TPA3D: Triplane Attention for Fast Text-to-3D Generation

Bin-Shih Wu[1*], Hong-En Chen[2*], Sheng-Yu Huang[1], and Yu-Chiang Frank Wang[1,3]

[1] Graduate Institute of Communication Engineering, National Taiwan University
[2] National Taiwan University
[3] NVIDIA
{r12942090, b08901058, f08942095}@ntu.edu.tw
frankwang@nvidia.com

A Implementation Details

A.1 Model Configuration

In our TPA3D, we set $N = 6$ layers in the configuration of our sentence-level triplane generators G. As for embedding text inputs, we employ the pre-trained CLIP ViT-B/32 [8] as our encoder, keeping its weight frozen during training. In all experiments, we employ 8 NVIDIA V100 GPUs with a batch size of 32 to train both the generator and discriminator, completing the total training duration in approximately three days. Specifically, we use the Adam [6] optimizer with $\beta = 0.9$, setting the learning rate to 0.001 for the generator and 0.002 for the discriminator. Finally, the whole model is implemented with the PyTorch [7] framework, and we render and visualize the generated 3D objects using Blender [2].

A.2 More Details about Datasets

In Sec. 5.1, we detail the preparation for datasets of ShapeNet [1] and OmniObject3D [10]. Specifically, we render 179,928, 162,672, 37,700, 19,200, and 9900 images with a resolution of 1024x1024 for *Car, Chair, Motorbike, Vehicle*, *Accessory*, respectively. And following the setting provided by GET3D [4], we split the training, validation, and testing set with the ratio of 7:1:2. Note that we also run the official implementation of TAPS3D [9] and GET3D with the same data split for comparison.

A.3 More Details about GET3D

In Sec. 3 of our main paper, we provide a brief review of GET3D [4]. Here, we detail the generator architecture of GET3D in Fig. A1. With modulated convolution layers [5] (denoted as Mod Conv) conditioned on latent vectors $\mathbf{w}_{\text{geo}}$

[*] These authors contributed equally to this work.

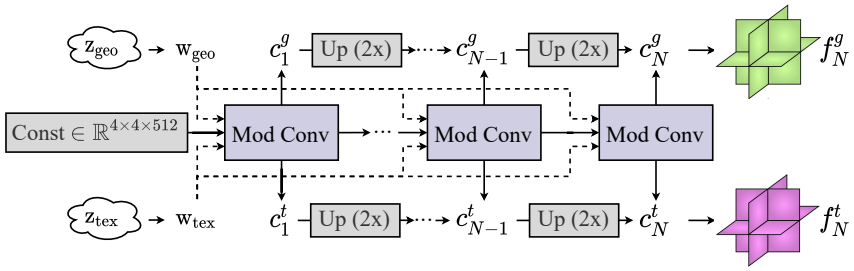


Fig. A1: Detailed architecture of GET3D [4]. Referenced from the original paper of GET3D, the design of GET3D processed the produced triplane features, where *Mod Conv* denotes the generator layers in GET3D, and c_i^g and c_i^t represents the i-th geometry and texture triplane, respectively. Note that the upsampling algorithm is bilinear upsampling.

Fig. A2: Rendered images of 3D objects and their corresponding pseudo captions predicted by InstructBLIP [3].

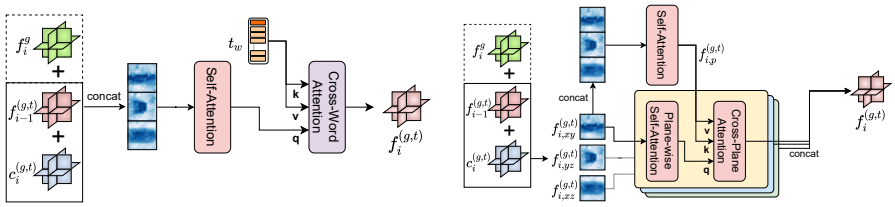
and $\mathbf{w}_{\text{tex}}$, the geometry triplane c_i^g and texture triplane c_i^t are generated from each layer i. By bilinear upsampling, the triplanes can be added together with the same resolution, and the final triplanes $f_N^g = \sum_{i=1}^{N}(c_i^g)$ and $f_N^t = \sum_{i=1}^{N}(c_i^t)$ are used for generating the output 3D textured mesh.

A.4 More Details of Pseudo Captioning

As for the production of pseudo captions, we generate them using InstructBLIP [3] with NVIDIA V100 GPUs beforehand to reduce the required computational resources during training. To elaborate, we load a pre-trained InstructBLIP (Vicuna-7B) and specify the input prompt as *"In the image, the background is black. Describe the design and appearance of the {category} in detail."* Subsequently, we refine the generated pseudo captions for each rendered image by filtering out redundant information. This involves eliminating background-related phrases like *"in the black background"* and *"with a black background"* to maintain focus on the geometry and texture of the 3D object. Additionally, distracting phrases such as *"This is a 3D model of"* or *"This is a 3D rendering of"* are removed to ensure a more precise and relevant description. In Figure A2, we sample several pseudo caption pairs with rendered RGB images of 3D objects from ShapeNet [1] and OmniObject3D [10].

Table A1: Ablation studies to the components in TPA blocks. Note that FID and CLIP R-precision@5 are reported.

Method	FID ↓			CLIP R-precision ↑		
	Car	Chair	Motorbike	Car	Chair	Motorbike
Ours	**18.5**	**38.1**	**77.7**	**80.94**	**38.58**	**24.76**
w/o cross-word attn	20.4	42.4	78.7	70.94	30.96	21.36
w/o cross-plane attn	30.0	47.7	80.9	79.73	35.49	23.04
w/o TPA	32.6	51.1	82.5	68.72	26.90	20.82



(a) w/o cross-plane attention **(b)** w/o cross-word attention

Fig. A3: The architectures for the ablation study in Sec. B.1. (a) Without cross-plane attention, the word features might be attended to the region with incomplete spatial information, which leads to a lower visual quality. (b) Without cross-word attention, triplanes lack detailed information in the description and only contain global information from sentence features.

B Ablation Study

Since the proposed TPA module serves as the major contribution to our TPA3D, we conduct an ablation study on TPA blocks to quantitatively assess the impact of this module. In addition, we also conduct an ablation study on the training losses we introduced to analyze how training objectives influence the training process. All results are presented in Table A1 and Table A2. Note that the baseline model (i.e., w/o TPA) removes both TPA_{geo} and TPA_{tex} and generates the final triplanes only using sentence-level triplanes conditioned on sentence features.

B.1 Components in TPA blocks.

To verify the function of each component in TPA blocks as our claim, we singly remove cross-plane attention or cross-word attention in TPA blocks (as shown in Figure A3). From Table A1, we can see that the removal of cross-plane attention leads to a significant decrease in FID. This result indicates that without the enhancement of plane features before cross-word attention, the visual quality severely decreases due to incomplete spatial information. On the other hand, the removal of cross-word attention brings a large drop in CLIP-R precision. This result verifies that cross-word attention promotes the correspondence between the detailed input text and the generated 3D object.

	FID ↓	CLIP-R ↑
Ours	**18.5**	**80.94**
w/o TPA_{tex}	18.6	76.85
w/o TPA_{geo}	20.4	79.46
w/o TPA	32.6	68.72

(a) TPA_{geo} and TPA_{tex} (Sec. B.2)

	FID ↓	CLIP-R ↑
Ours (full)	**18.5**	**80.94**
w/ TPA x 3 (half)	20.1	78.07
w/ TPA x 0	32.6	68.72

(b) Block numbers of TPA (Sec. B.3)

	FID ↓	CLIP-R ↑
Ours	**18.5**	**80.94**
w/o $\mathcal{L}_{\text{clip}}$	56.9	56.62
w/o $\mathcal{L}_{\text{mis}}$	68.6	68.19

(c) Training objectives (Sec. B.4)

Table A2: Other ablation studies of our TPA3D on *Car* in ShapeNet. Note that FID and CLIP R-precision@5 are reported.

B.2 TPA_{geo} and TPA_{tex}.

In order to generate textured meshes while achieving disentanglement of geometry and texture, TPA3D utilizes distinct branches to generate the final geometry and texture triplanes. To assess the efficacy of TPA blocks in these two branches, we symmetrically remove TPA_{geo} and TPA_{tex} and compare the results with the full architecture and baseline model without TPA blocks. The outcomes, presented in Table A2a, reveal a slight decline in FID and CLIP R-precision when either geometry TPA blocks or texture TPA blocks are omitted. Compared to the baseline model without TPA, the additional TPA blocks on either branch can still enhance visual quality and text-shape consistency.

B.3 Block Numbers of TPA.

In Table A2b, we compare models with varying numbers of TPA blocks. The original TPA3D applies six TPA blocks across all output resolutions of the sentence-level triplane generator G. To explore the impact of reducing the number of TPA_{geo} and TPA_{tex}, we create a variant with TPA blocks only applied in the first three resolutions of both the geometry and texture branches. From the findings shown in Table A2b, it is evident that a reduction in the number of TPA blocks corresponds to a decrease in both FID and CLIP R-precision metrics.

B.4 Training Objectives.

As noted in [9], training conditional GET3D [4] from scratch often results in collapsing shapes during training. To enable end-to-end one-stage training, we introduce additional $\mathcal{L}_{\text{mis}}$ in conjunction with $\mathcal{L}_{\text{clip}}$ to improve training stability and text-shape consistency. To investigate the impact of different objectives, we conduct an ablation study on these two training losses, $\mathcal{L}_{\text{mis}}$ and $\mathcal{L}_{\text{clip}}$. The results, as presented in Table A2c, reveal that the absence of either $\mathcal{L}_{\text{mis}}$ or $\mathcal{L}_{\text{clip}}$

Fig. A4: More text-guided 3D generation results of TPA3D. We formulate the input prompts as "a {color} {object}" with various colors and sub-classes for generation. Specifically, each column stands for a different color, while each row stands for a unique sub-class: (a) *"convertible"* (b) *"SUV"* (c) *"stool"* (d) *"plastic chair"* (e) *"bicycle"* (f) *"sport bike"*

leads to degraded performance in both FID and CLIP R-precision. This further validates our inclusion of additional negative pairs with mismatched text conditions in $\mathcal{L}_{\text{mis}}$ significantly benefits the discriminator and enhances the stability of training, resulting in improvement in FID and CLIP R-precision.

C More Qualitative Results

In this section, we present more qualitative results in terms of styles, subclasses, and more detailed text prompts to showcase TPA3D's ability to produce 3D shapes that closely align with the given text prompt.

C.1 Disentanglement of Geometry and Texture

In Figure A4, our TPA3D demonstrates its ability to disentangle geometry and texture information, resulting in shapes and textures that match the provided text prompt. By fixing the random seed and subclass in each row, the geometry of the generated shapes remains nearly unchanged, showcasing precise modifications to the texture. Similarly, each column illustrates that consistent color information can be applied to different geometries, even when different subclasses are specified in the text prompt. Additionally, our TPA3D can interpret rarer colors such as *"scarlet"*, *"light green"*, *"aqua"*, and *"indigo"*, generating textures that visually align with the given commands. In summary, our proposed TPA

Fig. A5: The interpolation results of our TPA3D. In each row, we use the same random noises $\mathbf{z}_{\text{geo}}$ and $\mathbf{z}_{\text{tex}}$, and perform interpolation on latent vectors $\mathbf{w}_{\text{geo}}$ and $\mathbf{w}_{\text{tex}}$ for different text inputs.

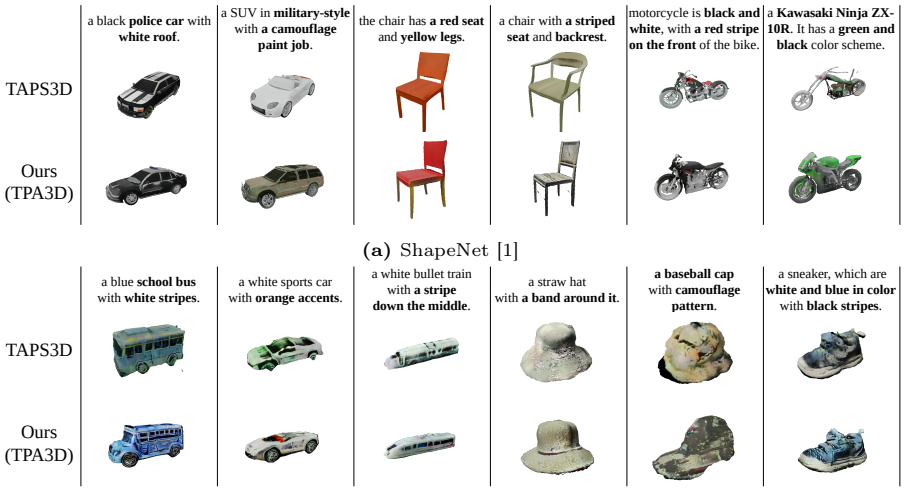
blocks excel in conducting word-level refinement for both texture and geometry, enabling precise generation to match the details specified in the given text prompt.

C.2 Interpolation of Geometry and Texture

In addition to disentanglement, our TPA3D also keeps the continuity of the latent space for generating 3D shapes and textures. In Figure A5, we use the same random noises $\mathbf{z}_{\text{geo}}$ and $\mathbf{z}_{\text{tex}}$ for each row, and then generate the shapes and textures with the interpolation on latent vectors $\mathbf{w}_{\text{geo}}$ and $\mathbf{w}_{\text{tex}}$ for different text prompts. We can see the continuous shapes and colors match the interpolation of text prompts, which demonstrates the continuity of the latent space.

C.3 Generation for Detailed Description

We provide more qualitative comparisons with TAPS3D [9] to show the ability to generate objects with detailed text descriptions in Figure A6. The results presented in Figure A6 confirm that TPA3D effectively leverages word-level information to refine the textured shape, ensuring alignment with the specific details mentioned in the textual description. Considering the generation of a military-style SUV as an example, TPA3D not only accurately captures the geometry of the *"SUV"* but also successfully retrieves detailed requirements such as *"military-style"* and *"camouflage paint"*. With word-level refinement provided by TPA blocks, our TPA3D also excels in specifying modifications to particular elements such as *"white roof"*, *"striped seat and backrest"* or *"with a band around it"*. In contrast, TAPS3D can only generate textured shapes at a more generalized class and color level. For a comprehensive evaluation of the 3D shape from various perspectives, a *.mp4* file (*detailed_text_prompt.mp4*) is provided for comparative analysis with TAPS3D.



(a) ShapeNet [1]

(b) OmniObject3D [10]

Fig. A6: More qualitative comparisons with TAPS3D [9] on (a) ShapeNet and (b) OmniObject3D. Given detailed textual descriptions, our TPA3D generates accurate shapes aligned to the texts, while TAPS3D only realizes general classes and simple colors.

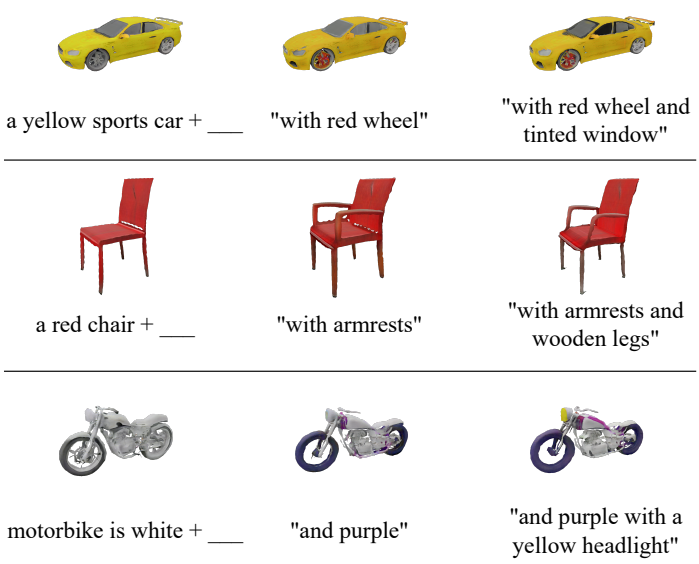


Fig. A7: More manipulation examples of adding different detailed text descriptions. Each row shows an example of manipulating the left-most object with detailed descriptions. With the same random seed for sampling $\mathbf{z}_{\text{geo}}$ and $\mathbf{z}_{\text{tex}}$, two distinct results are shown along with the original one in each row.

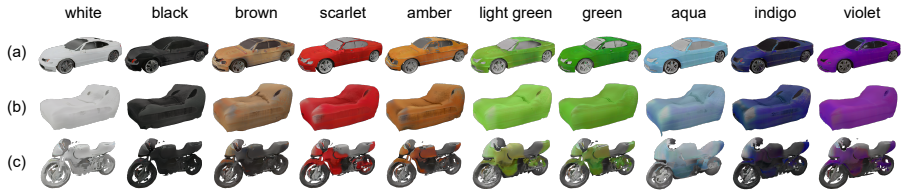


Fig. A8: Qualitative results of multi-class 3D generation. We formulate the input prompts as "a {color} {object}" with various colors and sub-classes for multi-class generation. Specifically, each column stands for a different color, while each row stands for a unique sub-class: (a) *"sports car"* (b) *"sofa"* (c) *"sport bike"*

Table A3: Comparisons between single-class and multi-class TPA3D. For multi-class generation, we train our TPA3D using data of *Car*, *Chair* and *Motorbike*. Following the single-class evaluation protocol, we assess this multi-class TPA3D for each class by randomly sampling pseudo captions from the respective test set to generate rendered images, with both FID and CLIP R-precision@5 as the metrics. With an equivalent model capacity, the outcomes exhibit a slight degradation but remain satisfactory.

Class	Method	FID(↓)	CLIP R-precision(↑)
Car	Ours (single-class)	18.50	80.94
	Ours (multi-class)	20.77	68.79
Chair	Ours (single-class)	38.11	38.58
	Ours (multi-class)	47.54	20.80
Motorbike	Ours (single-class)	77.69	24.76
	Ours (multi-class)	74.79	19.00

C.4 Controllable Manipulation

In Figure A7, we present additional manipulation experiments, offering users the ability to incrementally adjust the generated shape according to their specific stylistic and geometric preferences by fixing random seeds. For instance, after generating a *"white motorbike"*, users can add *"purple"* decorations to the same bike or even append *"a yellow headlight"* to the front of the motorbike. Likewise, users can incorporate geometry details such as *"armrests"* to the original simplistic *"red chair"*, maintaining nearly unchanged shapes and textures. This capability to produce 3D shapes rapidly and accommodate incremental text prompt requirements without significantly altering the initial shape enhances the utility of TPA3D as an interactive 3D asset creation tool, providing users with quick customization and control over the final output.

C.5 Multi-class Generation

To explore the potential of our TPA3D in the multi-class generation, we trained the model on a combined dataset comprising *Car*, *Chair*, and *Motorbike*. We con-

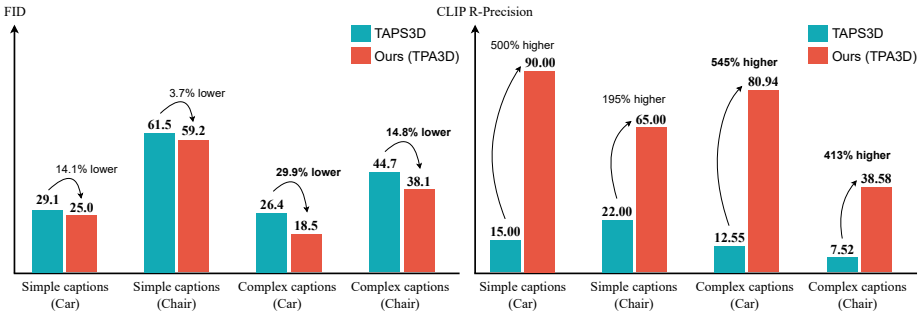


Fig. A9: Performance comparison with TAPS3D [9] using simple and complex captions in terms of FID and CLIP R-precision. Different from complex captions containing fine-grained descriptions, simple captions are only composed of *color* and *class*. The performance gap is larger when using complex captions, which validates the effectiveness of our TPA blocks in dealing with detailed text prompts.

ducted both qualitative and quantitative evaluations to assess its performance. By comparing with the single-class version of TPA3D, the quantitative result in Table A3 indicates that our TPA3D is capable of multi-class generation. Given the same model capacity as the single-class version, we anticipate a minor decrease in FID and CLIP R-precision scores, but the results remain satisfactory. Also, as depicted in Figure A8, the multi-class adaptation of TPA3D yields 3D textured shapes closely matching the detailed text prompts in texture and geometry.

D Human Evaluation for Verifying Fidelity

We conduct a human evaluation on 40 subjects to support our TPA3D. We use the same 100 captions for both our method and TAPS3D to generate 40, 40, and 20 objects for *Car*, *Chair*, and *Motorbike* in ShapeNet, respectively. Out of 4000 total responses, 2713 (67.8%) favor our method. Besides, our method is preferred in 73 out of 100 examples. Lastly, all 40 subjects indicate that our method generates higher fidelity shapes. These results demonstrate our method's superior performance in fidelity. It's worth noting that, quantitative evaluation has been provided in our main paper, which confirms that our method achieved better image fidelity than TAPS3D in terms of FID by a significant margin of 7.2 on average on ShapeNet, as shown in Table 1 in our main paper.

E TPA Effectiveness for Simple or Complex Captions

As our major contribution, TPA blocks are designed to enhance triplane features for capturing fine-grained information from the input text, especially when the text descriptions are complex and with details. To verify this claim, we test our

model and TAPS3D [9] with only simple captions (only *color* + *class*) from *Car* and *Chair* in ShapeNet [1]. As Shown in Figure A9, we can observe that the performance gap between our model and TAPS3D increases when we shift from simple captions to complex captions. For FID, the performance gap increases from 14.1% to 29.9% for *Car* and from 3.7% to 14.8% for *Chair*. As for CLIP R-precision, the performance gap increases from 500% to 545% for *Car* and from 195% to 413% for *Chair*. This result proves that our TPA blocks have better capability to deal with text prompts with detailed information.

F Limitations

Although our proposed TPA3D performs fast text-guided 3D generation with satisfactory alignment with text inputs, the retrieved text condition of TPA3D relies on pre-trained image captioning models and text encoders. While this suggests that our model does not limit the use of particular pre-trained vision-language models or text encoders, the quality of the resulting text embedding outputs would inevitably affect the learning and generation performances.

References

1. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
2. Community, B.O.: Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam (2018), `http://www.blender.org`
3. Dai, W., Li, J., Li, D., Tiong, A.M.H., Zhao, J., Wang, W., Li, B., Fung, P., Hoi, S.: Instructblip: Towards general-purpose vision-language models with instruction tuning (2023)
4. Gao, J., Shen, T., Wang, Z., Chen, W., Yin, K., Li, D., Litany, O., Gojcic, Z., Fidler, S.: Get3d: A generative model of high quality 3d textured shapes learned from images. Advances In Neural Information Processing Systems **35**, 31841–31854 (2022)
5. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8110–8119 (2020)
6. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
7. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
8. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)

9. Wei, J., Wang, H., Feng, J., Lin, G., Yap, K.H.: Taps3d: Text-guided 3d textured shape generation from pseudo supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16805–16815 (2023)
10. Wu, T., Zhang, J., Fu, X., Wang, Y., Ren, J., Pan, L., Wu, W., Yang, L., Wang, J., Qian, C., et al.: Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 803–814 (2023)