevation angle, complicating the selection of a unified rendering perspective for comparison. Our approach involved setting the elevation angle for Paint3D and TEXTure to 30°, while SyncMVD utilized a rendering perspective comprising eight evenly distributed views at 0°and two views at ±60°, which we retained. Our proposed method also employed four views at 30°and four views at −30°. Consequently, we selected 15° as the testing elevation and densely rendered 16 perspectives to ensure the validity of the evaluation. We also conduct additional ablation study on such setting design in Sec. D.1.

Figure 2. **Discussion on UV upscale or tiling.** Given groundtruth low-resolution UV map (first row), UV upscaling $\mathcal{D}_{\text{UP}}$ generates sharp and clean textures (second row), UV tiling $\mathcal{D}_{\text{TILE}}$ will add intricate but acceptable details (the left case in the third row) or wrong details (the right case in the third row).

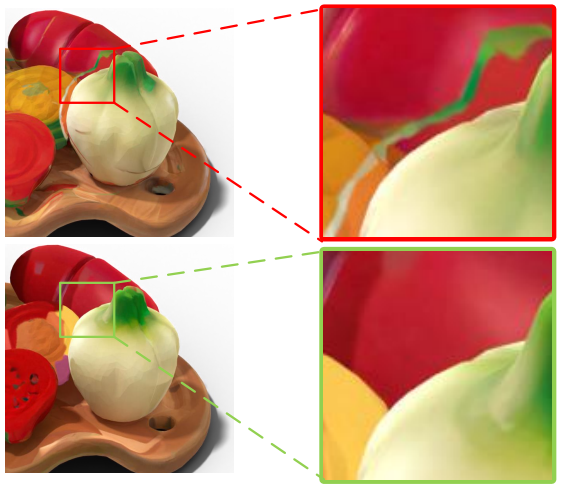

Figure 3. **Illustration of Projection Error**. When the Unprojection Reduction Algorithm is not used, occlusions may cause the color of the occluding object to be projected onto the occluded areas, resulting in artifacts. However, by applying the Unprojection Reduction Algorithm, this issue can be effectively resolved by preventing incorrect color projections onto occluded regions.

C.2. Quantitative and Qualitative Results

Additional Qualitative Results of T2T Evaluation. We provide additional qualitative results to showcase MVPaint's outcomes, including extra comparisons with baseline methods on border categories in Fig. 7). From the qualitative results, we can conclude that SyncMVD [16] can generate good results on general lifeless objects like shoes, bags, etc. While it has severe Janus problems on generating objects with heads, see the Santa, mouse, and eagle cases. Paint3D [25]'s performance is highly related to UV wrapping or texture complexity. When given easy prompts are given like 'copper cup' or 'black boots', Paint3D [25] can produce decent results when texture complexity is high like 'Santa' or 'eagle' it generates results with artifacts and seams. TEXTure [18] generally generates texture with high image saturation with large-scale artifacts and severe Janus problems, which accords with the user study in the main paper Tab. 1 and Tab. 2 where TEXTure [18] has the least user appealing scores, even though it has good subject evaluation results.

Quantitative Results of Ablation on SMG Designs. We also report the quantitative results of the ablation study on SMG designs on GSO benchmark in Tab. 1. Different from the results in in-domain analysis in Tab. 3 of the main paper where '**w/o MV Diff**' influence the most to the quantitative results, in cross-domain benchmark which mostly consists of scan objects, the influence of each design is much more even. What accords with the Objaverse [7] benchmark results is that the full design achieves the best objective metrics and subjective metrics except for CLIP scores.

Table 1. **Quantitative Results on SMG Designs on GSO Benchmark.**

Method	FID↓	KID↓	CLIP↑	User Study		
				Overall ↑	Seamless↑	Consistency↑
w/o MV Sync.	25.72	5.31	**22.49**	3.98	4.19	4.04
w/o MV Paint	25.23	5.17	23.36	4.04	3.99	3.84
w/o Geo. Refinement	25.56	5.27	**22.49**	3.85	3.87	4.14
Full Design	**20.02**	**3.12**	23.25	**4.13**	**4.51**	**4.21**

D. Additional Experiments

D.1. Ablation on View Selection

We conducted ablation experiments on the choice of the number of viewpoints based on the main paper's ablation study. Specifically, we tested several configurations: $N = 4$ with elevations $\phi = 0°$; $N = 8$ with elevation $\phi = 0°$; $N = 16$ with elevation at $\phi = 0°$; $N = 8$ with elevations interleaved $\phi = \pm 30°$; and $N = 16$ with elevations interleaved between $\phi = \pm 30°$. All the view azimuths θ are uni-

Figure 4. **Qualitative results of base model ablation.** Qualitative comparision between default setting (using default base models), and same base model setting on the same case are visualized.

formly distributed in all experiments. Our experiments were conducted on the Objaverse [7] dataset, we use the testing elevation $\phi = 15°$ and evaluation metrics. We organize quantitative results and report in Tab. 2. From the results, we observe that interleaved elevations at $\phi = \pm 30°$ yield better viewpoint coverage, achieving improved metrics at a novel elevation of $15°$. Additionally, we found that increasing N from 4 to 8 results in significant metric improvements, while further increasing to 16 leads to a decline in metrics. This is attributed to the high number of independently generated viewpoints during the refinement phase, which causes excessive overlap and leads to over-smoothed or blurry textures, resulting in lower performance metrics.

Table 2. **Quantitative Results on View Number on Objaverse [7] Benchmark.**

View Setting	FID↓	KID↓	CLIP↑
$N = 4, \phi = 0°$	35.48	11.24	23.21
$N = 8, \phi = 0°$	23.45	4.12	20.48
$N = 16, \phi = 0°$	25.71	4.51	21.45
$N = 8, \phi = \pm 30°$	**20.89**	**3.45**	**19.87**
$N = 16, \phi = \pm 30°$	21.58	3.87	20.21

D.2. Runtime Evaluation

We conducted runtime tests to comprehensively evaluate the proposed method's performance. Initially, we performed a detailed analysis of the average time required for each module, as illustrated in Tab. 3. The total runtime of our pipeline was measured at 97.79 seconds, with the majority of the computational load concentrated in the multi-view refinement step, which accounted for nearly 80% of the overall runtime with SDXL [17] model.

A runtime comparison of our method against several SOTA approaches is also conducted. Experiments were executed on a uniform hardware environment utilizing an single H800 GPU, ensuring consistency in testing conditions. Notably, the model loading time and the rendering time for intermediate results or video outputs were excluded from the reported statistics. As illustrated in the following Tab. 4, despite employing a greater number of steps and higher multi-view resolutions, our runtime remains on par with SOTA methods.

Table 3. **Runtime Evaluation of MVPaint pipeline.**

Stage	I. SMG		II. S3I	III. UVR		Overall
	T2MV	I2I		SR	Fix Seams	
Time (s)	10.51	78.04	0.82	8.30	0.12	97.79

Table 4. **Runtime Comparison between the MVPaint and SOTA methods.**

Methods	TEXTure [18]	Paint3D [25]	SyncMVD [16]	MVPaint
Time (s) ↓	142.52	104.36	**87.32**	97.79

D.3. Ablation on Base Model

To investigate the generality of the proposed method and verify that its performance improvement stems from the pipeline design rather than merely the base model upgrade, we conducted ablation studies on the base model. Specifically, we unified the base model across all methods to SD1.5. In detail, we replaced SDXL in our method with SD1.5 [1] and substituted SD2.1-depth in TEXTure [18] and TexPainter [26] with SD1.5 along with a depth controlnet. We performed the same evaluations on both the Objaverse and GSO benchmarks, and the qualitative results are presented in Tab. 5. The results demonstrate that our method achieves the second-best performance in terms of FID [10] and KID [4] across all settings, even when using the SD1.5 model, second only to our full setting. Additionally, like VCD-Texture [15] and P3G [13] we incorporated the CLIP-Var metric to reflect semantic consistency across different viewpoints, where our method achieves the best

Table 5. **Quantitative results of base model ablation.**

Method	Objaverse Benchmark				GSO Benchmark			
	FID↓	KID↓	CLIP↑	CLIP-Var↑	FID↓	KID↓	CLIP↑	CLIP-Var↑
TEXTure	28.03	7.60	20.30	87.41	24.76	5.50	**23.44**	83.41
Paint3D	25.28	5.19	19.27	84.21	37.29	10.24	21.21	79.81
SyncMVD	26.99	5.72	20.19	85.41	26.96	5.37	23.08	83.47
TexPainter	24.96	5.17	19.71	88.77	28.93	5.97	21.89	83.71
TEXTure-SD1.5	27.61	5.10	**20.42**	**87.82**	24.01	4.44	23.39	83.25
TexPainter-SD1.5	25.62	4.54	19.52	88.72	29.77	6.38	21.96	82.83
Ours-SD1.5	22.54	3.90	19.53	88.27	22.00	3.34	22.29	83.55
Ours	**20.89**	**3.45**	19.87	88.72	**20.02**	**3.12**	23.25	**85.61**

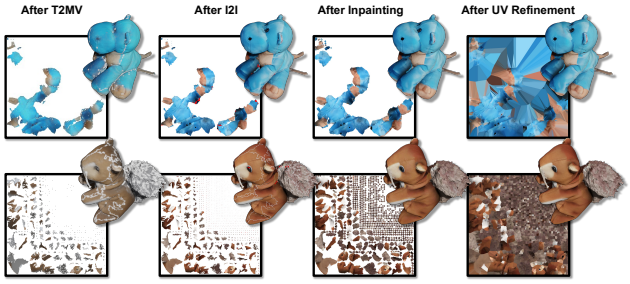

Figure 5. **UV visualization after different stages.** We visualize the generated UV map after each stage. Noted that we do not fill the hole in early stages for better coverage visualization, which causing seams in rendered images.

performance on the GSO benchmark. The qualitative comparisons, shown in Fig. 4, further illustrate that our method retains strong visual quality and avoids multi-head issues even with the SD1.5 base model. Overall, the ablation study validates the effectiveness of our proposed approach.

D.4. UV Visualizations

To better illustrate the generated UV in different stages, we visualized the generated UV map in each intermediate stages in Fig. 5. In the T2MV stage, the generated UV is in low resolution and quality due to the limited resolution of T2MV model, while after I2I stage, the UV maps are more vivid with more details. In both T2MV and I2I stages, the UV coverage is small because of only 8 views are used. After 3D inpainting the missing coverage in UV spaces is filled, and the final stage refine the texture map directly and fill all the empty space in UV. Note that in Fig. 5, we leave the uncovered UV space empty to better illustrate the UV coverage in early stages, so resulting seams in rendered images.

D.5. Synchronization Visualization

We also visualize the results of different synchronization setting in T2MV stage in Fig. 6. Four settings are tested, applying synchronization in the image space and the latent space with 1 step and 40 steps. As from the visualized results, we can observe that synchronizing in latent space usually fails with over-smooth textures due to the small resolution in latent space. However, synchronizing textures in

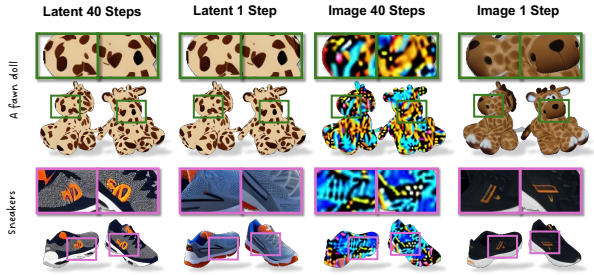

Figure 6. **Visualization of synchronization in T2MV stage.** We visualize the different synchronization setting in T2MV stage, synchronizing in latent/image space with 1/40 steps.

images space with dense steps will generate noisy results, due to the lossy compression of VAE. Thus, we choose to synchronize the mutli-view images in T2MV models in image space for 1 step.

E. Limitations

MVPaint achieves strong spatial consistency and high-resolution textures, but it also has certain limitations, primarily in three areas: the aesthetic problem and the lack of image prompt ability.

Aesthetic Problem. Compared to Meta 3D TextureGen [3], our model exhibits certain deficiencies in the aesthetic quality of texture generation. This can be attributed partly to Meta 3D TextureGen's [3] use of an aesthetically optimized diffusion model, Emu [6], which generates each viewpoint independently. While this approach ensures aesthetic appeal, it may compromise the consistency of the textures. In contrast, our chosen T2MV model, MVDream [19], is based on SD2 [2], which lacks diversity and aesthetic quality in its texture outputs. To address this limitation, we propose the I2I refinement method, which employs a higher-resolution and more aesthetically pleasing model, SDXL [17], to supplement details or re-render the multi-view images. This significantly enhances the aesthetic quality. However, increasing the re-rendering intensity raises the probability of encountering the Janus Problem, thereby limiting our ability to generate highly aesthetically pleasing textures. We believe that the emergence of superior T2MV models in the future will help mitigate the aesthetic shortcomings of our current prototype.

Lack of Image Prompt Capability. Similar to Meta 3D TextureGen [3] and SyncMVD [16], our MVPaint focuses on the generation of textures from text, without emphasizing the optimization of input diversity; consequently, we do not address image prompts in this paper. In contrast, some single-view methods, such as TEXTure [18] and Paint3D [25], leverage existing diffusion models along with plug-and-play image prompt module IP-Adapter [23].

We propose two potential solutions for incorporating image prompts into MVPaint. One approach is to train a Low-Rank Adaptation (LoRA) [11] model akin to CLAY [28], enabling image prompt functionality. Alternatively, we could directly replace the foundational model of T2MV with an image-to-multiview (I2MV) model, such as ImageDream [20].

References

[1] Stability AI. Stable diffusion 1.5. https://huggingface.co/stabilityai/stable-diffusion-2,. 6

[2] Stability AI. Stable diffusion 2. https://huggingface.co/stabilityai/stable-diffusion-2,. 7

[3] 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. In *arXiv preprint arXiv:2407.02430*, 2024. 1, 4, 7

[4] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. In *ICLR*, 2018. 6

[5] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. In *CVPR*, 2023. 1

[6] Xiaoliang Dai, Ji Hou, Chih-Yao Ma, Sam Tsai, Jialiang Wang, Rui Wang, Peizhao Zhang, Simon Vandenhende, Xiaofang Wang, Abhimanyu Dubey, et al. Emu: Enhancing image generation models using photogenic needles in a haystack. In *arXiv preprint arXiv:2309.15807*, 2023. 1, 7

[7] 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 *CVPR*, 2023. 5, 6, 9

[8] Diffusers. Controlnet depth sdxl 1.0. https://huggingface.co/diffusers/controlnet-depth-sdxl-1.0, 2023. 2

[9] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *ICRA*, 2022. 9

[10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017. 6

[11] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *ICML*, 2022. 2, 8

[12] Sagi Katz, Ayellet Tal, and Ronen Basri. Direct visibility of point sets. In *SIGGRAPH*, 2007. 2

[13] Kehan Li, Yanbo Fan, Yang Wu, Zhongqian Sun, Wei Yang, Xiangyang Ji, Li Yuan, and Jie Chen. Learning pseudo 3d guidance for view-consistent texturing with 2d diffusion. In *ECCV*, 2024. 6

[14] Zhiqi Li, Yiming Chen, Lingzhe Zhao, and Peidong Liu. Mvcontrol: Adding conditional control to multi-view diffusion for controllable text-to-3d generation. In *arXiv preprint arXiv:2311.14494*, 2023. 2

[15] Shang Liu, Chaohui Yu, Chenjie Cao, Wen Qian, and Fan Wang. Vcd-texture: Variance alignment based 3d-2d co-denoising for text-guided texturing. In *ECCV*, 2024. 6

[16] Yuxin Liu, Minshan Xie, Hanyuan Liu, and Tien-Tsin Wong. Text-guided texturing by synchronized multi-view diffusion. In *SIGGRAPH Asia*, 2024. 1, 2, 5, 6, 7

[17] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. In *arXiv preprint arXiv:2307.01952*, 2023. 2, 6, 7

[18] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. In *SIGGRAPH*, 2023. 1, 2, 5, 6, 7

[19] Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. In *ICLR*, 2024. 2, 7

[20] Peng Wang and Yichun Shi. Imagedream: Image-prompt multi-view diffusion for 3d generation. In *arXiv preprint arXiv:2312.02201*, 2023. 8

[21] Kailu Wu, Fangfu Liu, Zhihan Cai, Runjie Yan, Hanyang Wang, Yating Hu, Yueqi Duan, and Kaisheng Ma. Unique3d: High-quality and efficient 3d mesh generation from a single image. In *arXiv preprint arXiv:2405.20343*, 2024. 1

[22] Xinsir. Controlnet tile sdxl 1.0. https://huggingface.co/xinsir/controlnet-tile-sdxl-1.0, 2023. 2

[23] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. In *arXiv preprint arXiv:2308.06721*, 2023. 7

[24] Jonathan Young. xatlas: A Library for Mesh Parameterization. GitHub repository, 2018. 4

[25] Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. Paint3d: Paint anything 3d with lighting-less texture diffusion models. In *CVPR*, 2024. 1, 2, 4, 5, 6, 7

[26] Hongkun Zhang, Zherong Pan, Congyi Zhang, Lifeng Zhu, and Xifeng Gao. Texpainter: Generative mesh texturing with multi-view consistency. In *SIGGRAPH*, 2024. 6

[27] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *CVPR*, 2023. 1, 2

[28] Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. Clay: A controllable large-scale generative model for creating high-quality 3d assets. In *ACM TOG*, 2024. 2, 8

Figure 7. **Additional results with border categories.** The 3D models are from GSO [9] and Objaverse [7], and the text prompt is abbreviated.