SEOULTECH
SEOUL NATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY

MuLILAB

Vista4D: Video Reshooting with 4D Point Clouds

Kuan Heng Lin & Zhizheng Liu

2026.08.03.

BrainLAB Journal Club
Department of Applied Artificial Intelligence
Jeong SangYeop

Overview

Author

01 Author

MuLILAB

Kuan Heng Lin

팔로우

다른 이름 ▸

Columbia University
columbia.edu의 이메일 확인됨 - 홈페이지

Computer vision generative models graphics deep learning

제목	인용	연도
FreeControl: Training-free spatial control of any text-to-image diffusion model with any condition S Mo, F Mu, KH Lin, Y Liu, B Guan, Y Li, B Zhou Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern …	151 *	2024
Ctrl-X: Controlling Structure and Appearance for Text-To-Image Generation Without Guidance KH Lin, S Mo, B Klingher, F Mu, B Zhou Advances in Neural Information Processing Systems	59	2024
Vista4D: Video Reshooting with 4D Point Clouds KH Lin, Z Liu, P Salamanca, Y Kant, R Burgert, Y Xu, K Namekata, Y Zhao, ... Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern …	3	2026
Go-with-the-Track: Video Compositing and Motion Control with Point Tracking K Namekata, Y Kant, Z Liu, RD Burgert, Y Xu, KH Lin, E Steven, J Philip, ... arXiv preprint arXiv:2606.20891		2026

학술자료 1–4 더보기

Vista4D

Video Reshooting with 4D Point Clouds

Kuan Heng Lin[1,3†] Zhizheng Liu[1,4†] Pablo Salamanca[1,2] Yash Kant[1,2] Ryan Burgert[1,2,5†] Yuancheng Xu[1,2]
Koichi Namekata[1,2,6†] Yiwei Zhao[2] Bolei Zhou[4] Micah Goldblum[3] Paul Debevec[1,2] Ning Yu[1,2]

[1]Eyeline Labs [2]Netflix [3]Columbia University [4]UCLA [5]Stony Brook University [6]University of Oxford

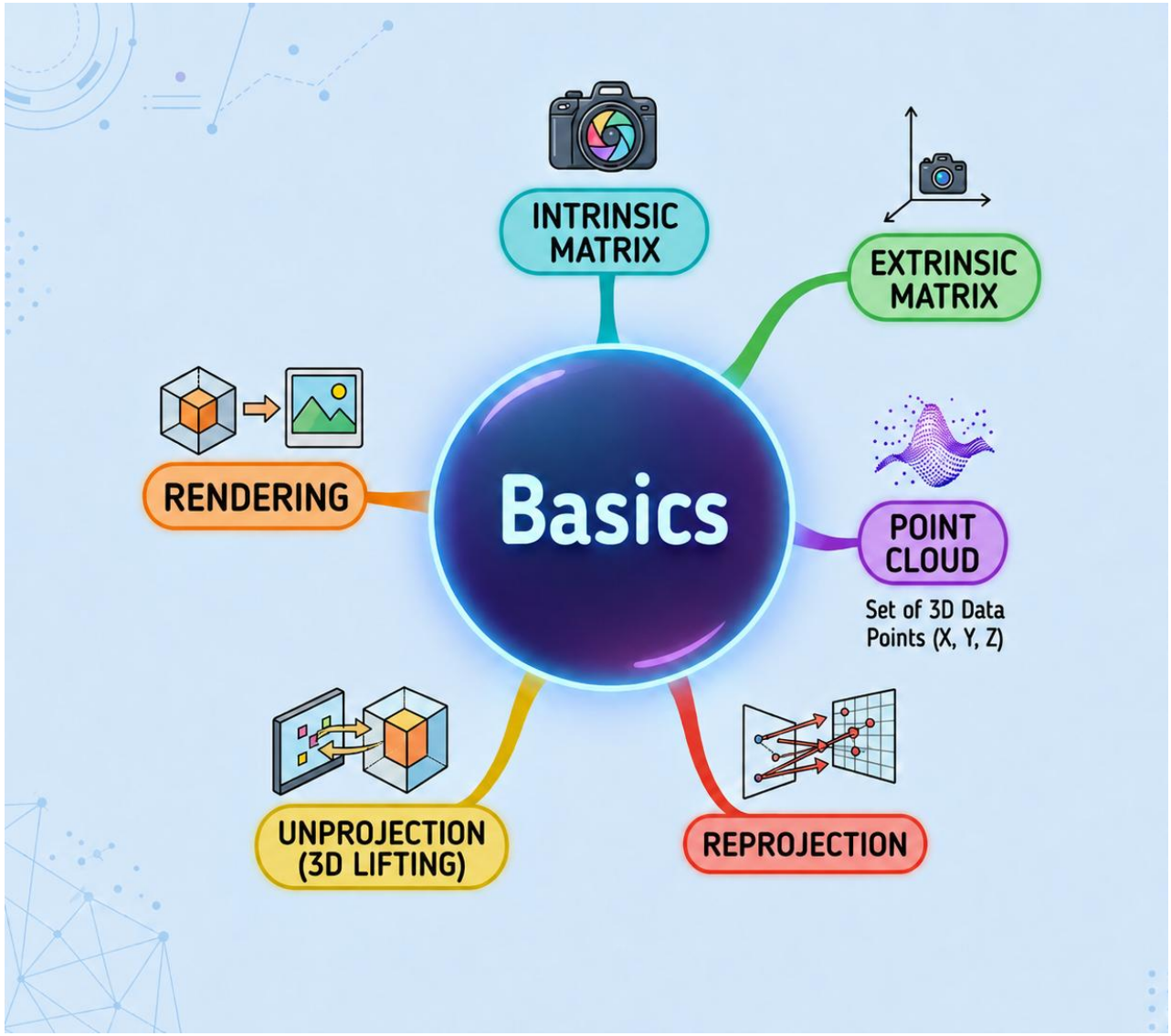
[†]Work done during an internship at Eyeline Labs

CVPR 2026 Highlight

Paper arXiv Code Models Eval

Source video as input

Challenges


INTRINSIC MATRIX
EXTRINSIC MATRIX
RENDERING
Basics
POINT CLOUD
Set of 3D Data Points (X, Y, Z)
UNPROJECTION (3D LIFTING)
REPROJECTION

02 Challenges

MuLILAB

“Video reshooting with explicit priors”

“Video reshooting with implicit priors”

“4D reconstruction”

02 Challenges

MuLILAB

“Video reshooting with explicit priors”

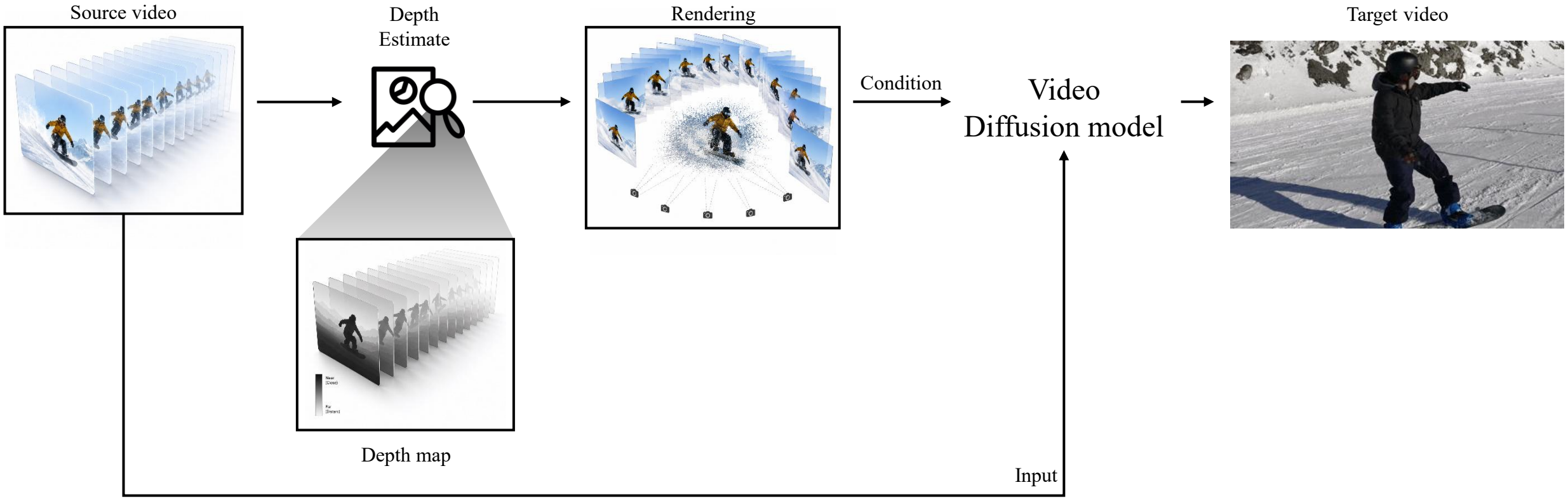
“Video reshooting with implicit priors”

“4D reconstruction”

MuLILAB

“Video reshooting with explicit priors”

Give explicit guidelines to Video Generative Model



MuLILAB

But…

1. Heavily rely on perfect dataset (fail to maintain generalization ability)
   - Many models trained with precise depth maps
   - Therefore, fail to predict videos in imperfect real-world depth estimation

2. Camera Control & Maintaining Contents
   - Point clouds are corrected in per-frame
   - Fail to control camera pose and maintain contents in the scene

02 Challenges

MuLILAB

“Video reshooting with explicit priors”

“Video reshooting with implicit priors”

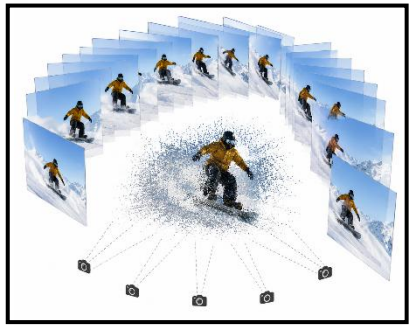
“4D reconstruction”

MuLILAB

“Video reshooting with implicit priors”

Inject video reference Info to pre-trained Video Generative Model

A great number of
Multi-view datasets



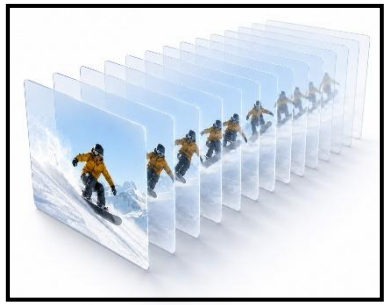
\+

“camera pose embeddings
or
video camera trajectory”

Inject

Video
Diffusion model
(finetuning)

Source video



\+

“camera pose embeddings
or
video camera trajectory”

Inject

Video
Diffusion model

Target video

MuLILAB

But…

1. Depth Scale Ambiguity
   - Hard to control camera pose for intended camera position
   - Because the information injected into the video diffusion model is ambiguous

2. Preview
   - There is no explicit point clouds
   - Therefore, cannot see the preview of the scene

02 Challenges

MuLILAB

“Video reshooting with explicit priors”

“Video reshooting with implicit priors”

“4D reconstruction”

MuLILAB

“4D Reconstruction”

Lifting the 2D video to 3D world-space including Temporal-Spatial Space

1. **Structure from Motion**
2. **Depth Priors + SLAM**
3. **End-to-End 4D reconstruction model**
4. **Predicting 4D Gaussians**

MuLILAB

But…

1. Dynamic Scene
   - Fail to maintaining geometry structure of dynamic objects

2. Limitation of Prediction
   - Predicting the 4D gaussians are very hard in novel view
   - Holes, Occlusions are observed in novel view

Vista4D

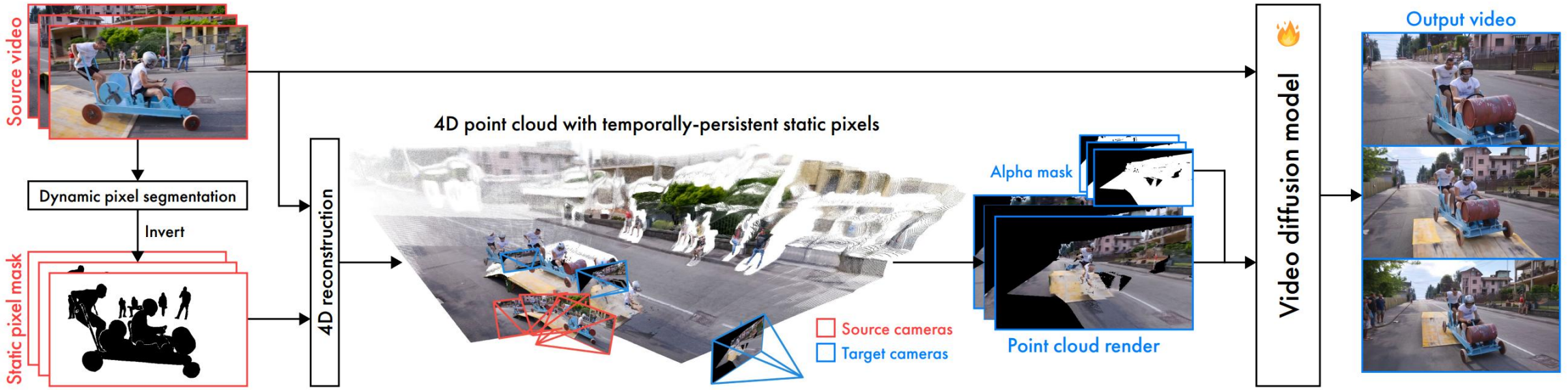
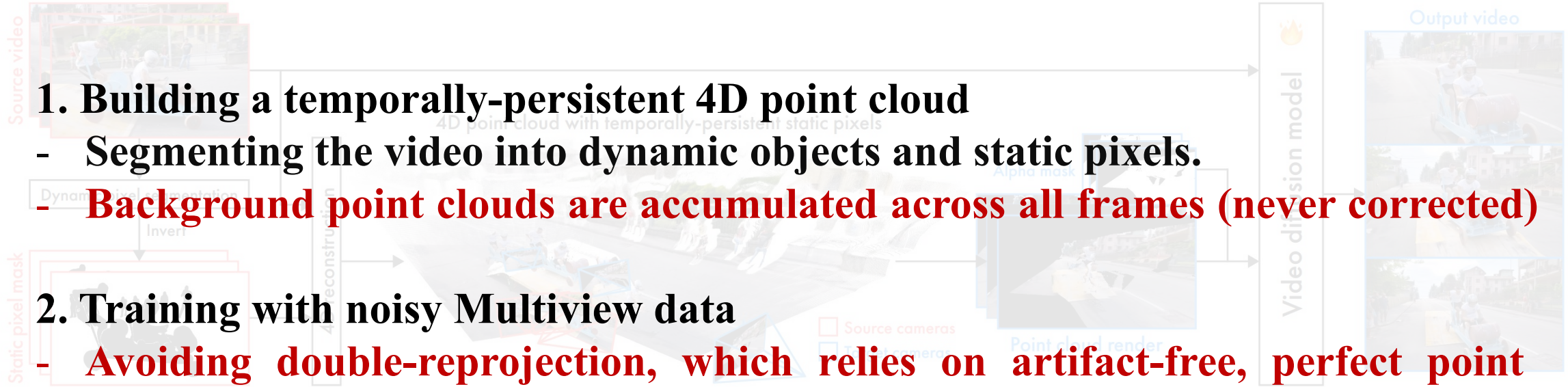


Figure 2. **Overview of Vista4D.** Given an input source video, we build a 4D point cloud where static pixels are temporally persistent via segmentation and 4D reconstruction. We then render the point cloud in the target cameras which users define. Lastly, the source video and point cloud render & alpha mask are jointly processed by the finetuned video diffusion model to generate a video of the same dynamic scene in the target cameras. We provide model architecture details in Supplementary B.



1. Building a temporally-persistent 4D point cloud
   - Segmenting the video into dynamic objects and static pixels.
   - Background point clouds are accumulated across all frames (never corrected)

2. Training with noisy Multiview data
   - Avoiding double-reprojection, which relies on artifact-free, perfect point clouds
   - Therefore, the model is robust to noisy 4D reconstructions, especially at non-frontal views.

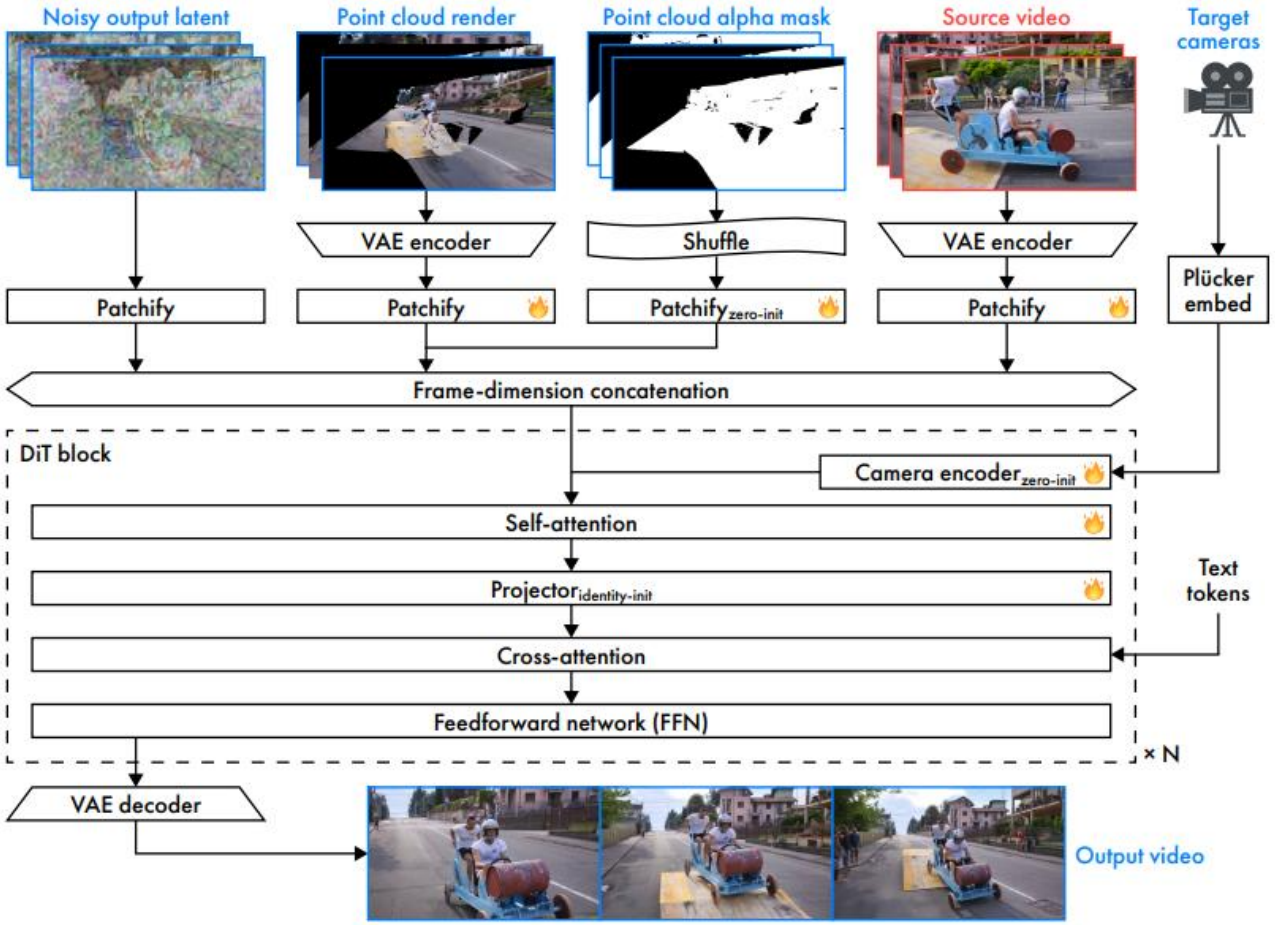
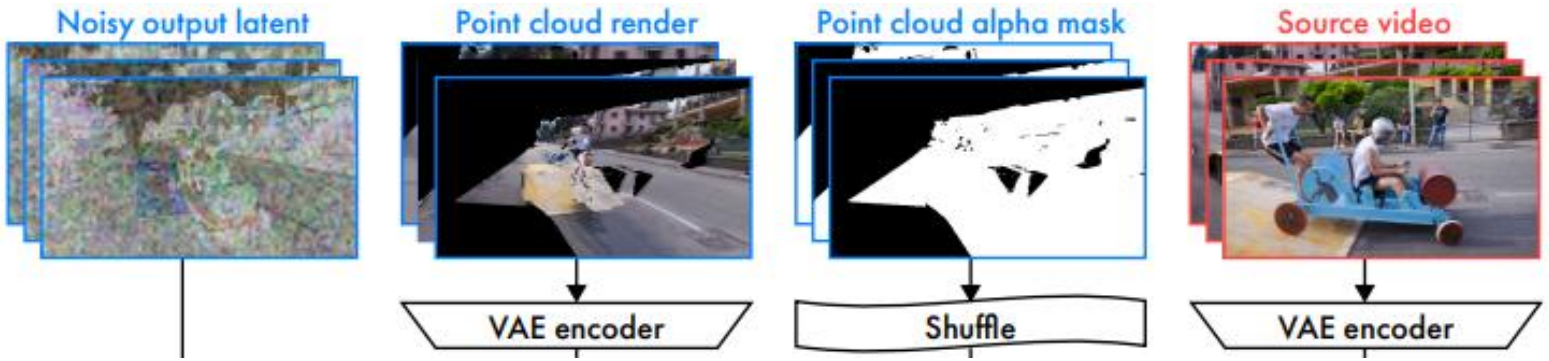
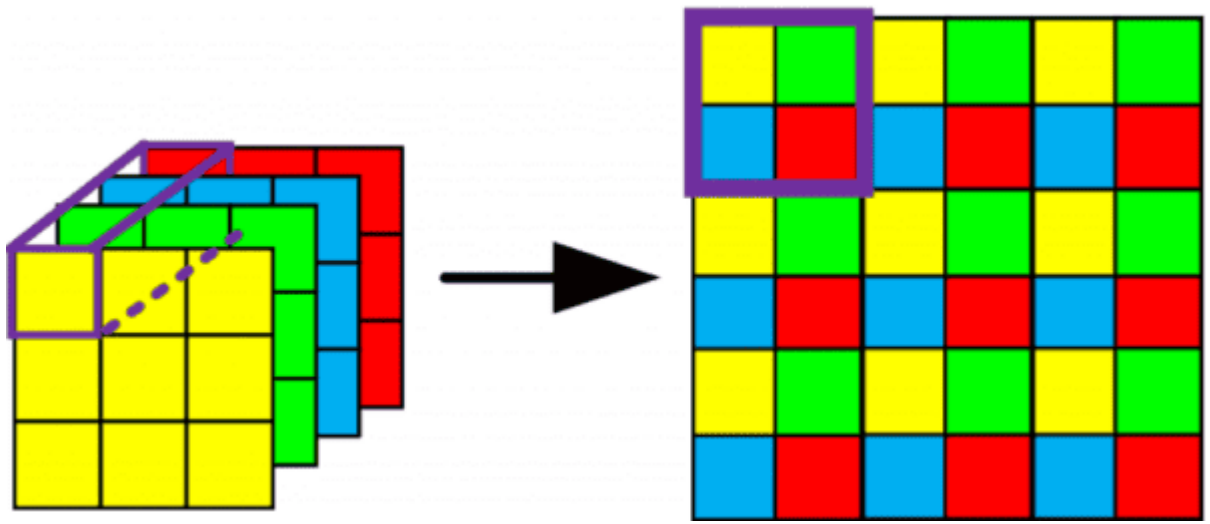


Figure 16. **Model architecture.** The above diagram shows the model architecture for Vista4D. The fire icon indicates trainable parameters. We build upon `Wan2.1-T2V-14B` [2], and we omit timestep conditioning, text prompt to token embedding, modulation, layer normalization, output unshuffle, and diffusion model denoising in the diagram for simplicity. All patchify layers are initialized from the base video model besides that of the point cloud render alpha mask, which is zero-initialized. The camera encoder is zero-initialized, and the projector after self-attention is initialized as the identity affine transformation.



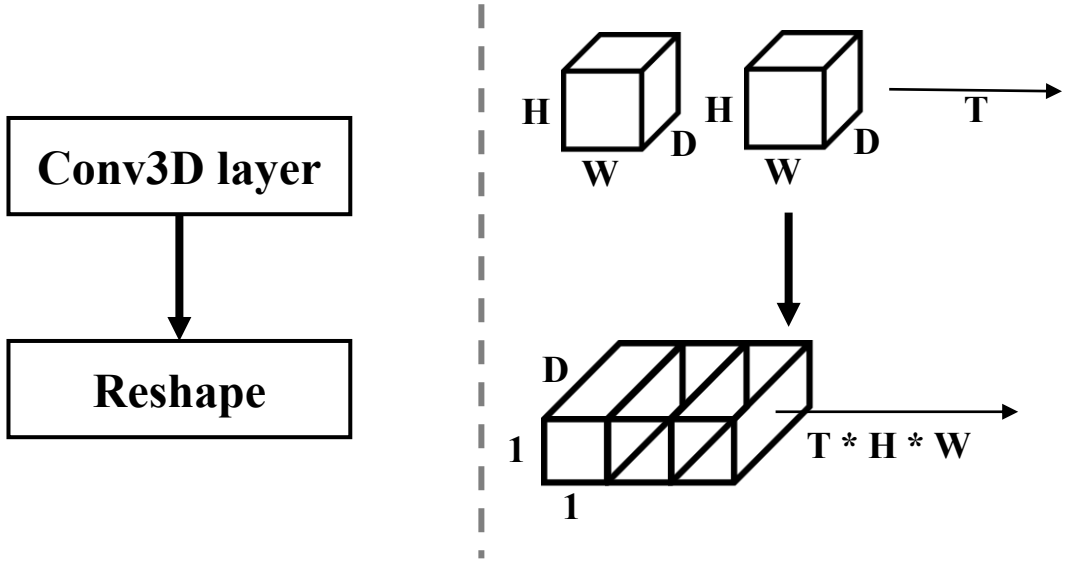
"why (Token) Shuffling?"

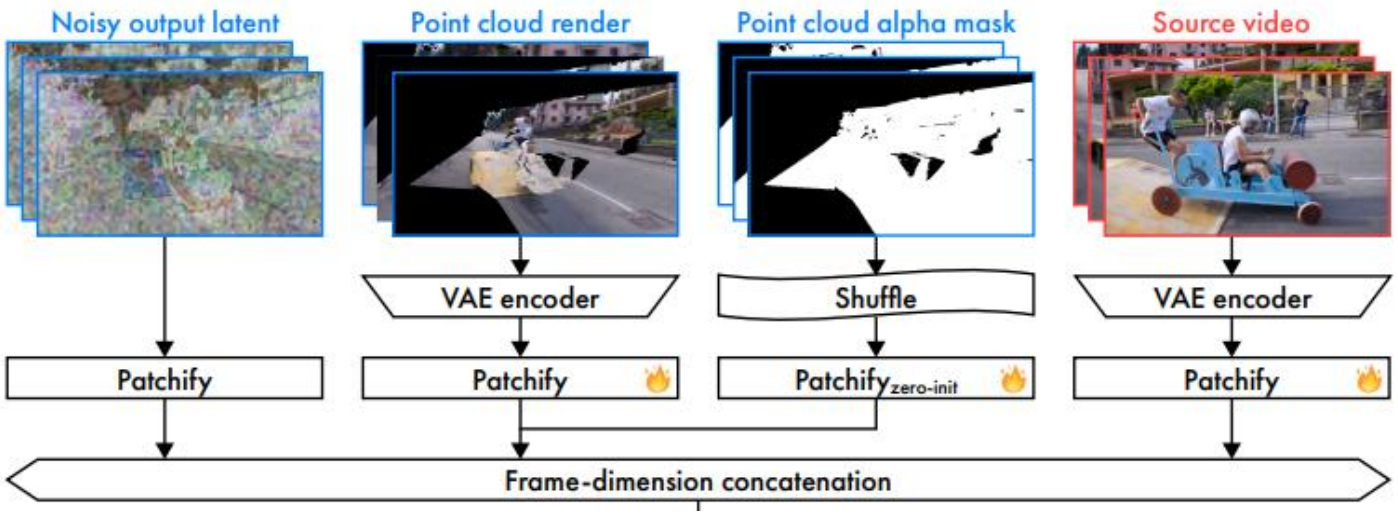
1. Alpha masks are binary and mostly empty, so a heavy VAE encoder is unnecessary.
2. Other inputs are spatially compressed by the VAE, so the mask resolution must be matched before concatenation.
3. The spatial resolution should be reduced while preserving as much spatial information as possible.



"why Patchify?"

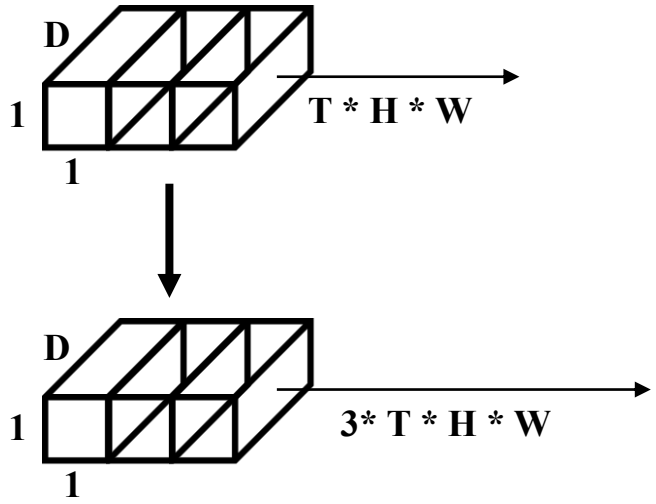
1. Patchification converts the shuffled mask into the token format required by the DiT.
2. Token shuffling only rearranges raw alpha values; it does not learn feature representations.
3. The mask tokens must math the point-cloud tokens in both token layout and embedding dimension for element-wise addition.

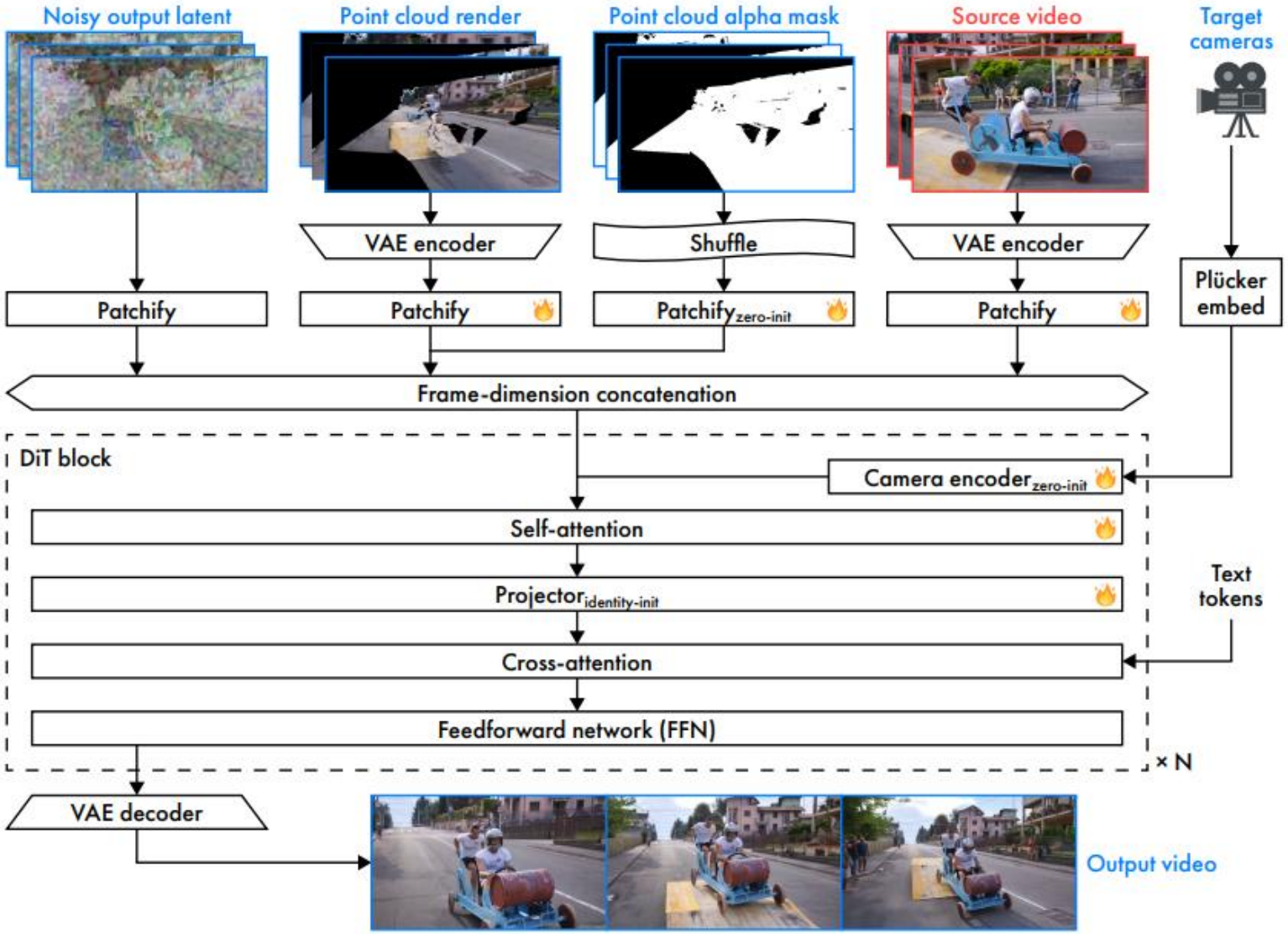


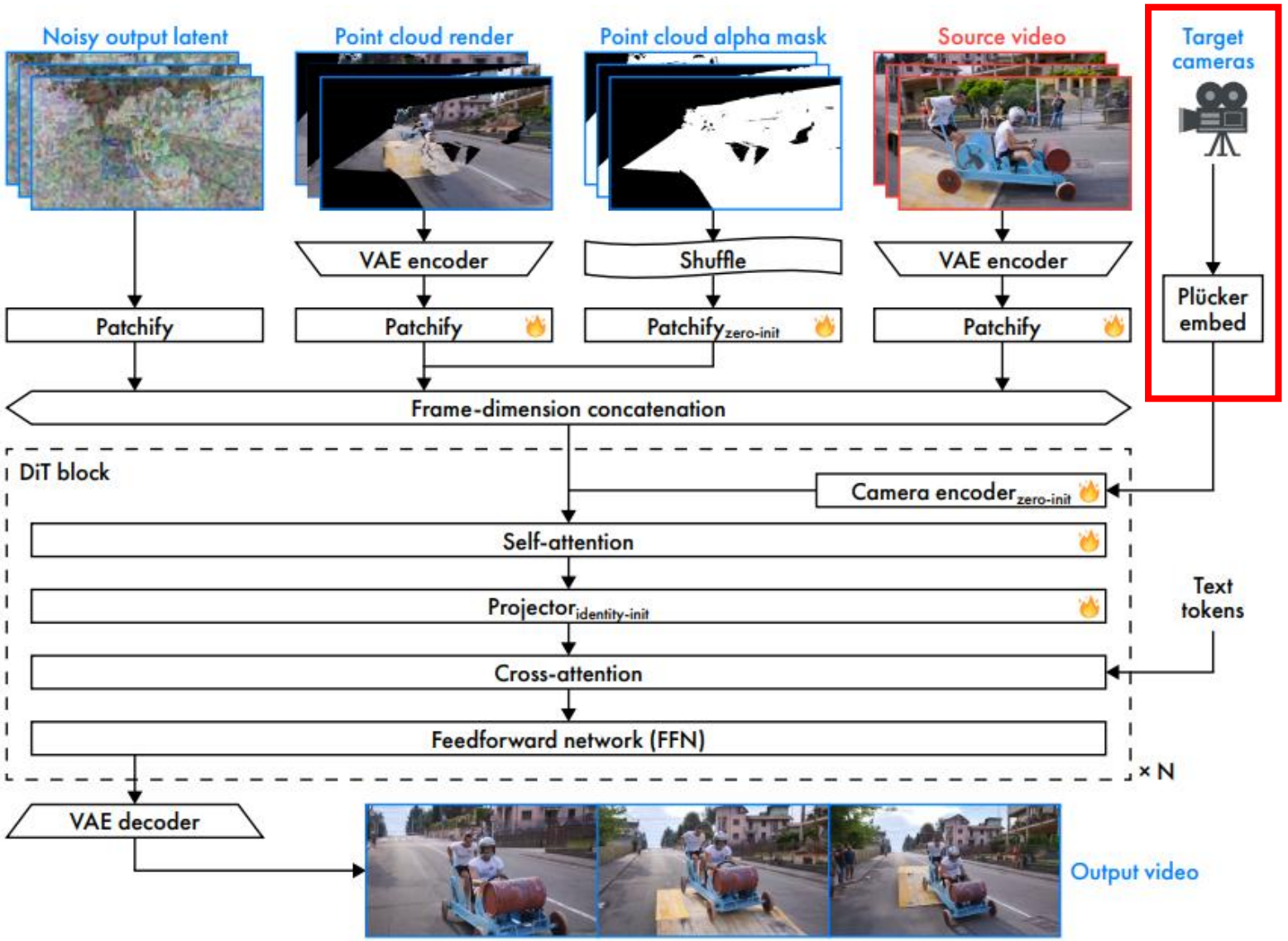


"why Concatenation?"

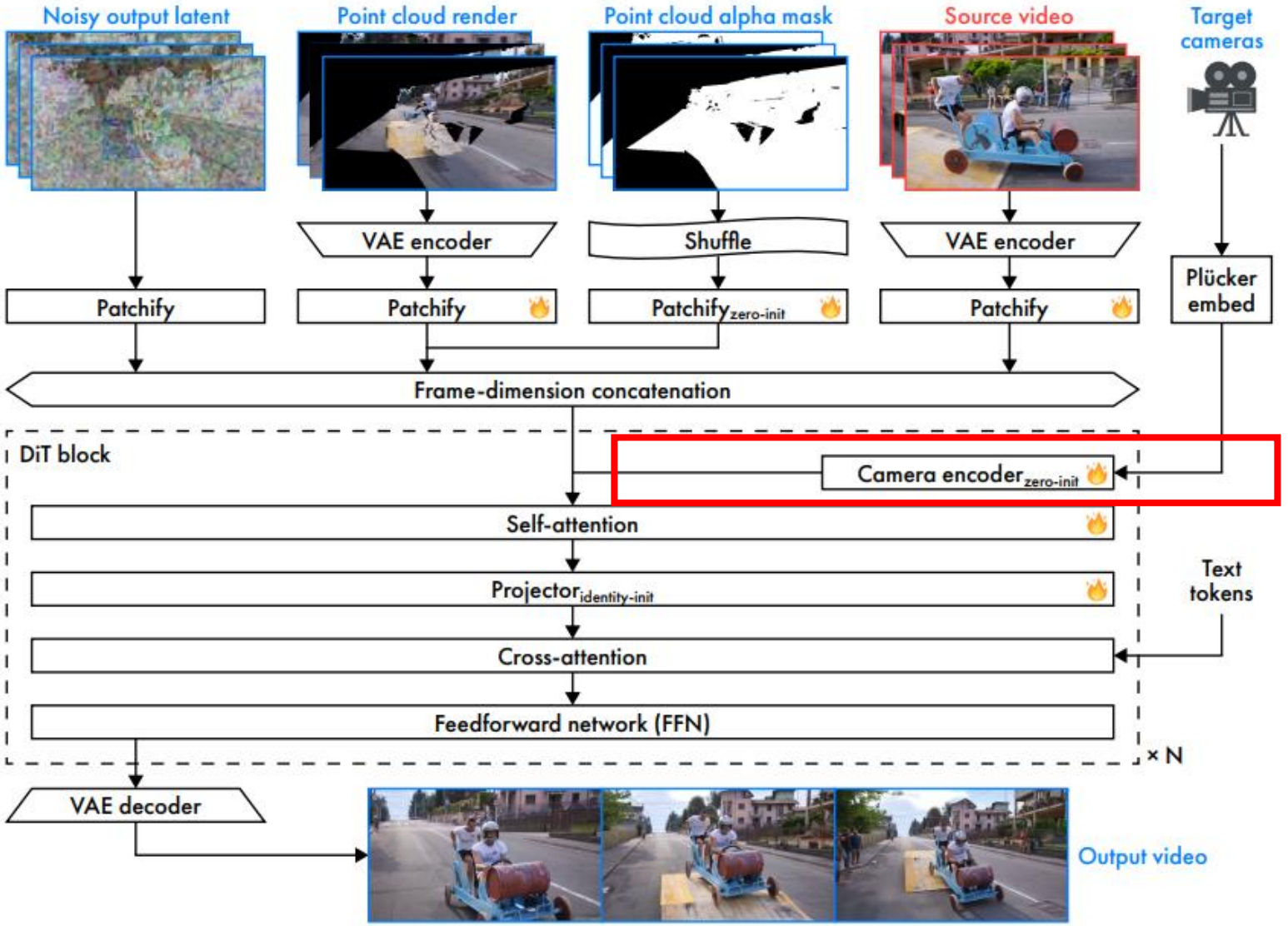
1. It places all visual inputs in a shared context while preserving their individual tokens.
2. Self-attention dynamically selects and integrates information from each input.
3. Frame-wise concatenation keeps the token dimension D unchanged for compatibility with the pretrained DiT.



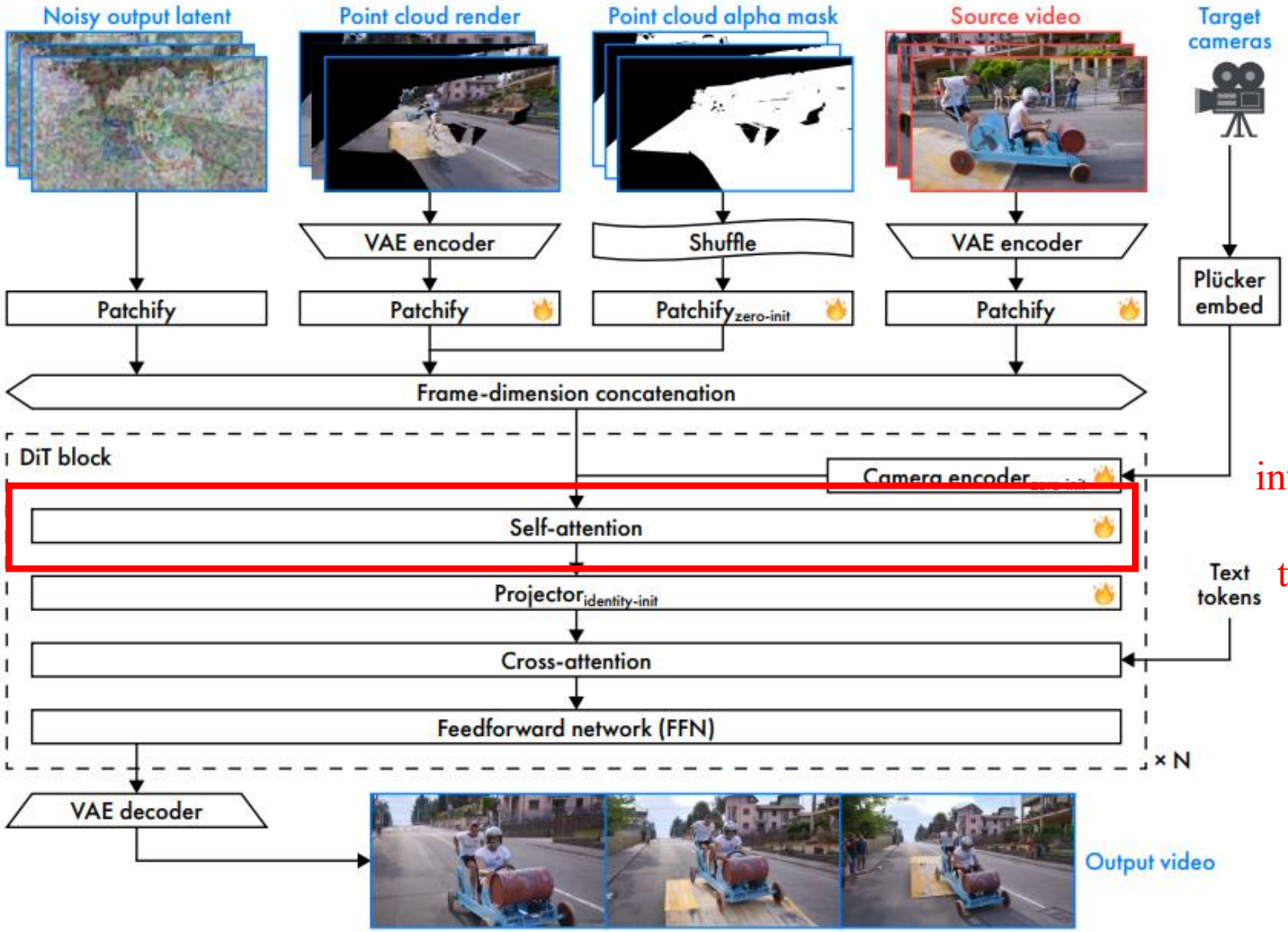

Noisy output latent
Point cloud render
Point cloud alpha mask
Source video
Target cameras
VAE encoder
Shuffle
VAE encoder
Plücker embed
Patchify
Patchify
Patchify zero-init
Patchify
Frame-dimension concatenation
DiT block
Camera encoder zero-init
Self-attention
Projector identity-init
Text tokens
Cross-attention
Feedforward network (FFN)
× N
VAE decoder
Output video



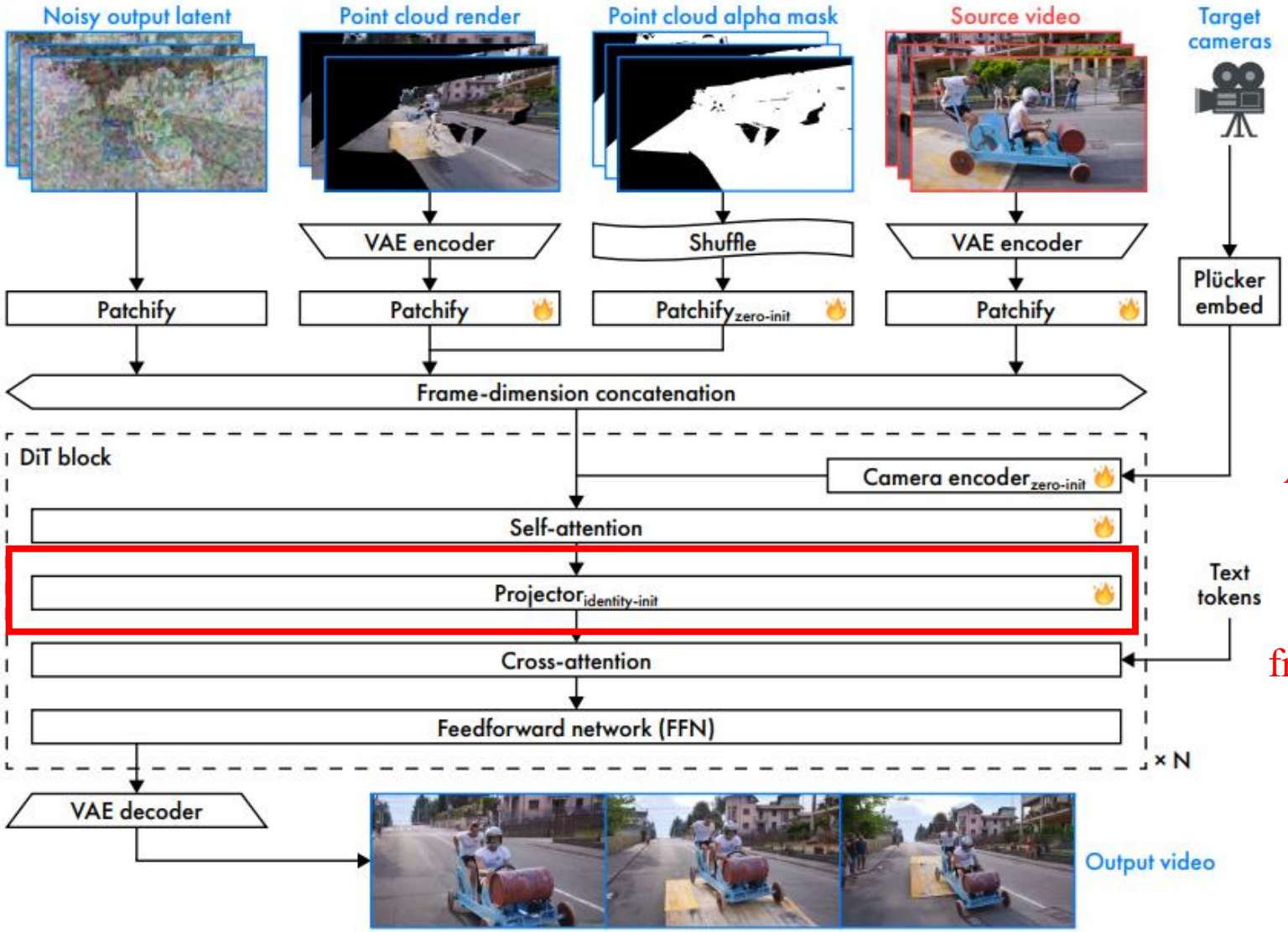
Converts camera pose information into numerical ray representation that the model can process.
(repeat 3 times)



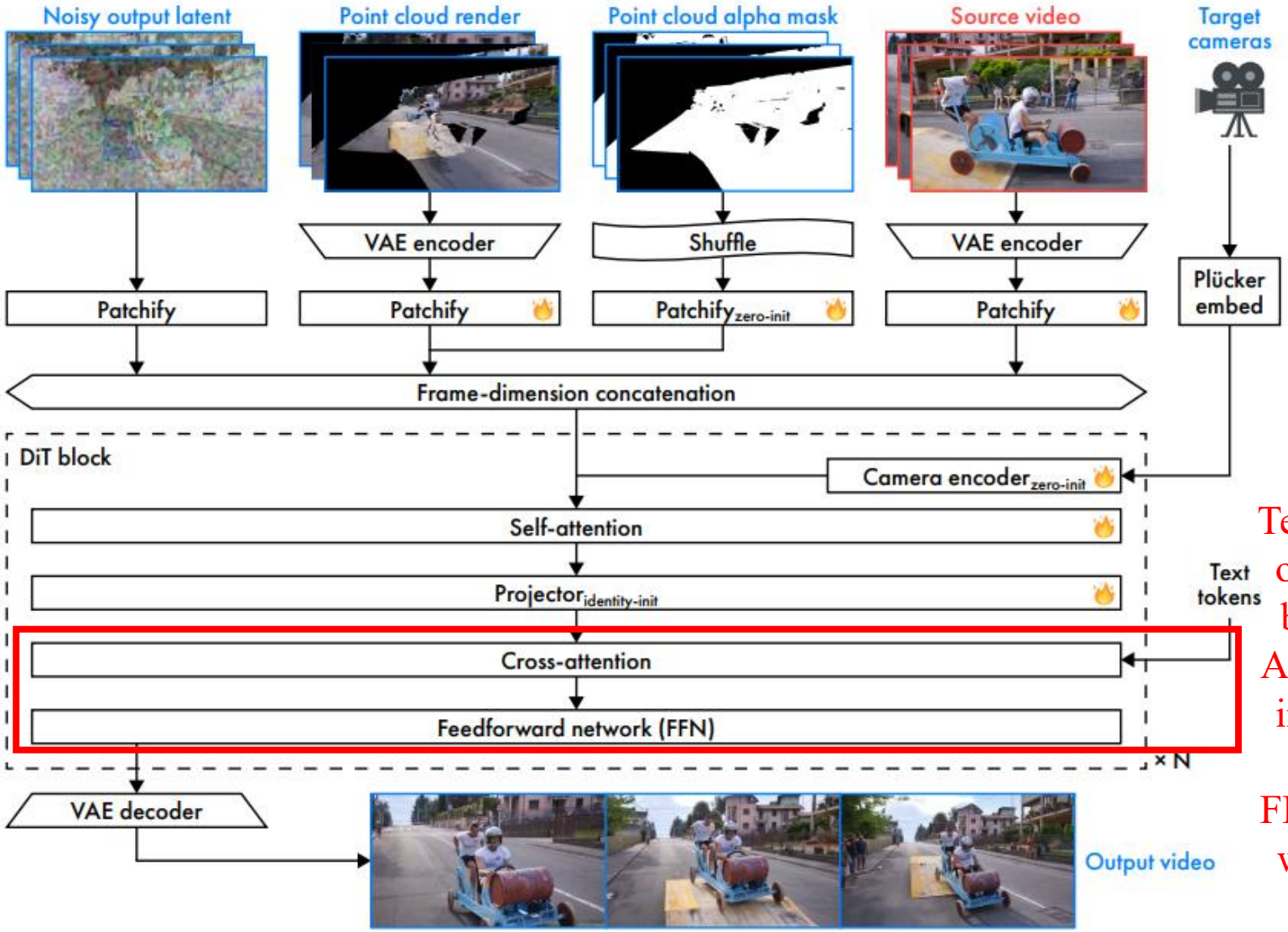
Projects Plucker embeddings into D-dimensional camera features



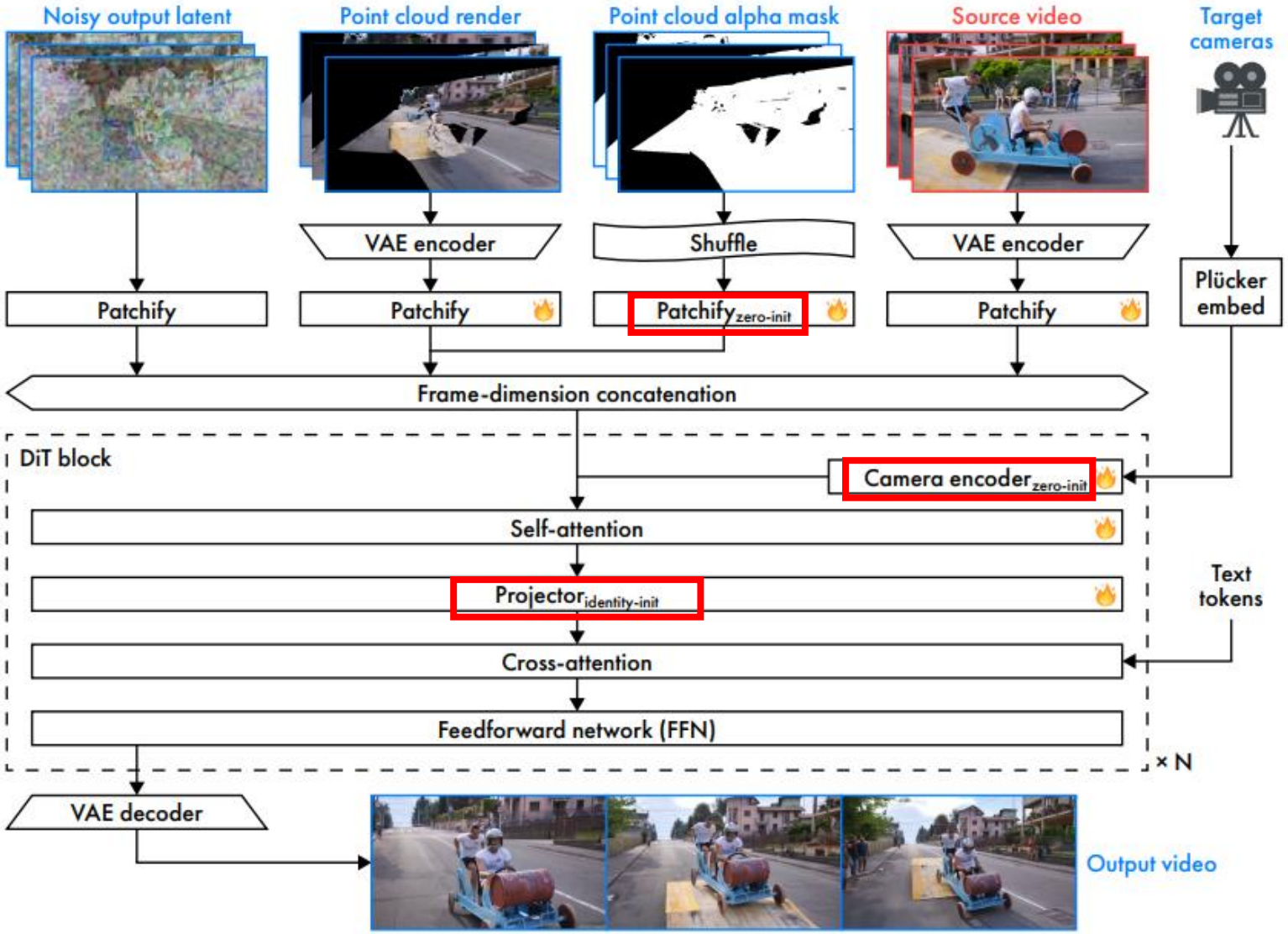
Exchange and integrate information across the noisy target, point-cloud condition, and source video.



Act as an adapter between the trainable self-attention and the frozen downstream layers.



Text token is caption of video (This will be used as Value). Attention exchanges information across tokens, while the FFN refines features within each token.


Noisy output latent
Point cloud render
Point cloud alpha mask
Source video
Target cameras
VAE encoder
Shuffle
VAE encoder
Plücker embed
Patchify
Patchify
Patchify zero-init
Patchify
Frame-dimension concatenation
DiT block
Camera encoder zero-init
Self-attention
Projector identity-init
Text tokens
Cross-attention
Feedforward network (FFN)
× N
VAE decoder
Output video

$$\mathcal{L} = \left\| \epsilon_\theta \left(\mathbf{X}_t^{\text{tgt}}, \mathbf{X}^{\text{src}\to\text{tgt}}, \mathbf{M}^{\text{src}\to\text{tgt}}, \mathbf{X}^{\text{src}}, \mathbf{C}^{\text{tgt}}, t \right) - \mathbf{V} \right\|$$

- ϵ_θ: $Denoising\ net\ work\ parameterized\ by\ \theta$
- X_t^{tgt}: $Noisy\ target - video\ latent\ at\ time\ step\ t$
- $X^{src\to tgt}$: $Point - clod\ render\ from\ the\ target\ cameras$
- $M^{src\to tgt}$: $Alpha\ mask\ of\ the\ point - cloud\ render$
- X^{src}: $Source - video\ latent$
- C^{tgt}: $Target\ camera\ trajectory$
- $V(= X^{tgt} - \epsilon)$: $Ground - truth\ velocity\ target$ (Flow matching)

Results

MuLILAB

Table 1. **Camera control accuracy and 3D consistency.** Vista4D consistently shows the most accurate camera control compared to baselines with superior rotation, translation, and intrinsics errors. Our method also significantly outperforms baselines in per-frame 3D consistency with the lowest reprojection error under SuperGlue (RE@SG) [57–59]. **Bold** indicates best results.

Method	Translation error ⇓	Rotation error ⇓	Intrinsics error ⇓	RE@SG ⇓
ReCamMaster [10]	1.574	12.79	11.16	23.66
CamCloneMaster [22]	2.132	23.77	6.422	23.38
TrajectoryCrafter [7]	1.434	6.838	6.671	120.5
EX-4D [9]	1.325	5.941	5.182	13.11
GEN3C [8]	1.309	4.751	5.085	12.99
Vista4D (ours)	**1.251**	**4.647**	**4.927**	**7.504**

Table 2. **Novel-view video synthesis.** Vista4D shows comparable to superior noel-view video synthesis performance on the `iphone` dataset [60]. EPE (endpoint error) measures optical flow error between the generated and ground truth videos and indicates scene motion reconstruction. **Bold** indicates best results.

Method	mPSNR ⇑	mSSIM ⇑	mLPIPS ⇓	PSNR ⇑	SSIM ⇑	LPIPS ⇓	EPE ⇓
ReCamMaster [10]	10.84	0.444	0.692	10.96	0.262	0.755	4.681
CamCloneMaster [22]	11.14	0.444	0.651	11.17	0.260	0.713	4.318
TrajectoryCrafter [7]	13.82	**0.492**	0.569	13.06	**0.320**	0.656	2.375
EX-4D [9]	12.85	0.479	0.596	12.64	0.305	0.669	4.269
GEN3C [8]	12.19	0.447	0.608	12.06	0.260	0.679	3.019
Vista4D (ours)	**14.09**	0.480	**0.461**	**14.14**	0.310	**0.514**	**1.142**

MuLILAB

Table 3. **Video fidelity.** Vista4D consistently outperform point-cloud-conditioned (explicit-prior) baselines for the video fidelity metrics FID, FVD, CLIP-T, and metrics from VBench [61] and VBench-2.0 [62]. Implicit-prior methods (ReCamMaster and CamCloneMaster) outperform our method in some metrics due to their low camera control accuracy (Table 1) that result in output videos with similar, usually more static, cameras to the input video which produces better FID, FVD, and VBench consistency metrics. **Bold** indicates best results.

Method	Camera control	FID ⇓	FVD $\times 10^3$ ⇓	CLIP-T ⇑	VBench [61] & VBench-2.0 [62]					
					Aesthetic quality ⇑	Imaging quality ⇑	Subject consistency ⇑	Background consistency ⇑	Temporal style ⇑	Human anatomy ⇑
ReCamMaster [10]	Extrinsics	**94.15**	**1.203**	0.319	0.552	0.701	**0.913**	**0.934**	0.243	0.759
CamCloneMaster [22]	Ref. video	101.4	1.406	0.321	0.560	0.709	0.886	0.915	0.247	0.711
TrajectoryCrafter [7]	Point cloud	125.6	1.640	0.305	0.509	0.650	0.854	0.906	0.241	0.790
EX-4D [9]	Point cloud	124.6	1.481	0.296	0.480	0.660	0.849	0.894	0.226	0.687
GEN3C [8]	Point cloud	113.5	1.441	0.318	0.519	0.660	0.857	0.913	0.245	0.775
Vista4D (ours)	Point cloud	105.4	1.418	**0.326**	**0.567**	**0.716**	0.883	0.916	**0.253**	**0.857**

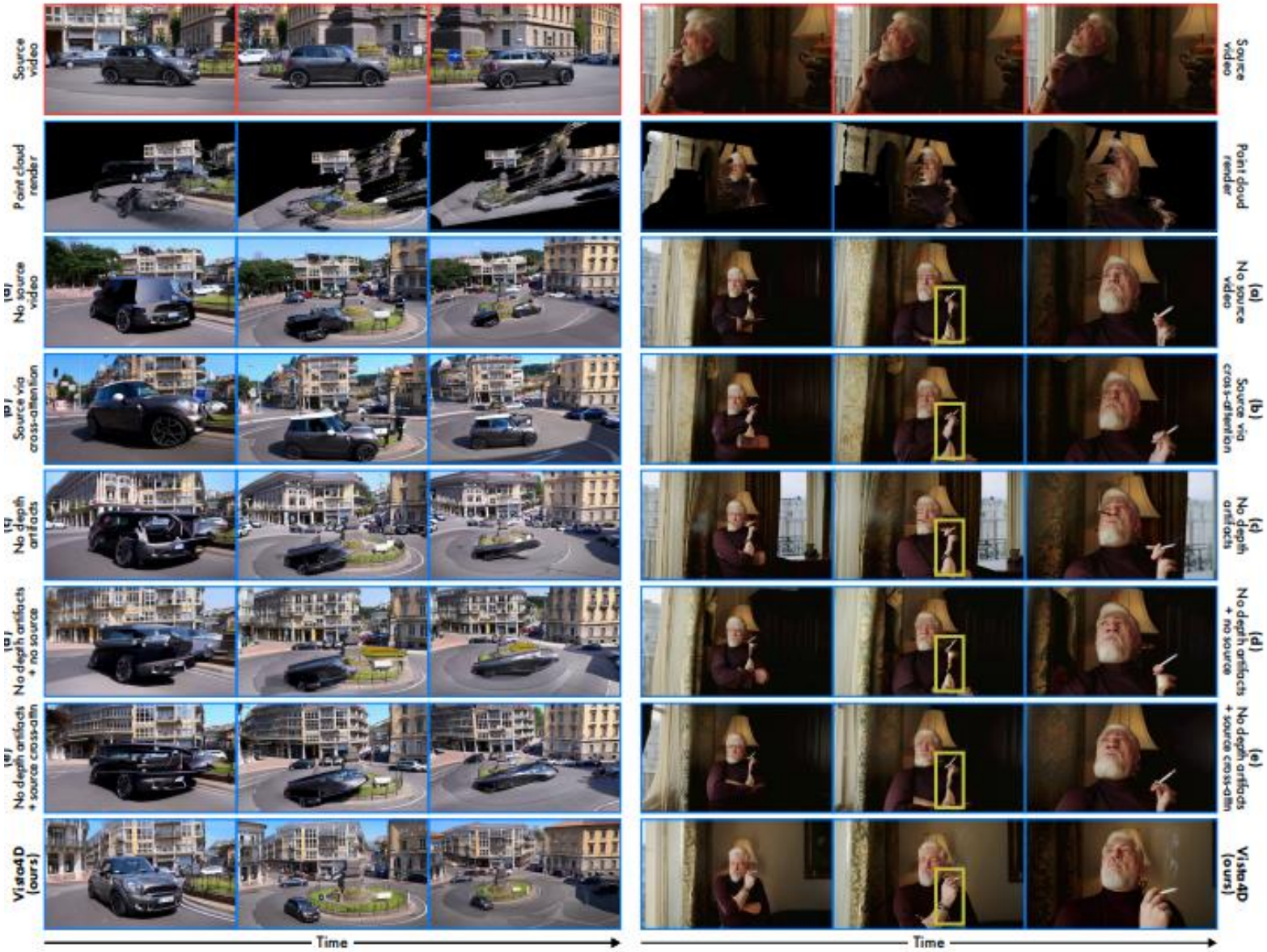
MuLILAB

Table 4. **User study.** Participants consistently perfer Vista4D over baselines on source video content preservation, camera control accuracy, and overall video fidelity. **Bold** indicates best results.

Method	Source preservation ⇑	Camera accuracy ⇑	Overall fidelity ⇑
ReCamMaster [10]	9.921%	1.905%	4.365%
CamCloneMaster [22]	15.63%	6.429%	11.03%
TrajectoryCrafter [7]	0.952%	5.952%	0.476%
EX-4D [9]	1.587%	6.508%	0.794%
GEN3C [8]	4.841%	11.03%	5.952%
Vista4D (ours)	**67.06%**	**68.17%**	**77.38%**

04 Results

MuLILAB



Conclusion

05 Conclusion

MuLILAB

1. **Proposal of a Novel Video Reshooting Framework**
2. **Robustness to Point Cloud Artifacts**
3. **State-of-the-Art Performance**
4. **Generalization to Real-World Applications**

Applications

Figure 7. **Dynamic scene expansion.** With our 4D-grounded temporally-persistent point cloud, Vista4D can do video reshooting with additional scene information from casual scene captures or alternate angles by doing joint 4D reconstruction of these frames with the source video. Doing so reduces video model hallucinations and provides stronger control beyond the source video.

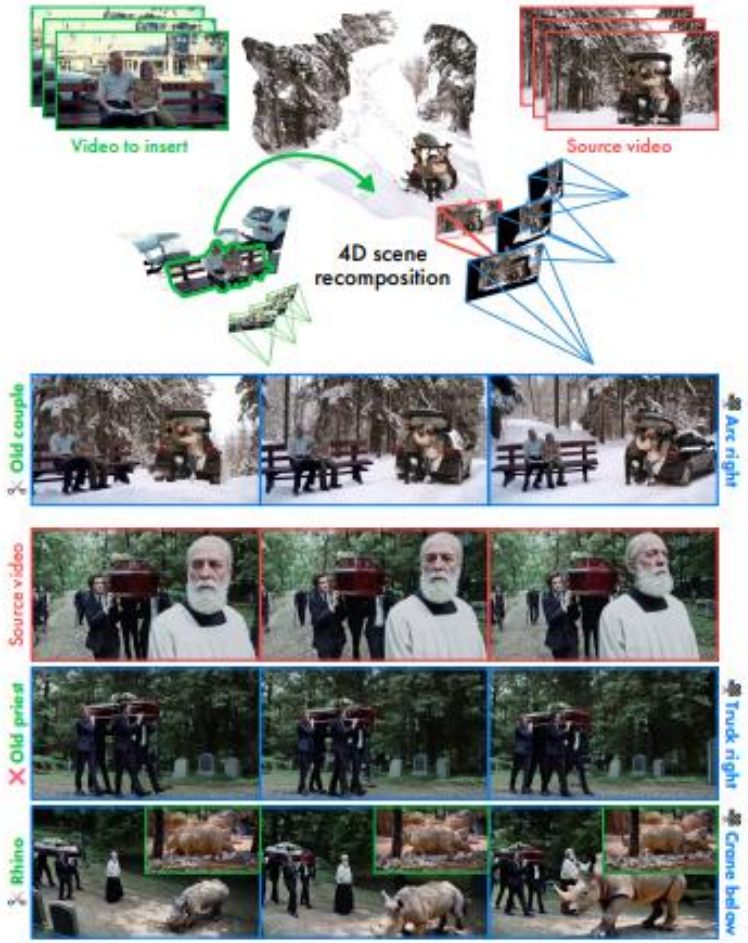


Figure 8. **4D scene recomposition.** By directly editing the 4D point cloud, Vista4D can recompose 4D scenes from the source video or other inserted videos. Importantly, our method synthesizes physically plausible lighting when inserting a rhino lit by sunlight through leaves into an otherwise overcast scene.

Figure 9. **Long video inference with memory.** Vista4D can reshoot long videos by doing inference in chunks. By registering static pixels of newly generated chunks back into the temporally-persistent 4D point cloud, Vista4D maintains an explicit, 4D-grounded memory of generated content. We showcase this above as the camera arcs around the scene, indicated with color-matched yellow and pink boxes.

Thank You

2026.08.03

BrainLAB Journal Club
Department of Applied Artificial Intelligence
Jeong SangYeop