A. Preliminary: Video Diffusion Models

Modern video diffusion models operate in a compact latent space learned by a spatio-temporal VAE. Given a video $x \in \mathbb{R}^{T\times H\times W\times 3}$, the encoder E maps it to latent features $z_0 = E(x) \in \mathbb{R}^{T'\times C\times H'\times W'}$, on which the generative process is defined [1, 9]. A standard forward diffusion process gradually perturbs z_0 into noisy variables z_t via

$$q(z_t \mid z_0) = \sqrt{\alpha_t}\, z_0 + \sqrt{1-\alpha_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \quad (1)$$

and a denoiser ϵ_θ is trained to predict the noise under time step t and conditioning signal c (e.g., text prompts or reference frames) as

$$\mathcal{L}_{\text{diff}}(\theta) = \mathbb{E}_{z_0,t,\epsilon}\big[\|\epsilon_\theta(z_t, t, c) - \epsilon\|_2^2\big], \quad (2)$$

following the DDPM formulation [3]. Recent video generators further adopt continuous-time flow matching. Given clean latents z_0 and a Gaussian samples z_1, one defines linear interpolants $z_\tau = (1-\tau)z_0 + \tau z_1$ with $\tau \in [0, 1]$ and learns a velocity field v_θ by

$$\mathcal{L}_{\text{flow}}(\theta) = \mathbb{E}_{z_0,\tau,\epsilon}\big[\|v_\theta(z_\tau, \tau, c) - (z_1 - z_0)\|_2^2\big], \quad (3)$$

as in flow-matching and related ODE-based generative formulations [5, 7]. These objectives are typically implemented with Diffusion Transformers (DiT), which operate on spatio-temporal latent tokens and inject (t, c) through attention [8], forming the backbone of current foundation video generators.

Wan2.1 instantiates the above latent video diffusion / flow-matching paradigm with a 3D VAE and a DiT-based denoiser, together with rich multi-modal conditioning trained on large-scale, diverse video–text data [10]. In VerseCrafter, we adopt Wan2.1-14B as a frozen latent video diffusion backbone and treat it as a generic video prior. Specifically, we keep the Wan Encoder, Wan-DiT, and Wan Decoder unchanged, and attach a lightweight geometry-aware control interface, namely GeoAdapter, to selected Wan-DiT blocks. The detailed architecture of VerseCrafter is provided in the Sec. B.

B. Model Architecture Details

VerseCrafter is built on top of the Wan2.1 T2V-14B backbone [10], a latent video diffusion / flow-matching model with a 3D VAE (Wan Encoder and Wan Decoder) and a DiT-based denoiser (Wan-DiT). As shown in Fig. 1, we keep the Wan2.1 backbone frozen and introduce a geometry-aware conditioning pathway with a lightweight GeoAdapter that conditions selected Wan-DiT blocks on rendered 4D control maps. Table 1 summarizes the input resolution, number of Wan-DiT layers, hidden dimension, GeoAdapter injection pattern, and fine-tuning configuration of VerseCrafter.

Table 1. **Model configuration of VerseCrafter.** Settings include the final output resolution, number of Wan-DiT layers, GeoAdapter injection blocks, pre-trained backbone, and training schedule.

	VerseCrafter
Final resolution	720P
Num. Wan-DiT layers	40
GeoAdapter injection blocks	$[0, 5, 10, 15, 20, 25, 30, 35]$
Pre-trained backbone	Wan2.1 T2V-14B
Hidden dimension	5120
Batch size	16
Training schedule	2,500 it. @480P + 2,500 it. @720P

Geometry encoding and tokenization. For each frame t, we render background RGB/depth RGB_t^{bg}, $\text{Depth}_t^{\text{bg}}$, 3D Gaussian trajectory RGB/depth $\text{RGB}_t^{\text{traj}}$, $\text{Depth}_t^{\text{traj}}$, and a soft merged mask M_t that marks regions where the diffusion model should synthesize or overwrite content. For t=1, we replace RGB_1^{bg} with the input image and set M_1=0. The four RGB/depth maps are encoded by the frozen Wan Encoder to obtain latent features at the VAE resolution, while the mask $M \in \mathbb{R}^{1\times T\times H\times W}$ is rearranged to align with the latent grid of Wan Encoder (the "Rearrange" module in Fig. 1). Let s_t, s_h, and s_w denote the temporal and spatial strides of Wan Encoder (we use s_t=4 and s_h=s_w=8). Following the practice in [4, 10], we drop the singleton channel dimension, split the spatial dimensions into $s_h \times s_w$ sub-cells, and fold these sub-cells into the channel dimension via a reshape–permute operation, yielding a tensor of shape $C_M \times T \times H' \times W'$ with C_M=$s_h s_w$, H'=H/s_h, and W'=W/s_w. We then downsample the temporal dimension using nearest-neighbor interpolation to match the latent depth $T' = (T + s_t - 1)/s_t$, producing $\hat{M} \in \mathbb{R}^{C_M\times T'\times H'\times W'}$. Finally, $\hat{M}$ is concatenated channel-wise with the encoded background and 3D Gaussian trajectory latents to form a unified spatio-temporal geometry feature $\mathcal{G} \in \mathbb{R}^{T'\times H'\times W'\times C_{\mathcal{G}}}$. Following Wan-DiT, we partition $\mathcal{G}$ into non-overlapping 3D patches and linearly project each patch into a token embedding, yielding a sequence of geometry tokens $\mathbf{g} \in \mathbb{R}^{L\times D}$, where $L = T'H'W'$ is the number of spatio-temporal patches and D matches the hidden width of Wan-DiT. Because we use identical strides, positional encodings, and patch sizes, the geometry tokens are spatially and temporally aligned with the latent video tokens processed by Wan-DiT.

GeoAdapter integration. GeoAdapter is a lightweight DiT-style branch that operates on geometry tokens $\mathbf{g}$. It shares the same hidden dimensionality and positional encodings as Wan-DiT, but contains far fewer layers. Let $\{\mathcal{B}_1, \ldots, \mathcal{B}_N\}$ denote N Wan-DiT blocks, and let $\{\mathcal{G}_1, \ldots, \mathcal{G}_M\}$ denote M GeoAdapter blocks. We attach GeoAdapter as a residual modulation branch to a subset of

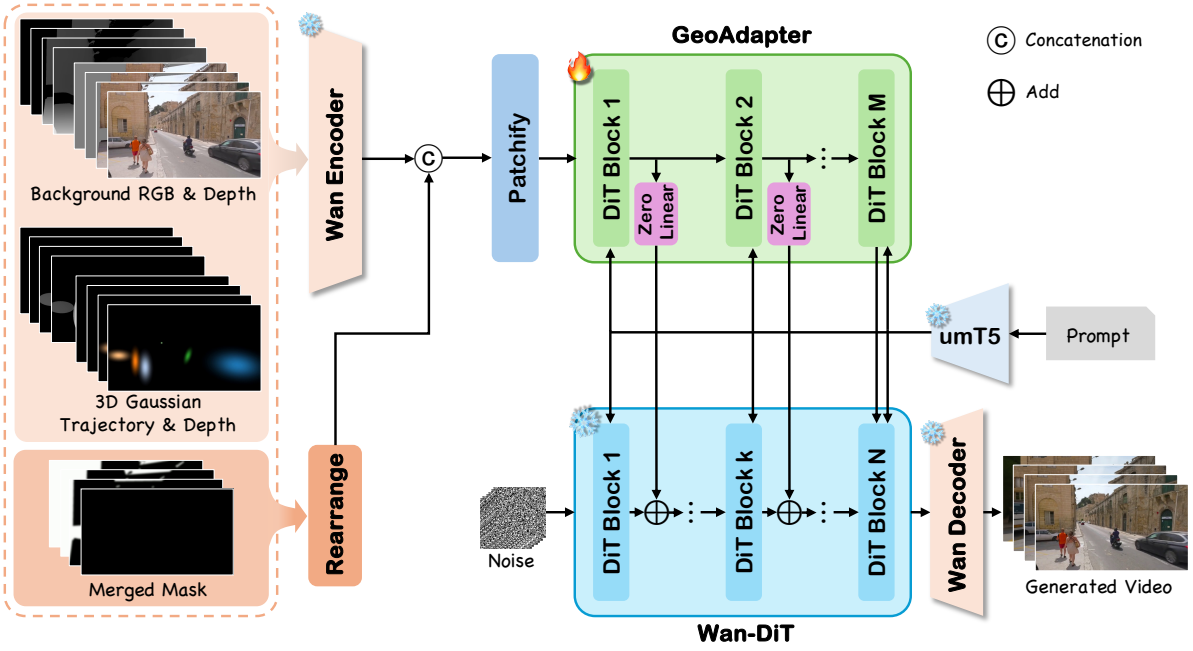

Figure 1. **Detailed architecture of VerseCrafter.** Background RGB & depth maps and 3D Gaussian trajectory RGB & depth maps are first encoded by the frozen Wan Encoder. The soft merged mask is rearranged into latent-aligned channels, and all geometry latents are then concatenated along the channel dimension to form a unified spatio-temporal geometry feature. This feature is patchified into tokens and processed by the GeoAdapter branch. At selected Wan-DiT blocks, GeoAdapter outputs are passed through zero-initialized linear layers and added to the backbone tokens as residual modulations, enabling geometry-consistent control over camera motion and multi-object motion.

Wan-DiT blocks. Concretely, we choose a stride k and inject GeoAdapter after every k-th Wan-DiT block; see Table 1 for the exact injection pattern and configuration. For each Wan-DiT block $\mathcal{B}_n$ whose index n belongs to the injection set, with input tokens $\mathbf{x}_n \in \mathbb{R}^{L\times D}$ and geometry tokens $\mathbf{g}$, we add a geometry-conditioned residual of the form

$$\mathbf{x}_{n+1} = \mathcal{B}_n(\mathbf{x}_n) + \mathcal{G}_m(\mathbf{g})\,\mathbf{W}_0^{(m)}, \tag{4}$$

where $\mathcal{G}_m$ is the corresponding GeoAdapter block and $\mathbf{W}_0^{(m)} \in \mathbb{R}^{D\times D}$ is its output projection. Each GeoAdapter block is initialized from the weights of its paired Wan-DiT block for stable training, while $\mathbf{W}_0^{(m)}$ is initialized to zero. As a result, VerseCrafter behaves identically to the original Wan2.1 backbone at the beginning of training. During fine-tuning, $\mathbf{W}_0^{(m)}$ gradually learns to inject geometry information through residual modulation, in the spirit of zero-initialized adapter designs such as ControlNet [12].

C. VerseControl4D Dataset Details

We construct **VerseControl4D**, a large-scale real-world video dataset with automatically derived prompts and rendered 4D control maps. As described in the main paper, VerseControl4D is built through four stages: data collection, clip extraction, quality filtering, and data annotation. The rendered 4D control maps comprise background RGB/depth maps, 3D Gaussian trajectory RGB/depth maps, and a soft merged mask.

Table 2. **VerseControl4D data split and scene-type statistics.** We report the number of samples from each source dataset and split. *Dynamic scenes* contain coupled camera motion and foreground object motion, while *static scenes* have negligible object motion and are used for camera-only evaluation.

Split	Sekai-Real-HQ	SpatialVID-HQ	
	Dynamic Scenes	Static Scenes	Dynamic Scenes
Train	9,071	7,000	18,929
Validation	468	250	282

VerseControl4D contains 35,000 training samples and 1,000 validation samples. Table 2 summarizes the data distribution by source dataset and scene type. Overall, 26% of the samples come from Sekai-Real-HQ and 74% from SpatialVID-HQ, reflecting their complementary scene coverage. To support both camera-only world exploration and joint camera-object control, VerseControl4D includes *dynamic scenes* (clips with salient foreground object motion together with camera motion) and *static scenes* (clips with negligible object motion and only camera motion). About

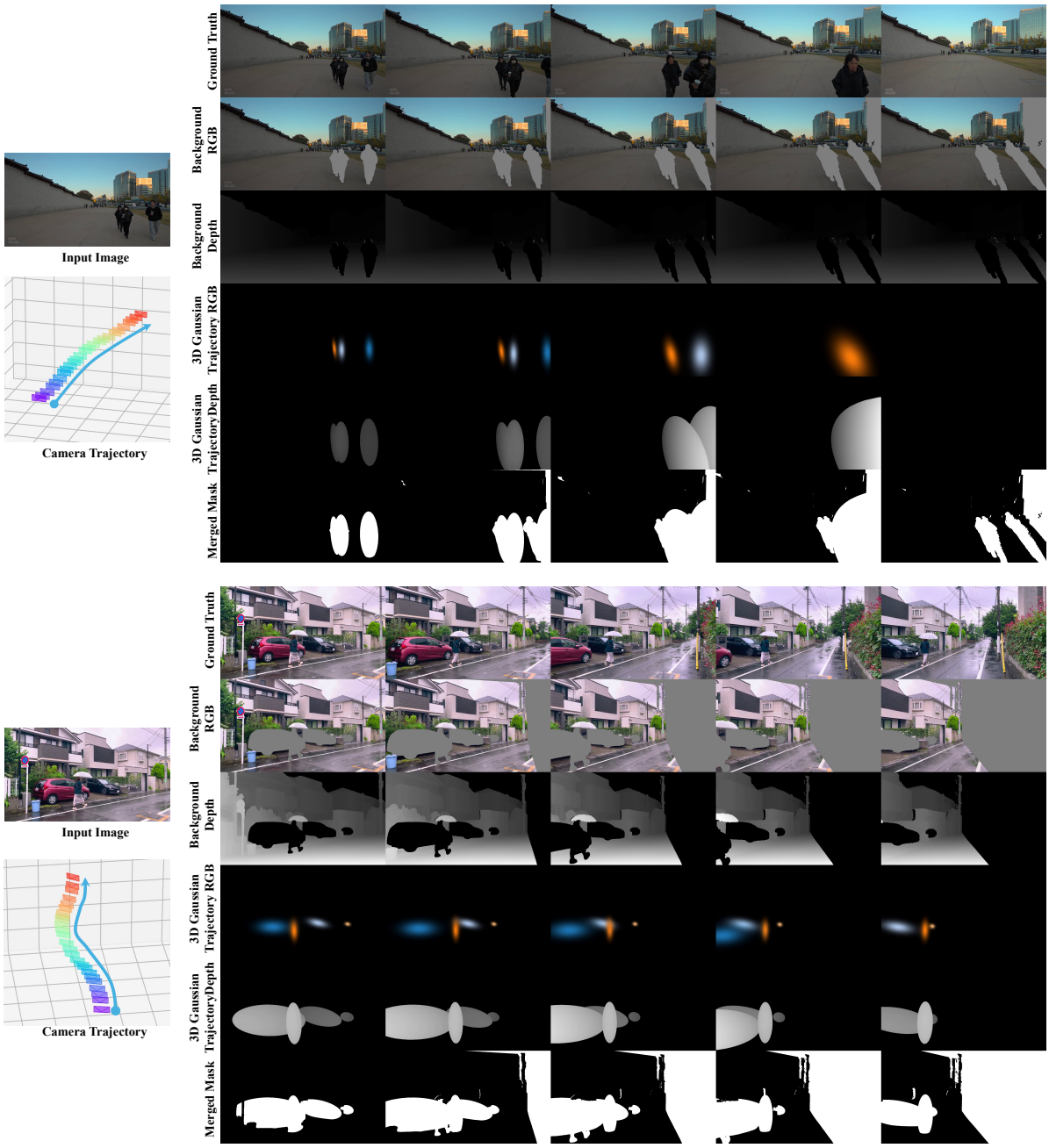

Figure 2. **VerseControl4D dataset examples.** For each clip, we visualize the input image and target camera trajectory (left), followed by several frames of ground-truth video and our rendered control signals (right): background RGB/depth, 3D Gaussian trajectory RGB/depth for controlled objects, and the final merged mask. These signals are automatically derived by our annotation pipeline in main paper.

20% of the training samples are *static scenes*, and the validation set includes 250 static-scene samples for dedicated camera-only evaluation. Representative samples and their rendered 4D control signals are shown in Fig. 2.

D. Evaluation Metrics

D.1. VBench-I2V

We evaluate image-to-video generation quality using the VBench Image-to-Video (I2V) evaluation suite, denoted as VBench-I2V. For each generated clip, we follow the official VBench-I2V protocol: the conditioning image and its corresponding generated video are fed into the evaluation

pipeline, which computes a set of learned, human-aligned metrics that jointly capture video-image consistency and perceptual video quality. In our experiments, we report the following eight VBench-I2V dimensions, and define the *Overall Score* as the simple arithmetic mean of these eight normalized scores, where higher values indicate better performance:

- **Imaging Quality.** This metric measures low-level image fidelity, including sharpness and the absence of artifacts such as blur, noise, or overexposure. VBench uses an image quality predictor (e.g., MUSIQ) and averages its scores across frames to obtain a video-level imaging-quality score.
- **Aesthetic Quality.** This dimension assesses the artistic and aesthetic appeal of individual frames, including composition, color harmony, and realism. VBench applies an aesthetic quality predictor (e.g., the LAION aesthetic model) to each frame and averages the predictions over the clip.
- **Dynamic Degree.** This metric quantifies how dynamic the generated video is. Optical flow magnitudes (e.g., estimated by RAFT) are used to measure the amount of motion, and the score reflects whether the model produces sufficiently active (non-static) content.
- **Motion Smoothness.** This metric evaluates whether subject and camera motion evolve smoothly and follows reasonable physical dynamics. VBench leverages a pretrained video frame interpolation prior to assess how well intermediate motion can be interpolated, with smoother and more physically plausible motion receiving higher scores.
- **Background Consistency.** This dimension measures the temporal stability of background layout and texture. Frame-level features (e.g., CLIP) are compared across time; large feature variations indicate flickering or unstable backgrounds and lead to lower scores.
- **Subject Consistency.** This dimension evaluates the temporal consistency of the foreground subject *within* the video, regardless of the input image. VBench computes subject-region features across frames and measures their similarity over time to penalize identity drift or sudden appearance changes.
- **I2V Background (Video–Image Background Consistency).** This metric evaluates how well the global background in the video matches the background in the input image, especially for scene-centric inputs. VBench uses background-sensitive features (e.g., DreamSim) and aggregates image–frame and inter-frame similarities into a single background consistency score.
- **I2V Subject (Video–Image Subject Consistency).** This metric measures how well the main subject in the generated video matches the subject in the input image. VBench extracts high-level visual features (e.g., DINO) from the conditioning image and from each video frame, and combines image–frame similarities with inter-frame similarities into a weighted average subject consistency score.

Formally, given these eight per-dimension scores $\{s_k\}_{k=1}^{8}$ returned by VBench-I2V for a video, we define

$$\text{Overall Score} = \frac{1}{8}\sum_{k=1}^{8} s_k, \tag{5}$$

which is the value reported as "Overall Score" in the main paper.

D.2. Rotation Error (RotErr)

To measure how well the generated camera motion follows the ground-truth camera trajectory, we adopt the camera-alignment metric from CameraCtrl [2]. For each generated video, we estimate its camera trajectory using the same geometry-annotation pipeline used for VerseControl4D, yielding rotation matrices $\{\mathbf{R}_{\text{gen}}^{j}\}_{j=1}^{n}$ and translation vectors $\{\mathbf{T}_{\text{gen}}^{j}\}_{j=1}^{n}$, where n is the number of frames. Let $\{\mathbf{R}_{\text{gt}}^{j}\}_{j=1}^{n}$ denote the corresponding ground-truth rotation matrices. The rotation error is computed by comparing the ground-truth and generated rotation matrices at each frame:

$$\text{RotErr} = \sum_{j=1}^{n} \arccos\left(\frac{\text{tr}\big(\mathbf{R}_{\text{gen}}^{j}\mathbf{R}_{\text{gt}}^{j\,\top}\big) - 1}{2}\right), \tag{6}$$

where $\text{tr}(\cdot)$ denotes the matrix trace. Lower RotErr indicates better alignment between the generated and ground-truth camera orientations.

D.3. Translation Error (TransErr)

We also evaluate the accuracy of the generated camera positions. Let $\{\mathbf{T}_{\text{gt}}^{j}\}_{j=1}^{n}$ and $\{\mathbf{T}_{\text{gen}}^{j}\}_{j=1}^{n}$ be the ground-truth and generated camera translation vectors for a video with n frames. Following CameraCtrl [2], the translation error is defined as the sum of per-frame Euclidean distances between the translation vectors:

$$\text{TransErr} = \sum_{j=1}^{n} \big\|\mathbf{T}_{\text{gt}}^{j} - \mathbf{T}_{\text{gen}}^{j}\big\|_2, \tag{7}$$

Lower TransErr indicates that the generated camera positions more closely match the ground-truth camera positions.

D.4. Object Motion Control (ObjMC)

For object-motion control, we follow the ObjMC metric proposed in MotionCtrl [11] and extend it to the multi-object setting under our 3D Gaussian trajectory representation. Given a generated video, we run the same geometry annotation pipeline used for VerseControl4D to estimate per-object 3D Gaussian trajectories, and compare

them with the corresponding ground-truth trajectories from our dataset.

Let N_{gt} and N_{pred} denote the numbers of ground-truth and predicted controlled objects in a sample, and let T be the number of frames. For each ground-truth object $o \in \{1, \dots, N_{\text{gt}}\}$ and frame $t \in \{1, \dots, T\}$, we denote the ground-truth 3D Gaussian mean by $\boldsymbol{\mu}_o^{(t)} \in \mathbb{R}^3$ and the estimated mean from the generated video by $\hat{\boldsymbol{\mu}}_k^{(t)} \in \mathbb{R}^3$ for a predicted object k.

Multi-object matching. Since N_{gt} and N_{pred} may differ, we first define the trajectory distance between a ground-truth object o and a predicted object k as the average Euclidean distance between their 3D Gaussian means over time:

$$d(o,k) = \frac{1}{T} \sum_{t=1}^{T} \left\| \hat{\boldsymbol{\mu}}_k^{(t)} - \boldsymbol{\mu}_o^{(t)} \right\|_2. \tag{8}$$

We then build a cost matrix $\mathbf{C} \in \mathbb{R}^{N_{\text{gt}} \times N_{\text{pred}}}$ with entries $C_{ok} = d(o,k)$. To handle unmatched objects, we pad this matrix with dummy rows and columns and fill them with a constant penalty λ (set to $10.0\,\text{m}$ in our experiments). Finally, we apply the Hungarian algorithm [6] to obtain an optimal one-to-one matching between ground-truth and predicted trajectories. This step assigns each ground-truth object either to a predicted trajectory or to a dummy entry when no suitable match exists.

ObjMC score. Given the optimal matching, we define the per-object trajectory error for a ground-truth object o as

$$d_o = \begin{cases} d(o,k) & \text{if } o \text{ is matched to a predicted object } k, \\ \lambda & \text{if } o \text{ is unmatched,} \end{cases} \tag{9}$$

and compute the final ObjMC score as the average over all ground-truth controlled objects:

$$\text{ObjMC} = \frac{1}{N_{\text{gt}}} \sum_{o=1}^{N_{\text{gt}}} d_o. \tag{10}$$

Lower ObjMC indicates more accurate multi-object 3D motion control, and the unmatched penalty λ penalizes missed objects under the one-to-one matching formulation.

E. Additional Qualitative Results

We provide additional qualitative comparisons on VerseControl4D, following the same evaluation settings and baselines as in the main paper. Fig. 3 and Fig. 4 present *dynamic scenes* with joint camera and multi-object motion control. Perception-as-Control and Uni3C often exhibit noticeable object deformation, while Yume roughly follows the text-described motion but lacks precise camera control. Uni3C is also limited to a single human and does not generalize well to diverse multi-object scenarios. In contrast, VerseCrafter more faithfully follows both the camera trajectory and multi-object motion while maintaining sharp appearance and geometrically consistent backgrounds.

Fig. 5 and Fig. 6 present *static scenes* for camera-only motion control. ViewCrafter, Voyager and FlashWorld often exhibit distorted facades, drifting structures, or inaccurate camera motion. In contrast, VerseCrafter better follows the target camera trajectory while preserving sharp details and globally consistent 3D geometry. These additional examples further demonstrate VerseCrafter's robustness under real-world 4D Geometric Control in both dynamic and static settings.

F. Additional Analysis of Control Fidelity and Robustness

We further provide targeted qualitative analyses to clarify the fidelity, scope, and robustness of our 4D Geometric Control. Specifically, we analyze orientation controllability, dynamic background modeling, articulated and non-rigid object controllability, the effect of multi-view input, and robustness to monocular-depth errors.

F.1. Control Fidelity and Boundary Cases

Orientation controllability. Our representation provides *ellipsoid-level* orientation control through the principal axes of each 3D Gaussian, rather than fine-grained 6D pose control. As shown in Fig. 7(a), this control is reliable for strongly anisotropic rigid objects such as cars and buses, where rotation induces clear changes in rendered footprint and depth, leading to stable orientation cues after 3D→2D rendering. However, it can fail for human-like subjects approximated by a single ellipsoid. In such cases, heading changes mainly correspond to rotation around the ellipsoid's major principal axis, and when the other two axes are similar, the projected footprint/depth variation becomes subtle and ambiguous. As a result, the geometric cue may be too weak to fully determine facing direction, and the diffusion prior may dominate, occasionally causing heading mismatches.

Dynamic background modeling. Our background point cloud is reconstructed from the first frame and therefore serves as a mostly static geometric scaffold. It anchors scene geometry under viewpoint changes, but does not explicitly model per-frame non-rigid background deformation. Fig. 7(b) shows that this design still works well for moderate dynamic-background effects such as wind-swaying grass and a flowing river, where the diffusion prior can synthesize plausible temporal variation while the 4D controls maintain camera and object consistency. In contrast, the waterfall example exhibits weaker motion. This failure is expected because fine, texture-dominant, highly

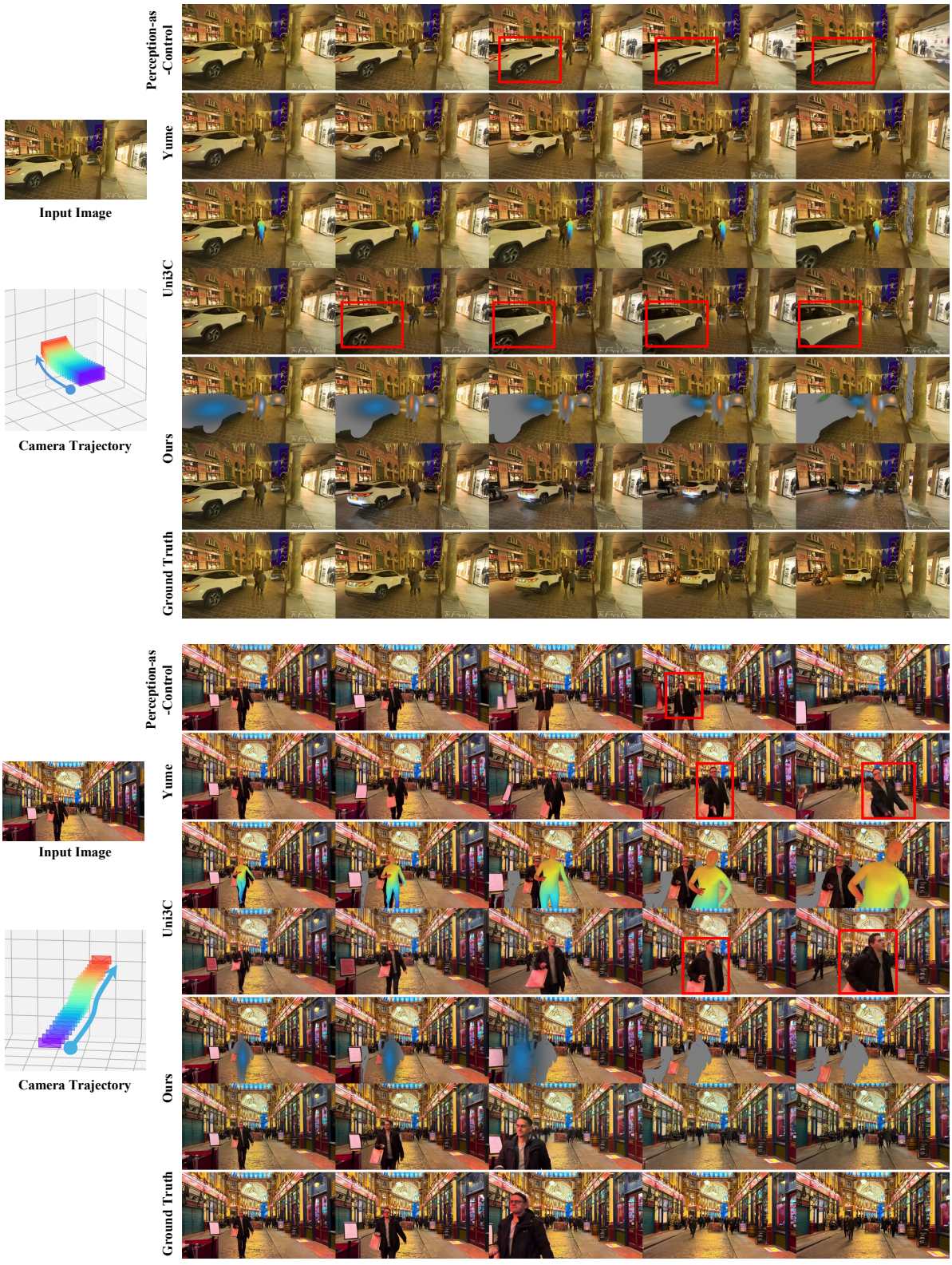

Figure 3. **Additional qualitative comparison of joint camera and object motion control.** Perception-as-Control and Uni3C exhibit noticeable object deformation, while Yume roughly follows the text-described motion but lacks precise camera control. Uni3C is also limited to a single human. In contrast, VerseCrafter more faithfully follows both the camera trajectory and multi-object motion while maintaining sharp appearance and geometrically consistent backgrounds.

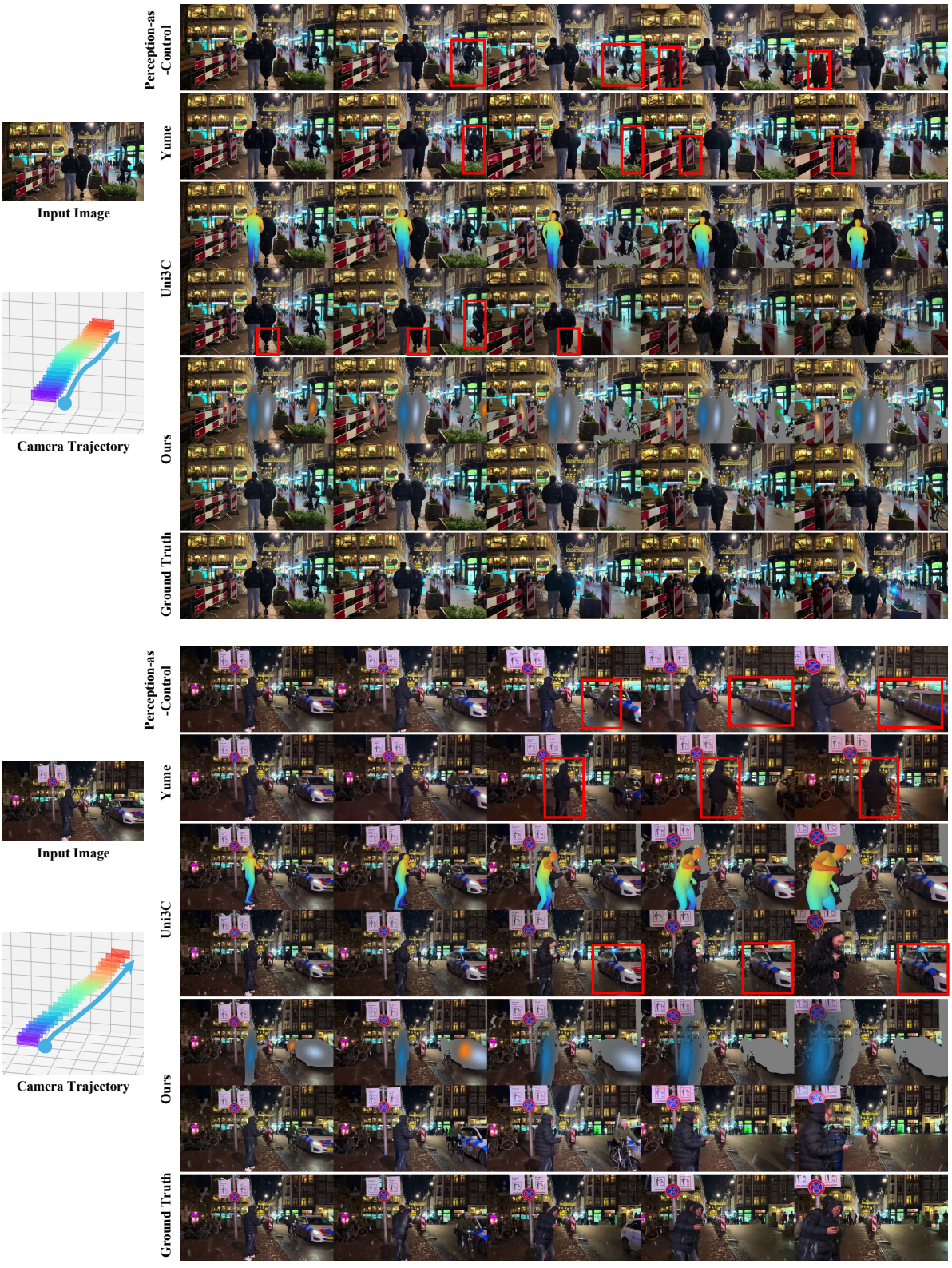

Figure 4. **Additional qualitative comparison of joint camera and object motion control.** Perception-as-Control and Uni3C exhibit noticeable object deformation, while Yume roughly follows the text-described motion but lacks precise camera control. Uni3C is also limited to a single human. In contrast, VerseCrafter more faithfully follows both the camera trajectory and multi-object motion while maintaining sharp appearance and geometrically consistent backgrounds.

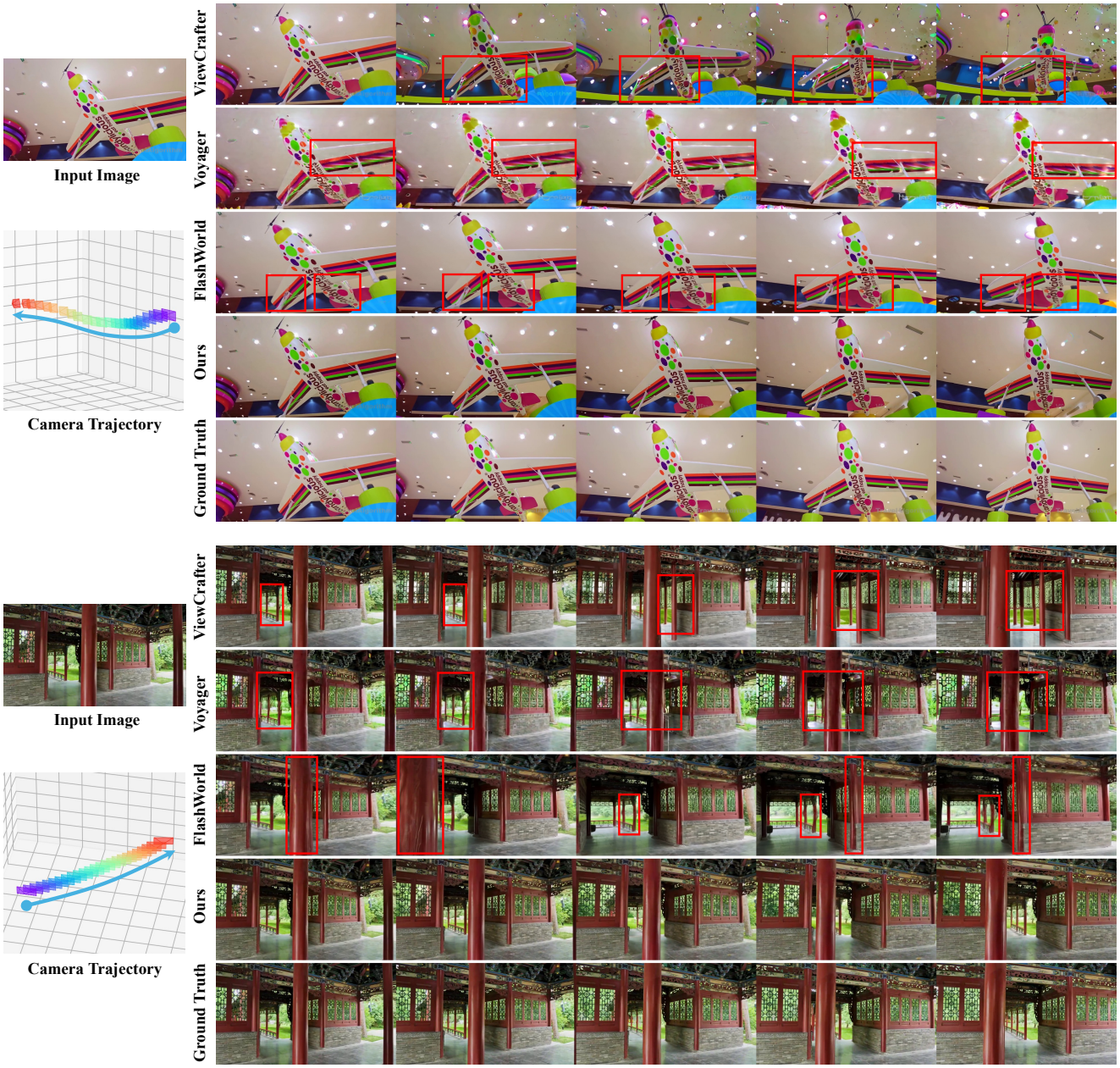

Figure 5. **Additional qualitative comparison of camera-only motion control on static scenes.** ViewCrafter, Voyager and FlashWorld exhibit distorted facades, drifting structures, or inaccurate camera motion. In contrast, VerseCrafter better follows the target camera trajectory while preserving sharp details and globally consistent 3D geometry.

non-rigid dynamics are only weakly constrained by a static 3D scaffold after rendering to 2D control maps. Thus, VerseCrafter currently handles dynamic backgrounds mainly through the interaction between static geometry anchoring and the video prior, rather than through explicit background dynamics modeling.

Articulated and non-rigid object controllability. VerseCrafter uses a *single* 3D Gaussian per controlled object and guides its motion coarsely by changing its position, scale, and orientation over time. This representation is not designed to explicitly encode part-level articulation. Nevertheless, Fig. 7(c) shows that it remains effective for object-level motion control in both articulated and non-rigid scenarios, including a robotic arm extension and wind-blown clothes. Although the guidance is coarse, the generated videos follow the intended object-level motion while remaining visually coherent. These examples suggest that even a simple object-level 3D Gaussian can provide a useful

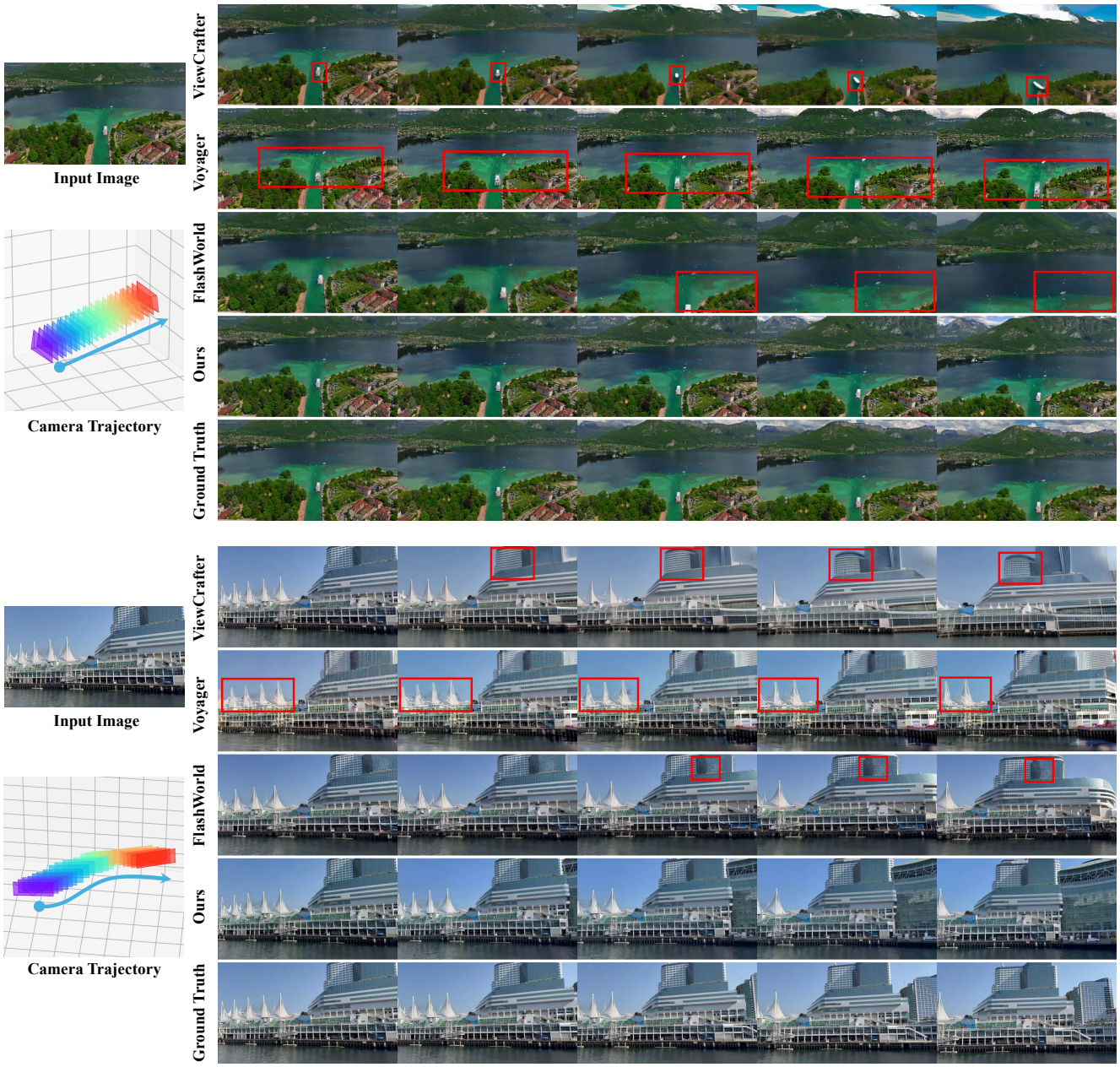

Figure 6. **Additional qualitative comparison of camera-only motion control on static scenes.** ViewCrafter, Voyager and FlashWorld exhibit distorted facades, drifting structures, or inaccurate camera motion. In contrast, VerseCrafter better follows the target camera trajectory while preserving sharp details and globally consistent 3D geometry.

control signal for a broad range of dynamic objects, though finer articulation control remains an important direction for future work.

F.2. Geometry Coverage and Robustness

Single-view vs. multi-view input. Multi-view input improves geometry coverage and therefore improves novel-view faithfulness. As shown in Fig. 8(a), using multiple views to reconstruct the scene expands the point cloud to cover regions that are weakly observed or fully invisible from a single reference image, such as the tram side and rear structure. Consequently, the generated video is more faithful under larger viewpoint changes; for example, the rear door is recovered only in the multi-view case. By comparison, single-view reconstruction still produces plausible videos because the diffusion prior can fill in under-constrained regions, but the missing geometry leads to less faithful novel-view synthesis. This result supports our claim

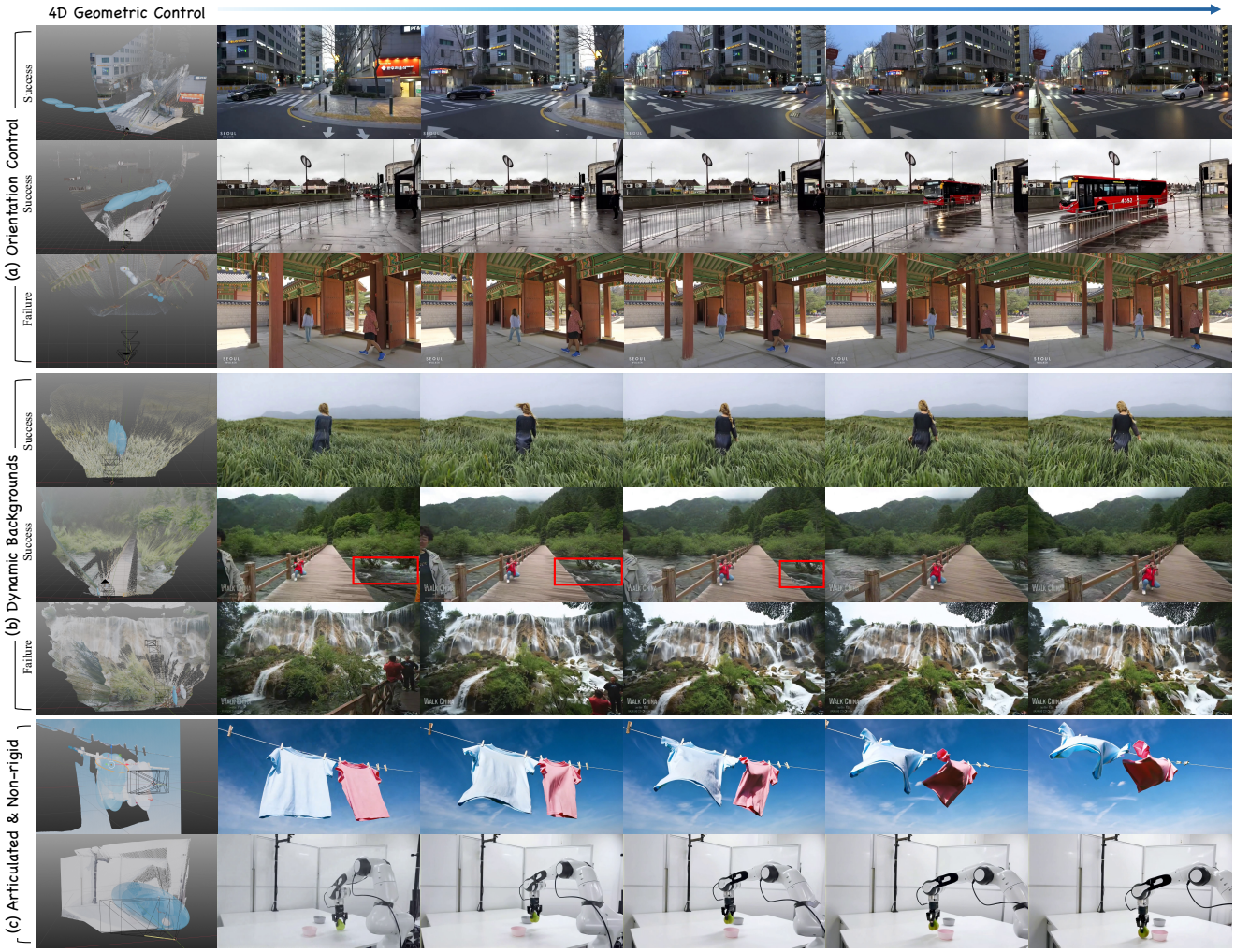

Figure 7. **Additional analysis of control fidelity and boundary cases. (a) Orientation controllability.** Two success cases on rigid anisotropic objects and one failure case on a human-like subject. **(b) Dynamic background modeling.** Two success cases on moderate background dynamics and one failure case on highly non-rigid background motion. **(c) Articulated and non-rigid object controllability.** Two examples showing effective coarse object-level control on articulated and non-rigid objects.

that more complete 3D reconstruction directly benefits controllable video generation when the target camera motion departs significantly from the reference view.

Robustness to monocular-depth errors. We also test robustness to imperfect monocular depth estimation in challenging conditions with heavy occlusion and strong illumination variation. In Fig. 8(b), replacing MoGe-2 with MiDaS v2.1 produces visibly noisier depth and distorted point clouds in difficult regions such as the pillars highlighted by the red boxes. Despite these geometry errors, the generated videos remain visually similar and preserve the main building structure. This robustness is expected because the reconstructed point cloud acts as a *coarse* geometric scaffold rather than a per-pixel hard constraint. After 3D→2D rendering, the diffusion prior can compensate for moderate depth noise and still generate structurally plausible results. Therefore, while better monocular geometry generally improves controllability, VerseCrafter does not critically depend on perfectly accurate depth estimates.

Overall, these analyses suggest that VerseCrafter is most effective when the underlying 4D geometric cues are sufficiently informative, while failures mainly arise in underconstrained cases such as subtle human orientation changes or highly non-rigid background dynamics.

G. Inference Efficiency and Memory Usage

We further analyze the inference cost of VerseCrafter. For generating an 81-frame 720P video on 8×96GB GPUs, Table 4 shows that diffusion inference is the dominant bottleneck, while 4D geometric state construction is cacheable across repeated edits of the same scene and diffusion model loading is a one-time startup cost. Accordingly, the per-edit

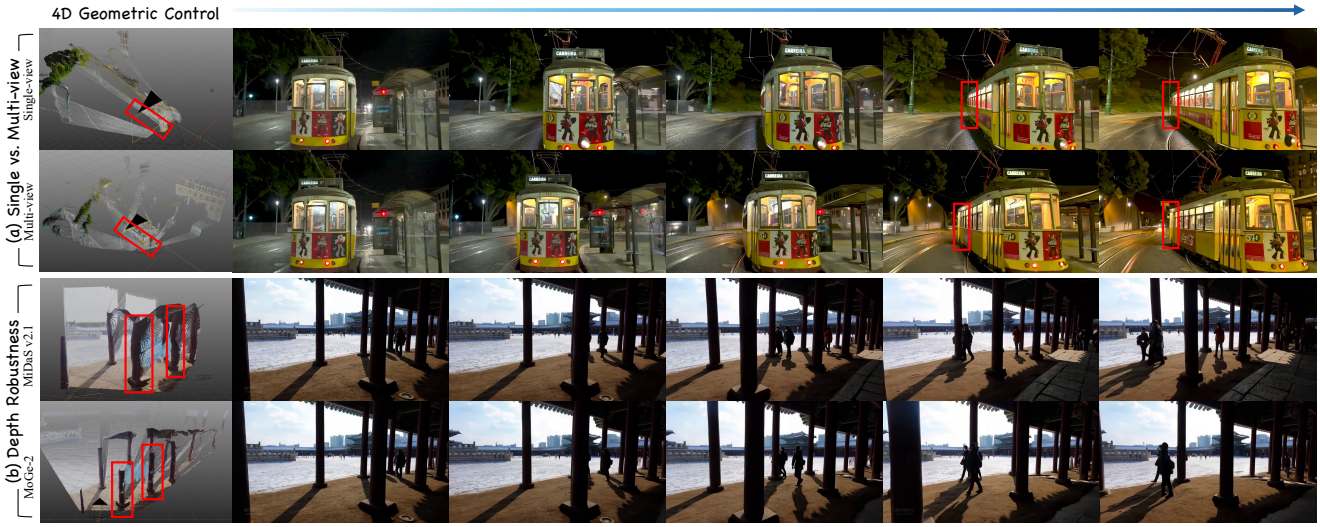

Figure 8. **Additional analysis of geometry coverage and robustness. (a) Single-view vs. multi-view input.** Multi-view reconstruction improves geometry coverage and novel-view faithfulness. **(b) Robustness to monocular-depth errors.** Even with noisier depth and distorted point clouds, the generated videos remain visually similar and preserve the main scene structure.

Table 3. **Memory–time trade-off under different inference settings.** We report peak per-GPU memory and diffusion inference time for the 50-step setting. FSDP reduces peak memory from 90 GB to 70 GB with negligible runtime change, and FSDP + CPU offload further reduces it to 57 GB (36.7% reduction) with only a small increase in diffusion inference time.

Inference setting	Peak GPU memory (GB)	Memory reduction (%)	Diffusion inference time (s)
Baseline	90	0.0	866
Baseline + FSDP	70	22.2	870
Baseline + FSDP + CPU offload	57	36.7	880

Table 4. **Stage-wise end-to-end inference latency and cacheability.** We report the runtime breakdown for generating an 81-frame 720P video on 8×96GB GPUs. 4D geometric scene state is reusable across repeated edits of the same scene, and model loading is a one-time startup cost, whereas 4D control rendering and diffusion sampling must be rerun when the edited controls change.

Stage	Time (s) ↓	Cacheable?
4D Geometric State Construction	∼23	✓
4D Control Maps Rendering	∼60	✗
Diffusion Sampling		
Diffusion Model Loading	∼203	✓
Diffusion Inference	∼866 (50 steps)	✗
Diffusion Inference	∼715 (30 steps)	✗

latency is substantially reduced for subsequent edits, and can be further lowered by using fewer denoising steps.

Table 3 summarizes the memory and runtime trade-off under different inference settings. FSDP substantially reduces peak per-GPU memory with negligible runtime overhead, and FSDP + CPU offload further lowers memory at only a small additional cost. These results suggest that the current practical bottleneck is diffusion inference rather than 4D geometric state construction.

H. Limitations and Future Work

Despite the encouraging results, VerseCrafter still has several limitations that suggest promising directions for future work.

First, our current object representation provides only *ellipsoid-level* control through a single 3D Gaussian per object, which limits fine-grained pose and part-level articulation, especially for human-like or near-symmetric objects. More expressive object representations, such as multiple Gaussians per object or articulated 3D structures, may improve fine-grained orientation and pose control.

Second, our background point cloud is reconstructed from the first frame and serves as a mostly static geometric scaffold, which limits controllability for highly non-rigid and texture-dominant scene dynamics such as waterfalls. Incorporating explicit dynamic background representations or temporally evolving scene geometry may improve controllability in such cases.

Third, although VerseCrafter enforces 4D geometric consistency through explicit camera control and 3D Gaus-

sian trajectory control, it does not impose explicit physical constraints during generation. Integrating stronger physics priors, such as collision-aware losses, contact constraints, ground constraints, or differentiable physics guidance, could improve physical realism and controllability in complex interactions.

Finally, VerseCrafter remains computationally expensive at high resolution and long temporal horizons because it conditions a large frozen video diffusion backbone and renders multi-channel 4D controls for all frames. Future work may explore more efficient backbones, distilled sampling, cached control encoding, and streaming or long-video synthesis to enable faster and longer world rollouts.

References

[1] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22563–22575, 2023. 1

[2] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for text-to-video generation. *arXiv preprint arXiv:2404.02101*, 2024. 4

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1

[4] Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. Vace: All-in-one video creation and editing. *arXiv preprint arXiv:2503.07598*, 2025. 1

[5] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022. 1

[6] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 5

[7] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 1

[8] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023. 1

[9] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1

[10] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 1

[11] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan. Motionctrl: A unified and flexible motion controller for video generation. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 4

[12] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847, 2023. 2