Generation of Animated 3D Characters

Amir Alimohammadi Leo Luo Roozbeh Razavi

Simon Fraser University

Abstract

In this report, we present a novel approach for the generation of animated 3D characters through a three-phase methodology. Our method begins with the Multi-Frame Novel View Synthesis (MNVS) phase, where we carefully design a network architecture that combines the strengths of Stable Video Diffusion and Zero123++ models. This phase enables the generation of coherent novel view videos, leveraging spatial and temporal priors for accurate frame synthesis from a single input image. In the subsequent phase, we explore the integration of the Score Jacobian Chaining loss within the framework to provide learning signal for novel views in rendering process of Proximity Attention Point Rendering (PAPR) network. Finally, we expand the PAPR framework to Animate-PAPR, enabling the generation of multi-view images for each frame of the video. This involves a freezing and initialization strategy to systematically propagate features across consecutive frames, thus enhancing the network's capacity to produce consistent multi-view renderings throughout the entire video sequence.

1 Method

Our approach comprises three main phases: Multi-Frame Novel View Synthesis (MNVS), employing new loss functions, such as Score Jacobian Chaining, and Generalization to Animated PAPR.

In the initial phase of our approach, we meticulously crafted the MNVS network architecture, drawing upon the strengths of Stable Video Diffusion and Zero123++. This fusion enabled us to generate coherent novel view videos by leveraging the inherent capabilities of these models, which provide both spatial and temporal priors for accurate frame synthesis from a single input image.

While in simple scenarios or when the data quality is very high only training on limited set of view angels are enough, when the ground truth data itself comes from a trained model (like here) we need to provide learning signal for angels that are not in our set of angels. In the subsequent phase, our focus shifted towards exploring the integration of the Score Jacobian Chaining [45] within the framework of the PAPR network [53], aiming to provide learning signal for 3D assets and neural render parameters by using learned 2D score function in a diffusion-based model name Zero123 [23].

In the final segment, we expand the PAPR framework to Animate-PAPR, enabling the generation of multi-view images for each frame of the video. Initially, we train the 3D asset features and the renderer of the PAPR exclusively for the initial frame. Subsequently, we implement a freezing and initialization strategy to systematically propagate the features across consecutive frames, thus enhancing the network's capacity to produce consistent multi-view renderings throughout the entire video sequence. This involves freezing the weights of the renderer (UNet) and initializing the 3D asset's features of the first frame for the subsequent frames, with this iterative process extending to all frames of the video.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

2 Datasets

We employed the well-known Objaverse dataset [6] for training and testing our network. Objaverse presents unique challenges, as many objects lack clear categorization, and rendering them requires additional efforts compared to datasets like ShapeNet [4]. Preparing Objaverse objects involves normalization and adjustment of lighting settings to achieve the desired output.

To create the fine-tuning dataset, we utilized animation samples from Objaverse. By selecting a constant camera coordinate for rendering a video as input, we generated six additional videos from relative camera coordinates specified in Zero123++ [37] and concatenated them as the output. This approach enables fine-tuning of our network.

The decision to use only six constant views rather than providing camera coordinates as input conditions stems from the nature of the Objaverse dataset [6]. Objects in Objaverse do not consistently align in a canonical pose, resulting in a wide range of absolute orientations. Training models on absolute camera poses can lead to orientation ambiguities. Zero-1-to-3 [23], for example, is trained on camera poses with relative azimuth and elevation angles, requiring knowledge of the input view's elevation angle to determine relative poses between novel views. To address this, we adopt fixed absolute elevation angles and relative azimuth angles as novel view poses, eliminating orientation ambiguity without the need for additional elevation estimation. Specifically, the six poses include interleaving elevations of 30° downward and 20° upward, combined with azimuths starting at 30° and increasing by 60° for each pose.

However, in order to train the PAPR [53] model to work on the output frames of the video, a much larger data sample is required to fine-tune the model. Hence the animation sample is then filtered through a blender pipeline to obtain 500 distinct viewpoints every 5 frames, split amongst training, testing, and validation sets with depth maps are also included for the test sets. Camera angle and distance parameters are calculated to entire the model is fully captured for every image, before saving to a NeRF json format [27] for every training, test, and validation set for every rendered frame.

3 Experiments

3.1 Phase 1

In this phase, we commenced by studying the Zero123 code base [23]. After gaining a comprehensive understanding of its functionality, we endeavored to execute it, resulting in the following outcome. Subsequently, we authored a new script to operationalize its functionalities, necessitating a thorough comprehension of the codebase. Furthermore, we engaged with AnimateDiff [12], ensuring the installation of all requisite dependencies and obtaining a sample from AnimateDiff. This endeavor proved to be demanding, consuming a significant amount of time dedicated to reading the respective papers and familiarizing ourselves with the associated codebases.

However, challenges arose when attempting to amalgamate the Zero123 and AnimateDiff baselines due to their differing foundations—one based on Compvis and the other on Diffusers. Consequently, we corresponded with the instructor, proposing a transition from Zero123 to Zero123++ [37] to facilitate a smoother integration. Following this adjustment, we commenced the process of amalgamating Zero123++ with AnimateDiff, encountering the subsequent error:

```
RuntimeError: Error(s) in loading state_dict for UNet3DConditionModel:
        size mismatch for down_blocks.0.attentions.0.proj_in.weight: copying
        a param with shape torch.Size([320, 320, 1, 1]) from checkpoint, the
        shape in current model is torch.Size([320, 320]).
        size mismatch for down_blocks.0.attentions.0.transformer_blocks.0.att
        n2.to_k.weight: copying a param with shape torch.Size([320, 768]) fro
        m checkpoint, the shape in current model is torch.Size([320, 1024]).
        size mismatch for down_blocks.1.attentions.0.transformer_blocks.0.att
        n2.to_v.weight: copying a param with shape torch.Size([640, 768]) fro
        m checkpoint, the shape in current model is torch.Size([640, 1024]).
        size mismatch for down_blocks.1.attentions.0.proj_out.weight: copying
        a param with shape torch.Size([640, 640, 1, 1]) from checkpoint, the
        shape in current model is torch.Size([640, 640]).
```

```
RuntimeError: Error(s) in loading state_dict for UNet3DConditionModel:
[List of size mismatch errors]
```

This error stemmed from the disparity between the Zero123++ version 2.1 and AnimateDiff version 1.5 state dictionaries.

To address this issue, the instructor recommended transitioning to Stable Video Diffusion (SVD) [2]. After thoroughly studying the associated paper and codebase, we proceeded to modify both the SVD and Zero123++ codebases.

To facilitate the comparison of two UNets, we utilized DiffChecker, manually replacing the weights of corresponding blocks. These modifications were documented on our GitHub repository.

The following files within the Diffusers repository were modified to enable this integration:

```
diffusers/pipelines/zero123plus/pipeline_zero123plus.py
diffusers/pipelines/pipeline_utils.py
diffusers/pipelines/stable_diffusion/pipeline_stable_diffusion.py
diffusers/models/transformers/transformer_temporal.py
diffusers/models/unets/unet_3d_blocks.py
diffusers/models/unets/unet_spatio_temporal_condition.py
```

These alterations necessitated a profound understanding of both the Zero123++ and SVD codebases.

Additionally, we incorporated LORA into our system to support various fine-tuning modes for our network. These modes include:

1. **noxattn**: Excludes training on cross-attention and time embedding layers.
2. **innoxattn**: Excludes training on self-attention layers.
3. **selfattn**: Trains only self-attention layers.
4. **xattn**: Trains only cross-attention layers.
5. **full**: Trains all layers.
6. **xattn-strict**: Trains only the q and k values of cross-attention layers.

Throughout various experiments, as evidenced in the provided code for this phase, we explored different LORA parameters. Initially, we utilized the original diffusion loss function, but later opted for a new loss function as outlined in the methodology. We experimented with both ADAM and ADAMW optimizers, varying hyperparameters such as learning rate, beta1, beta2, and weight decay. Despite encountering noisy results when attempting to overfit on a sample, this phase involved extensive preparation, including two unsuccessful attempts to integrate networks, in-depth comprehension of Diffusers, incorporation of LORA, and testing across various settings.

During the rendering phase of our project, we familiarized ourselves with Blender. Our initial attempts at rendering images were met with several challenges. We encountered frequent crashes, largely due to the intensive CPU usage demanded by rendering tasks. Additionally, we grappled with issues related to lighting, which significantly impacted the quality of our renders.

To address the lighting concerns, we experimented with different light settings, particularly focusing on the "Area" type light. By adjusting parameters such as energy and scale, we were able to mitigate the lighting discrepancies across various samples, thereby enhancing the overall visual output.

Furthermore, a pivotal realization emerged regarding the normalization of objects within Objaverse [6]. We learned that object normalization and center adjustment were imperative steps for achieving consistent rendering results. Subsequently, by enabling the transparent background option in Blender, we successfully captured images with a clean, white background.

These rendered images served as the foundation for generating ground truth data for Zero123++ [37], a crucial component of our project's pipeline. Through meticulous refinement and iterative adjustments, we overcame rendering obstacles, ultimately preparing our dataset.

3.2 Phase 2

Transferring the mentioned loss functions in 6.5 from [23] to [53] was a demanding task which required a depth understanding of both methods since Zero123 uses a volumetric-based renderer named NeRF [27], while PAPR employs point clouds, so there are many differences in the way they render an image and estimate depth. The next challenge was to calculate depth estimation based on attention mechanism in PAPR to calculate last two loss functions, while it directly computed in NeRF method. These challenged derived me to modify, rewrite a implement method in different files such as `train.py` and `model.py`.

However, the main challenge was working with compute Canada as the resource. There were some tricks like loading `rust` before installing dependencies or loading `open-cv` instead of installing it with `pip`. The other thing was missing packages in the `requirements.txt` file, and it took a while to solve the conflicts between missing packages or the ones used by both PAPR and Zero123 but in different versions. After resolving these conflicts, we started to run the model with the newly introduced loss functions, but we got GPU memory error because both pretrained Zero123 and PAPR model should be loaded in `cuda` device. In order to solve the last problem, we went through configuration files and changed `num points` and `patch size`.

Introducing new loss function will inevitably lead to new hyper-parameters whose weights should be tuned alongside three important learning rate (lr) in PAPR which are point `mapping mlp`, `transformer`, `points` which each one has it's own `base_lr` etc. It worth mentioning that while PAPR is not the fastest rendering method due to the attention mechanism, it will get even slower because of the introduced losses (we need to render for extra views). It leads to the fact that running the modified version of PAPR takes around 20 hours for 5000 iteration slowing down the debugging process. In the next section, we will see the results of our experiments for different hyper-parameters.

3.3 Phase 3

In this phase, the first course of action was to set up the Compute Canada environment, which encountered some challenges at the time due their overhaul of the system as well as their compatibility issues with Objaverse. This is because the model training for PAPR is considerably heavy, and requires approximately 50,000 steps to generate a proper point cloud. We anticipated it taking even longer with our samples due to the complex geometry of our model.

Given the compatibility issues between Objaverse and Compute Canada at the time, dataset generation was adjusted to downloading blender files from the Objaverse source and rendering the images through a blender pipeline.

The PAPR model was also augmented so that the dataloader would be compatible with the new blender dataset and weight freezing for the UNet was incorporated for training point-clouds beyond the initial frame. Furthermore by using point clouds of the past frames, we would hopefully retain most of the features of the render model across subsequent frames. Additionally the weight freezing can be turned off through the parameter file should the error propagation from the initial render become magnified across too far across time.

The initial experiment of dataset 1 consisted of a much more difficult dataset to train on, consisting of less training data and incomplete shots of the model. This is reflected in the result PSNR, which gradually converged at 9.7 at twenty thousand iterations.

The dataset was subsequently adjusted to include more render images as well as more fully complete captures of the model, dubbed dataset 2. Point trimming was also adjusted so less points were relevant which resulted in a higher, but not yet fully optimal PSNR of about 12.7 at twenty thousand iterations.

4 Results

4.1 phase1

In Section 3, we note that this phase did not yield significant results. However, we present the output of the model along with frames from videos rendered using Objaverse. For additional results or to view video samples used for finetuning, please refer to our Github page.

In addition, we showcase some frames from rendered videos:

(a) Frame from a rendered video featuring an eagle (b) Frame from a rendered video featuring Woody

Figure 1: Sample frames from rendered videos. Each frame represents a random moment from the respective video.

These visuals provide a glimpse into the output of our model and the quality of rendered videos. For further exploration and access to more samples, please refer to our Github repository.

4.2 phase2

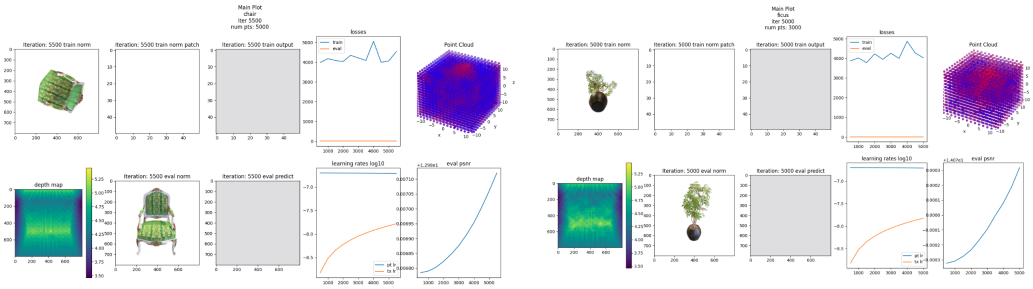
Our first attempt was to set coefficients the same for all losses which leads to very poor results shown for chair, and ficus for around 5500 iteration 2. By monitoring the magnitude of SJC gradient, we noticed that we should increase weight of others loss in comparison with SJC loss. Then by this intuition, we test the method again and set the weights of other losses more than SCJ to `1e5` and tested it on the same benchmarks as before, which the results are reported 3. It is turns out that the `lr` should obviously decreased when loss magnitude increases which leads to the third experiment where we decreased `lr_factor` from `1e-4` to `1e-7` and tested on chair benchmark for 48hrs 4. The result, this time, was more promising, but after some iteration, around 10000, the loss function was getting diverge. We guess that the problem raise from two things:

- The loss functions' weights should be tuned more carefully
- We reduced batch size and number of points in order to be able to work with `v100` GPUs which might had degenerative effect on convergence.

Experiment	Depth Smoothing Weight	Near-view Weight	MSE Weight	SJC Weight	`lr_factor`
1	1	1	1	1	1e-4
2	1e5	1e5	1e5	1	1e-4
3	1e5	1e5	1e5	1	1e-7

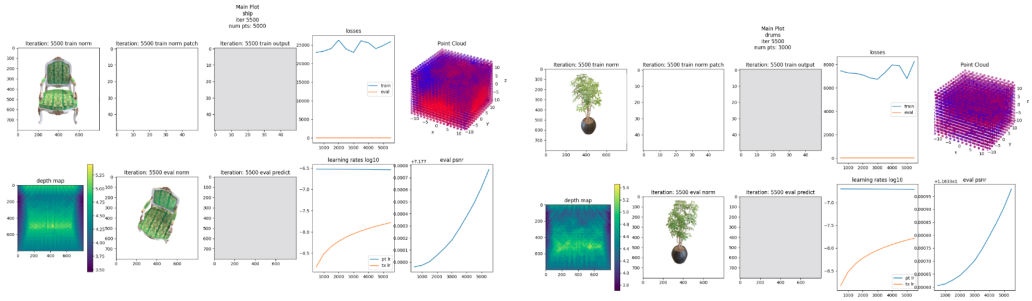
Table 1: Hyper-parameters tuning

In this part, only results of `chair` and `ficus` benchmarks (only for the last iteration) are reported, for check out the full results please find the supplementary materials zip file for this phase here. In the last few days before the deadline we start changing `goems.background.constant` value to prevent squeezing all points of background and foreground in one place.



(a) Chair (b) Drums

Figure 2: Experiment 1 with mentioned hyper-parameters in Table 1 on chair and ficus benchmarks



(a) Chair (b) Ficus

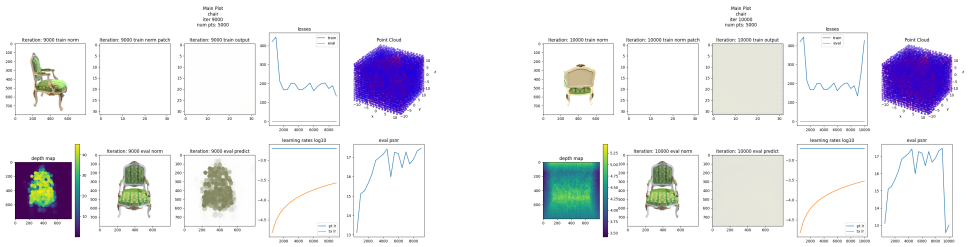
Figure 3: Experiment 2 with mentioned hyper-parameters in Table 1 on chair and ficus benchmarks

4.3 phase3

Resulting plot graphs from the augmented PAPR can be seen in figure 6 below, as well as the model point clouds. While noticeably improving in PSNR from the lower quality dataset, the final model point clouds are not yet accurate enough to complete the frame by frame pipeline of the augmented PAPR as the propagating errors would result in nonsense after just one time interval.

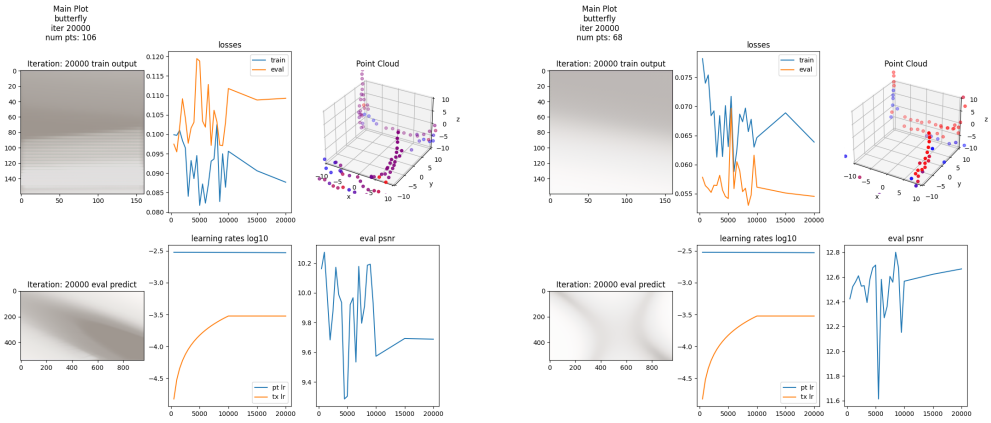
In order to proceed with the experiment further changes to the dataset need to be implemented in order to ensure higher quality point clouds. We surmise the reason for points congregating at the edges as seen in figure 1 is due to two reasons:

1. Lighting issues of the original model: inconsistent lighting across the top and bottom of the model could result in inaccurate point cloud convergence. Adjusting the lighting to be similar to the NeRF [27] dataset would likely impact eval PSNR positively.



(a) Before exploding gradient (b) After exploding gradient

Figure 4: Experiment 3 with mentioned hyper-parameters in Table 1 on chair benchmark



(a) Dataset 1 (b) Dataset 2

Figure 5: Phase3 Experimental Comparison

2. Render dimensions: While dataset 2 does have the model in frame for all its five hundred render images, the camera also captures a significant amount of white background. Given that PAPR works with training norm patches, this would result in many patches having tiny parts of the rendered model, or perhaps no parts at all. This is particularly prevalent along the edges which could explain why the point clouds congregate around the edges, perhaps trying to model the white background rather than the desired butterfly model. Adjustments should be made so the render camera dimensions are square while still fully capturing the butterfly model.

5 Breakdown of Contributions

5.1 Phase1

- Integration of Zero123++ and Stable Video Diffusion: **Amir**
- Learning how to work with Blender: **Amir**
- Using Blender to render Objeverse dataset: **Amir**
- Making rendered images compatible with Zero123++ and SVD (tiling images and adding white background and make them videos): **Amir**
- adding LORA supporting different modes: **Amir**
- testing different optimizers + their hyperparameters: **Amir**

5.2 Phase2

- Getting in depth knowledge on Zero123 codebase **Roozbeh**
- Getting in depth knowledge on PAPR codebase **Roozbeh**
- Setting up compute Canada, fixing dependencies conflicts **Roozbeh**
- Adjusting `train.py` and scripts in `model` module of PAPR implementation **Roozbeh**
- Tuning hyper-parameters, learning rates and loss function's coefficients to get an acceptable intermediate results on PAPR with new losses **Roozbeh**

5.3 Phase3

- Compute Canada Setup **Leo**
- Initial Blender pipeline and code for Windows Blender **Leo**
- Assistance with Phase 1 Blender Code (Objaverse) **Leo**
- Butterfly model for Animate-PAPR: inital and large dataset **Leo**

- Advice for adjusting PAPR model **Amir**
- Adjusting PAPR dataloader, model, and main functions **Leo**
- Animate-PAPR Butterfly Experiments **Leo**

References

[1] K. V. Alwala, A. Gupta, and S. Tulsiani. Pretrain, self-train, distill: A simple recipe for supersizing 3d reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022*, pages 3763–3772, New Orleans, LA, USA, June 18-24 2022. IEEE.

[2] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint*, 2023.

[3] A. Blattmann, R. Rombach, H. Ling, T. Dockhorn, S. W. Kim, S. Fidler, and K. Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. 2023.

[4] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[5] X. Dai, J. Hou, C.-Y. Ma, S. Tsai, J. Wang, R. Wang, P. Zhang, S. Vandenhende, X. Wang, A. Dubey, M. Yu, A. Kadian, F. Radenovic, D. Mahajan, K. Li, Y. Zhao, V. Petrovic, M. K. Singh, S. Motwani, Y. Wen, Y. Song, R. Sumbaly, V. Ramanathan, Z. He, P. Vajda, and D. Parikh. Emu: Enhancing image generation models using photogenic needles in a haystack. 2023.

[6] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv*, 2022.

[7] C. Deng, C. M. Jiang, C. R. Qi, X. Yan, Y. Zhou, L. J. Guibas, and D. Anguelov. Nerdi: Single-view nerf synthesis with language-guided diffusion as general image priors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023*, pages 20637–20647, Vancouver, BC, Canada, June 17-24 2023. IEEE.

[8] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794, 2021.

[9] Y. Furukawa, C. Hernandez, et al. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

[10] S. Ge, S. Nah, G. Liu, T. Poon, A. Tao, B. Catanzaro, D. Jacobs, J.-B. Huang, M.-Y. Liu, and Y. Balaji. Preserve your own correlation: A noise prior for video diffusion models. pages 22930–22941, 2023.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[12] Y. Guo, C. Yang, A. Rao, Z. Liang, Y. Wang, Y. Qiao, M. Agrawala, D. Lin, and B. Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. 2024.

[13] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, and T. Salimans. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[14] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[15] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 2020.

[16] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[17] H. Jun and A. Nichol. Shap-e: Generating conditional 3d implicit functions. *CoRR*, 2023.

[18] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, pages 4401–4410, 2019.

[19] Y. Lin, H. Han, C. Gong, Z. Xu, Y. Zhang, and X. Li. Consistent123: One image to highly consistent 3d asset using case-aware diffusion priors. *arXiv preprint arXiv:2309.17261*, 2023.

[20] M. Liu, R. Shi, L. Chen, Z. Zhang, C. Xu, X. Wei, H. Chen, C. Zeng, J. Gu, and H. Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. *CoRR*, 2023.

[21] M. Liu, C. Xu, H. Jin, L. Chen, M. Varma T, Z. Xu, and H. Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without pershape optimization. *CoRR*, 2023.

[22] M. Liu, C. Xu, H. Jin, L. Chen, Z. Xu, H. Su, and et al. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *arXiv preprint arXiv:2306.16928*, 2023.

[23] R. Liu, R. Wu, B. Van Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309, 2023. 2, 3, 5.

[24] Y. Liu, C. Lin, Z. Zeng, X. Long, L. Liu, T. Komura, and W. Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. *arXiv preprint arXiv:2309.03453*, 2023.

[25] X. Long, Y.-C. Guo, C. Lin, Y. Liu, Z. Dou, L. Liu, Y. Ma, S.-H. Zhang, M. Habermann, C. Theobalt, and W. Wang. Wonder3d: Single image to 3d using cross-domain diffusion. *CoRR*, 2023.

[26] L. Melas-Kyriazi, I. Laina, C. Rupprecht, and A. Vedaldi. Realfusion 360° reconstruction of any object from a single image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8446–8455, Vancouver, BC, Canada, June 2023. IEEE.

[27] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *CoRR*, abs/2003.08934, 2020.

[28] A. Nichol, H. Jun, P. Dhariwal, P. Mishkin, and M. Chen. Point-e: A system for generating 3d point clouds from complex prompts. *CoRR*, abs/2212.08751, February 2022.

[29] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Muller, J. Penna, and R. Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. 2023.

[30] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[31] G. Qian, J. Mai, A. Hamdi, J. Ren, A. Siarohin, B. Li, H.-Y. Lee, I. Skorokhodov, P. Wonka, S. Tulyakov, and B. Ghanem. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. *CoRR*, abs/2306.17843, February 2023.

[32] A. Raj, S. Kaza, B. Poole, M. Niemeyer, N. Ruiz, B. Mildenhall, S. Zada, K. Aberman, M. Rubinstein, J. T. Barron, Y. Li, and V. Jampani. Dreambooth3d: Subject-driven text-to-3d generation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2349–2359, Paris, France, October 2023. IEEE.

[33] A. Ramesh. How dall·e 2 works. 2022.

[34] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[35] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022.

[36] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *arXiv*, 2023.

[37] R. Shi, H. Chen, Z. Zhang, M. Liu, C. Xu, X. Wei, L. Chen, C. Zeng, and H. Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint*, 2023.

[38] U. Singer, A. Polyak, T. Hayes, X. Yin, J. An, S. Zhang, Q. Hu, H. Yang, O. Ashual, O. Gafni, D. Parikh, S. Gupta, and Y. Taigman. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

[39] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[40] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*, 2015.

[41] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[42] J. Tang, J. Ren, H. Zhou, Z. Liu, and G. Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023.

[43] J. Tang, T. Wang, B. Zhang, T. Zhang, R. Yi, L. Ma, and D. Chen. Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22762–22772, Paris, France, October 2023. IEEE.

[44] V. Voleti, A. Jolicoeur-Martineau, and C. Pal. Mcvd: Masked conditional video diffusion for prediction, generation, and interpolation. *arXiv preprint arXiv:2209.14792*, 2022.

[45] H. Wang, X. Du, J. Li, R. A. Yeh, and G. Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. *arXiv preprint arXiv:2212.00774*, 2022.

[46] H. Wang, X. Du, J. Li, R. A. Yeh, and G. Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. *arXiv preprint arXiv:2212.00774*, 2022.

[47] P. Wang and Y. Shi. Imagedream: Imageprompt multi-view diffusion for 3d generation. *CoRR*, abs/2312.02201, 2023. 2, 3, 7, 1.

[48] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023.

[49] C. Wen, Y. Zhang, C. Cao, Z. Li, X. Xue, and Y. Fu. Pixel2mesh++: 3d mesh generation and refinement from multi-view images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(2):2166–2180, 2023.

[50] C.-Y. Wu, J. Johnson, J. Malik, C. Feichtenhofer, and G. Gkioxari. Multiview compressive coding for 3d reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9065–9075. IEEE, 2023.

[51] J. Z. Wu, Y. Ge, X. Wang, W. Lei, Y. Gu, Y. Shi, W. Hsu, Y. Shan, X. Qie, and M. Z. Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. *arXiv*, 2023.

[52] P. Yoo, J. Guo, Y. Matsuo, and S. S. Gu. Dreamsparse: Escaping from plato's cave with 2d frozen diffusion model given sparse views. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023.

[53] Y. Zhang, S. Peng, A. Moazeni, and K. Li. Papr: Proximity attention point rendering. *Journal of Computer Vision and Image Understanding*, 2023.

6 Supplementary

6.1 Introduction

In today's rapidly evolving digital content landscape, the demand for high-fidelity 3D content spans industries such as film, video gaming, online streaming, and virtual reality (VR). While manual creation of 3D content yields impressive results, the significant time and labor investment required pose substantial challenges. To address this, a surge of innovative research [1, 7, 17, 20, 21, 25, 26, 28, 31, 32, 43, 47, 49, 50, 52] focuses on generating 3D models from single images, promising to lower barriers to entry for creators and revolutionize content creation accessibility.

Advancements in 3D content generation are propelled by novel view generative models, leveraging powerful 2D diffusion generative priors derived from extensive Internet datasets. Groundbreaking efforts like Zero-1-to-1 (Zero123) [23] pioneered open-world single-image-to-3D conversion, albeit facing challenges with geometric inconsistency, hindering seamless integration with multi-view images and 3D scenes. Recent enhancements such as One-2-3-45 [22], SyncDreamer [24], and Consistent123 [19] augment Zero-1-to-3 with additional layers to achieve more 3D-consistent outcomes. Moreover, optimization-based approaches like DreamFusion [30], ProlificDreamer [48], and DreamGaussian [42] distill 3D representations from inconsistent models to improve results. However, there's room for improvement by employing a base diffusion model capable of generating consistent multi-view images. Leveraging the state-of-the-art model Zero123++ [37], we fine-tune a multi-view consistent base diffusion model derived from Stable Diffusion [34] to enable the generation of 4D characters.

We integrate motion priors from the Stable Video Diffusion (SVD) model [2] into our multi-frame novel view synthesis (NVS) network, enhancing its ability to incorporate dynamic movements. Originally designed for video generation, SVD demonstrates exceptional consistency across multiple views and strong generalization due to its training on extensive image and video datasets. By leveraging SVD, our NVS network combines precise pose control, consistent multi-view synthesis, and broad generalizability, empowering it to produce diverse and lifelike multi-view renderings of 3D objects with explicit control over camera orientations.

Moreover, in many complex scenarios, like the setting of this paper in which the pipeline should generate 3d assets only give a single picture, reconstruction losses like mean square error is not sufficient to reach the goal and problem will probably remains severely underspecified In order to bring 3d reconstruction ability by using a evolved version of 2d stable diffusion we used score Jacobian chaining loss [46] to guide neural renderer for those views for which we do not access to ground truth. For this aim, we added several loss functions including SJC and some regularization techniques.

Additionally, by feeding in the inputs from our generated 4D images into an augmented point cloud rendering model following PAPR [53], we hope to be able to render video models to leverage for real world applications.

6.2 Related Work

6.2.1 Novel View Synthesis

Conventional 3D reconstruction approaches [9] typically rely on multi-view reasoning techniques. However, recent advancements in generative AI have led to a shift towards single-view 3D reconstruction methodologies. A prevalent approach involves utilizing generative models like GANs [11, 18] and diffusion models [8, 14, 35, 39] to generate multiple novel views from a single input. These generated views are subsequently fed into a multi-view 3D reconstruction module, such as NeRF. Currently, Zero123++ [37] stands as the state-of-the-art method for single- to multi-view synthesis. This model is built on Stable Diffusion (SD) and fine-tuned using a dataset of 800K 3D models from Objaverse [6].

6.3 Video Diffusion

Recent advancements in video generation primarily utilize diffusion models [15, 40, 41] to synthesize multiple coherent frames simultaneously, conditioned on either text or images. These models employ

an iterative refinement procedure, gradually removing noise from samples originating from a normal distribution. Diffusion models have demonstrated effectiveness in various applications, including high-resolution text-to-image [5, 29, 33] and video synthesis tasks [3, 10, 13, 38, 44].

6.4 Multi-Frame Novel View Synthesis (MNVS)

The MNVS network generates consistent novel view videos based on a single input image. The architecture of MNVS combines elements from Stable Video Diffusion and Zero123++. Zero123++ generates six images from six constant views, providing a prior for novel view synthesis. To incorporate motion priors for generating consistent frames, we replace the temporal transformer blocks with those from the Stable Video Diffusion model. However, this replacement requires fine-tuning, which we accomplish using the LORA algorithm [16].

To facilitate training, we require a dataset and experiment with different fine-tuning settings. We leverage the Objaverse dataset [6], which includes animation samples. By selecting a constant camera coordination and rendering six additional videos from relative camera coordinations specified in Zero123++, we create our dataset for network fine-tuning.

Fine-tuning the model involves exploring various modes, including full model fine-tuning, fine-tuning only self-attentions or cross-attentions, fine-tuning only the to_k and to_q parts of the attentions, and fine-tuning only the temporal transformer blocks [36, 51].

Since Zero123++ is trained on clean images, obtaining a clean image x_0 from each noisy latent x_t during the denoising diffusion process is essential. This process estimates the noise $\epsilon_\theta(x_t, t)$ added to x_0 to obtain x_t. Thus, x_0 can be obtained from $\epsilon_\theta(x_t, t)$ using this equation:

$$\widehat{x}_0 = \frac{x_t}{\sqrt{\bar{\alpha}_t}} - \frac{\sqrt{1-\bar{\alpha}_t}\epsilon_\theta\left(x_t, t\right)}{\sqrt{\bar{\alpha}_t}}$$

So, instead of using original diffusion loss, we can use the following loss function:

$$E_{t,x_0,\epsilon}\left[\|x_0 - \left(\frac{x_t}{\sqrt{\bar{\alpha}_t}} - \frac{\sqrt{1-\bar{\alpha}_t}\epsilon_\theta\left(x_t, t\right)}{\sqrt{\bar{\alpha}_t}}\right)\|_2\right]$$

6.5 Score Jacobian Chaining and Regularization Losses

In phase 2 we were supposed to bring loss functions from Zero123 [23] method into PAPR [53]. By doing so both the 3D assets of PAPR (point cloud) and attention-base neural renderer would be updated by the gradient of these losses. All in all, loss functions are as follow:

- **MSE loss**: Mean Squared Error tries to make the model able to reconstruct an image from a given ground truth image.
$$MSE_\theta = ||y - f_\theta(view)||_2$$
- **SJC**: Score Jacobian Chaining loss, firstly introduced in [46], tries to use the score function extracted by output of a 2D stable-diffusion based model to provide a learning signal for those views that we do not have the ground truth image for them.
$$\underbrace{\nabla_\theta \log \tilde{p}_\sigma(\theta)}_{3Dscore} = \mathbb{E}_\pi[\underbrace{\nabla_{x_\pi} \log p_\sigma\left(x_\pi\right)}_{2Dscore;pretrained} \cdot \underbrace{J_\pi}_{rendererJacobian}]$$
- **Near-view Consistency**: This loss ensures that a generated 3D model remains consistent when viewed from closely related angles. It enhances the realism and detail of models by penalizing variations in appearance between similar viewpoints, typically used alongside other loss functions in neural networks.
$$L_{nvc} = \sum_{i=1}^{N}\left(\|I_{pred}^{(i)} - I_{gt}^{(i)}\|_2^2 + \lambda\|D_{pred}^{(i)} - D_{gt}^{(i)}\|_2^2\right) \quad (1)$$
where $I_{pred}^{(i)}$ and $I_{gt}^{(i)}$ are the predicted and ground truth images for the i-th viewpoint, respectively, and $D_{pred}^{(i)}$ and $D_{gt}^{(i)}$ are the corresponding depth maps. The parameter λ is a weighting factor that balances the importance of image consistency versus depth accuracy.

- **Depth Smoothing**: This loss ensure that the depth values of adjacent pixels in the reconstructed model change gradually, promoting smoothness and reducing noise. This function helps in generating more visually coherent surfaces by minimizing abrupt depth transitions between neighboring points.

$$L_{depth-smooth}(depth) = \frac{1}{2}\left(\overline{\left|\frac{\partial depth}{\partial x}\right|} + \overline{\left|\frac{\partial depth}{\partial y}\right|}\right) \tag{2}$$

where $\frac{\partial depth}{\partial x}$ and $\frac{\partial depth}{\partial y}$ represent the gradients of the depth values in the x and y directions, respectively, and the overline denotes the mean value of the absolute gradients.