CODE VIDEO MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Video diffusion models generate realistic videos in a black box. Physics is imagined rather than computed; object placement, motion, and camera trajectories are resolved internally and only visible in the final clip. Inspired by film production, we present Code Video Model, which gives video generation an executable previs, with code as the engine. Given a prompt, Code Video Model writes a program that lays out a scene, animates objects, places a camera, and computes dynamics from equations; rendering the program yields a proxy video and a reference image. Rendering the resulting proxy video takes less than a minute, making it *fast* to preview; it is *grounded*, since dynamics are computed, and *directly and precisely controllable*, because code can modify each element. A pretrained video model is then conditioned on the proxy video and reference image to generate the final clip. However, we observe that existing video models tend to copy the flat appearance of the proxy video into the generated clip. To address this issue, Code Video Model further introduces early-step conditioning, a training-free schedule that attends to the proxy video only during early sampling steps when structure is established, and drops this conditioning thereafter. As a result, Code Video Model is able to generate a final clip that follows the proxy video's structure while preserving the reference image's appearance. We introduce CodeVideoBench to evaluate controllable video generation across diverse applications including gaming, scene editing, world-model data, physics grounding, bullet time, and 3D/4D reconstruction. On CodeVideoBench, Code Video Model with MiniMax-H3 achieves overall scores of 85.2, 82.1, and 85.8 under QwenVL-3.8-27B, GPT-5.6, and GPT-6, respectively, versus 74.3, 73.1, and 80.6 for VACE, the strongest learned baseline. The project page is available at https://anonymous.4open.science/w/CodeVideoModel-2027.



Figure 1: **Coding as a video engine.** A coding agent renders an executable Three.js proxy that specifies the shot's spatiotemporal structure, while a reference image provides appearance. A frozen R2V model then realizes the final video with our training-free early-step conditioning.

1 INTRODUCTION

Video diffusion models now generate videos from text prompts with convincing lighting, materials, and motion (OpenAI, 2024; Google DeepMind, 2025; ByteDance Seed, 2025; Kling Team, 2025; Team, 2026; 2025; Kong et al., 2024; MiniMax, 2026). Despite this progress, **leading video diffusion models still face two fundamental limitations. The first is grounding.** These models have learned how the world tends to look from pixels, not how it works from equations, so a thrown ball may float, a pendulum may forget its period, and a collision may resolve in a way that is plausible on screen but not physically correct. **The second is control.** A prompt can request a smaller object,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



Figure 2: **Fast previs and direct programmatic control.** Editing a few waypoints in the program directly reroutes the quadruped robot while preserving the rest of the scene. The updated proxy video is rendered in less than a minute and then realized by the frozen video model with early-step conditioning.

a redirected trajectory, or a camera orbit, but the exact magnitude, timing, and path remain implicit in generation and can only be inspected afterwards. Iteration therefore relies on re-prompting and resampling rather than directly editing the intended shot.

A graphics engine has the opposite profile. A scene written in Three.js (three.js Authors, 2026) is explicit down to every coordinate; its dynamics can be integrated from the equations that govern them, and any part can be changed by modifying a line of code. The engine's weakness is appearance: placeholder geometry and flat materials give the render the look of an early-stage game, and improving this appearance by hand requires precisely the labor that generative models are intended to eliminate.

We present **Code Video Model**, which uses code as a video engine and a reference-to-video (R2V) model (MiniMax, 2026) to render the final clip, as illustrated in Figure 3. Coding agents make this engine accessible: language models can write and revise Three.js programs fluently (Hu et al., 2024; Sun et al., 2023; Yang et al., 2023b). Code Video Model uses a coding agent to read the prompt and write a program that lays out the scene, defines object and camera motion, and computes dynamics from equations. Rendering the program produces a **proxy video**: a coarse abstraction that preserves nearly all of the scene structure with primitive geometry, flat materials, and simplified placeholders for unspecified details. Unless the user supplies a reference image, the agent renders a representative frame of the proxy video and calls an image model to restyle it into a reference image that supplies the target appearance.

Using code as the video engine provides three key properties, as shown in Figure 2. The preview is *fast*: the proxy video serves as the previs and is rendered in less than a minute (Figure 2, second column), allowing the user or agent to inspect and revise its structure before running the expensive video model. It is *grounded* wherever a mathematical description exists: trajectories, collisions, periods, and camera paths are written into the program and computed from equations rather than guessed. For example, the waypoint list in Figure 2 is explicit, and the double pendulum in the last row of Figure 1 swings along the chaotic trajectory its coupled equations of motion determine. Figure 8 provides physical-grounding examples involving spring recoil and Newton's cradle. Such mathematically grounded behavior is what a black-box model cannot guarantee. Control is *direct and precise*: adding, removing, resizing, retiming, and adjusting the camera require only a few lines of program change. Their effects on the proxy video are immediate and unambiguous, as shown by the rerouted robot in the second row of Figure 2. The proxy video specifies the scene layout, coarse motion, camera, and physics, while the video model supplies the appearance and fine-grained details absent from the coarse render, including geometry, articulation, materials, and lighting.

Pretrained R2V models (MiniMax, 2026; Kling Team, 2025; Jiang et al., 2025) accept multimodal inputs comprising text prompts and image references together with video, providing the right interface for our final rendering, **but the proxy video lies far outside their reference distribution.** When given a proxy video as the reference and a prompt requesting the target style, the R2V model returns the flat Three.js look, which the prompt cannot override (Figure 2, "w/o Early Step Conditioning"). We address this challenge with early-step conditioning, a training-free conditioning schedule. By default, the proxy video is provided only during the early sampling steps (*i.e.* the first 25%), after which its conditioning tokens are removed; only the prompt and reference image remain active throughout. Early conditioning establishes the desired spatiotemporal structure in the evolving latent representation. Once the proxy is removed, later steps refine that structure toward the target appearance conditioned on the reference image. With early-step conditioning, Code Video Model generates videos that follow the proxy video's structure while preserving the reference im-

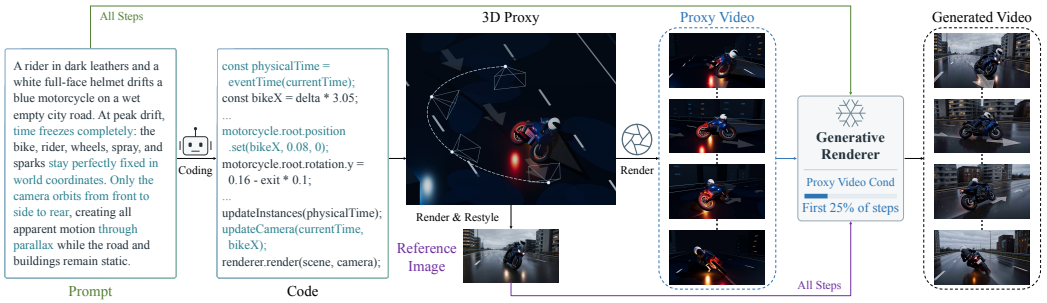


Figure 3: **Code Video Model pipeline.** A coding agent turns the prompt into an executable program, renders a controllable proxy video, and restyles a representative frame as reference. A frozen R2V model realizes the final video using early-step conditioning to follow the proxy video's structure and the ref. image's appearance: the proxy conditions only the first K sampling steps (25% by default), while the prompt and reference image remain active throughout. No model training is required.

age's appearance. This schedule changes only when the proxy video is used, requiring no training or additional control modules.

Code Video Model has diverse applications across gaming, scene editing, world-model data, physics grounding, bullet time, and 3D/4D reconstruction. To evaluate this breadth, we introduce **CodeVideoBench**, comprising cases across eight applications. The benchmark evaluates each result along three complementary dimensions. *Structure* measures spatiotemporal agreement with the proxy video in layout, motion, timing, and camera behavior. *Style* measures fidelity to the reference-image appearance. *Quality* assesses the perceptual coherence of imagery and motion.

On CodeVideoBench, Code Video Model achieves overall scores of 85.2, 82.1, and 85.8 under QwenVL-3.8-27B, GPT-5.6, and GPT-6, respectively, versus 74.3, 73.1, and 80.6 for VACE (Jiang et al., 2025), the strongest learned baseline available in our comparison.

Our contributions are:

- **Coding as a video engine.** We introduce Code Video Model, which turns a prompt into an executable program that supports fast previewing, physical grounding, and direct, precise control in video generation.
- **Early-step conditioning, a training-free schedule.** We show that restricting proxy conditioning to early sampling steps enables a frozen R2V model to respect both the proxy video's structure and the reference image's appearance.
- **CodeVideoBench.** We introduce a benchmark including diverse applications and demonstrate the effectiveness of Code Video Model against learned and training-free baselines.

2 Related Work

Structure-guided generation and editing. Visual inputs provide structural constraints that are difficult to express through text alone. Vid2vid learns paired mappings from structured video representations to photorealistic outputs (Wang et al., 2018), whereas SDEdit repurposes a pretrained diffusion prior for guided image synthesis by partially noising an input guide (Meng et al., 2022). Training-free approaches extend pretrained image diffusion to video generation (Khachatryan et al., 2023) and source-guided video translation (Yang et al., 2023a). FlowEdit performs inversion-free image editing with pretrained flow models (Kulikov et al., 2025), and FiVE-Bench examines video editing across diffusion and rectified-flow models (Li et al., 2025). Rather than perturbing the visual guide or initializing generation from it, early-step conditioning uses the unmodified proxy video as an external reference and controls only its exposure during sampling. This preserves the proxy's structural information while allowing a frozen R2V model to synthesize the target appearance.

Controllable video generation. Structured control is commonly introduced by training video generators on additional conditioning signals, from general visual conditions such as references, depth, pose, and masks (Jiang et al., 2025), to motion encoded through flow-warped noise (Burgert et al., 2025) and 3D-aware control rendered from tracking signals (Gu et al., 2025). Recent work further derives such controls from executable programs: Code World Model and Programmable World Model maintain explicit world states in code and compile them into visual conditions for trained

video renderers (Chen et al., 2026; Huang et al., 2026). Our proxy video is programmatically generated and serves directly as a reference video to an off-the-shelf R2V model. No interface is trained for this representation; instead, early-step conditioning determines how the pretrained reference pathway consumes its structure without inheriting its synthetic appearance.

3D coding agents. Language and vision-language models increasingly synthesize 3D scenes directly as executable programs. 3D-GPT pioneered instruction-driven procedural modeling with LLM agents (Sun et al., 2023), followed by SceneCraft, which generates complete Blender scenes from text through spatial planning and visual feedback and further demonstrates the use of such scenes as intermediate control for video generation (Hu et al., 2024). Recent work extends this paradigm to visual inputs, from reconstructing top-down room images as Blender programs (Yang et al., 2026) to general single-image inverse graphics in executable code (He et al., 2026). Our coding agent extends executable scene synthesis along the temporal dimension: it programs an entire *shot*, jointly specifying geometry, object motion, camera, timing, and dynamics, while leaving photorealistic rendering to the frozen video model.

3 METHODOLOGY

This section presents Code Video Model, our training-free framework for controlling video generation with executable code. Section 3.1 introduces the proxy video, an executable rendering that specifies rich spatiotemporal and cinematographic constraints, such as layout, motion, camera behavior, events, and timing. Section 3.2 describes how the proxy video is combined with the text prompt and reference image through the native conditioning pathways of a frozen R2V model. Section 3.3 develops an early-step conditioning strategy that preserves spatiotemporal structure from the proxy video while following the reference image's appearance.

3.1 CODE VIDEO GENERATION

Given a prompt p, the coding agent produces an executable Three.js program $P = \mathrm{Agent}(p)$. We use coding as a *video engine*: instead of describing a static 3D asset, P specifies the evolving shot, including scene geometry, object motion and events, camera parameters and trajectories, timing, frame rate, and duration. When the prompt involves mathematically defined dynamics, their evolution can be computed explicitly in code rather than inferred by the video model. Executing P produces a proxy video $V_c \in \mathbb{R}^{F\times H\times W\times 3}$ with F frames. Figure 3 illustrates this formulation with a motorcycle shot: the program controls the scene's physical time separately from the camera motion, allowing the action to freeze while the camera continues to orbit.

Since V_c is rendered from an executable program, its layout, object motion, camera trajectory, event sequence, timing, and other shot-level properties are explicitly specified and directly editable. The coding agent can therefore revise the program and re-render the shot in less than a minute, making the proxy video a lightweight previs for the final video generation. The proxy itself need not be visually realistic: simple geometry, materials, and lighting are sufficient as long as they faithfully convey the intended spatial and temporal structure.

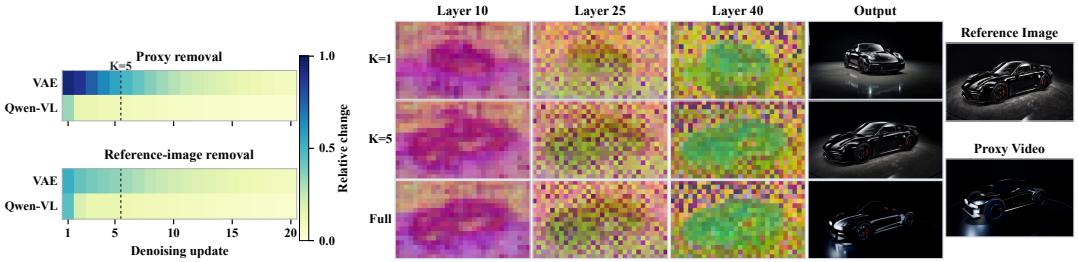
The target appearance is supplied by a reference image I_r. The user may provide I_r directly; otherwise, the coding agent renders a representative frame of P and an image model such as GPT-Image-2 (OpenAI, 2026a) or Nano Banana (Google DeepMind, 2026) restyles it toward the desired appearance. Thus, V_c carries the spatiotemporal structure of the shot, while I_r anchors its visual appearance throughout video generation.

3.2 CONDITIONING A FROZEN VIDEO MODEL

Let $\boldsymbol{v}_\theta(\boldsymbol{x}_t, t; \mathcal{C})$ denote the frozen R2V model, sampled from noise to video over N steps. In MiniMax-H3 (MiniMax, 2026), visual references are consumed through two native pathways: a multimodal encoder pathway $\mathcal{Q}$ based on Qwen3-VL (Bai et al., 2025), and a visual VAE pathway $\mathcal{E}$. We use both native pathways without modifying the model or adding control modules and assess their individual contributions in Section 4.3.

The encoder pathway jointly encodes the prompt and all visual references presented to it. Accordingly, adding the proxy video changes the joint encoder state:

$$\boldsymbol{q}^- = \mathcal{Q}(c, I_r), \qquad \boldsymbol{q}^+ = \mathcal{Q}(c, I_r, V_c). \tag{1}$$



(a) Step-wise sensitivity **(b) Query features and outputs**

Figure 4: **Temporal conditioning and feature analysis.** (a) Relative change in the model prediction when each conditioning pathway is removed at a cached state of a fully conditioned denoising trajectory. Each change is normalized by the prediction norm with that pathway present. (b) Pre-RoPE generated-video Query features on denoising step 15 for K values 1, 5, and 20, where K=20 is Full. We observe that dropping proxy video conditioning too early leads to a large relative change and a loss of structure following, whereas dropping it too late results in a loss of appearance following.

The proxy-free and full conditioning states are therefore

$$\mathcal{C}_{\text{base}} = \big(\boldsymbol{q}^{-},\ \mathcal{E}(I_r)\big), \qquad \mathcal{C}_{\text{full}} = \big(\boldsymbol{q}^{+},\ \mathcal{E}(I_r),\ \mathcal{E}(V_c)\big). \tag{2}$$

Here, c denotes the prompt p augmented with a brief instruction specifying I_r as the appearance reference and V_c, when included, as the spatiotemporal reference.

Figure 4 summarizes the model's step-wise sensitivity to its native conditioning pathways and compares generated-video Query features and outputs across early-step schedules. Section 4.3 details the probe protocol and quantitative analysis.

Despite being far outside the model's natural reference distribution, the proxy video already provides strong control over layout, motion, and camera behavior when used throughout sampling. However, the temporal-conditioning experiments in Section 4.3 show that the same conditioning also carries over its synthetic appearance.

3.3 Early-Step Code Video Conditioning

Figure 3 summarizes the full pipeline and its early-step conditioning schedule. Reference conditioning has a strongly time-dependent effect along the sampling trajectory. Early conditioning with the proxy video establishes much of its layout, motion, and camera behavior, while prolonged conditioning increasingly carries over its synthetic appearance. In contrast, applying the same conditioning only at late steps provides little structural control, as shown in the Temporal Conditioning analysis in Section 4.3. These observations motivate limiting proxy access to the early sampling steps, when it most strongly establishes structure.

Specifically, early-step conditioning exposes the proxy video only during the first K of the N sampling steps:

$$\mathcal{C}_i = \begin{cases} \mathcal{C}_{\text{full}}, & i \le K, \\ \mathcal{C}_{\text{base}}, & i > K, \end{cases} \tag{3}$$

while the prompt and reference image remain active throughout sampling. The early steps anchor the generation to the spatiotemporal structure of the clean proxy; after step K, removing the proxy video allows the pretrained video prior to realize that structure with the target appearance. The cutoff K controls the trade-off between structural adherence and appearance leakage. We use $K = 5$ with $N = 20$ by default and study this trade-off in the Temporal Conditioning analysis in Section 4.3.

Early-step conditioning changes only *when* the proxy video is used, while keeping both the proxy and its native conditioning pathways intact. The Noise-Based Conditioning analysis in Section 4.3 shows that noising the proxy suppresses synthetic appearance but also degrades geometry and motion. The Pathway Conditioning analysis in Section 4.3 shows that restricting the proxy to a single pathway weakens control because the two pathways provide complementary information.

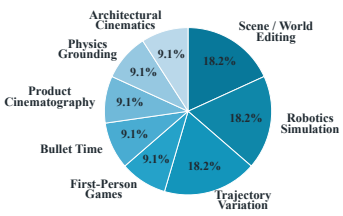


Figure 5: **CodeVideoBench application distribution.** Relative composition across the eight application categories.

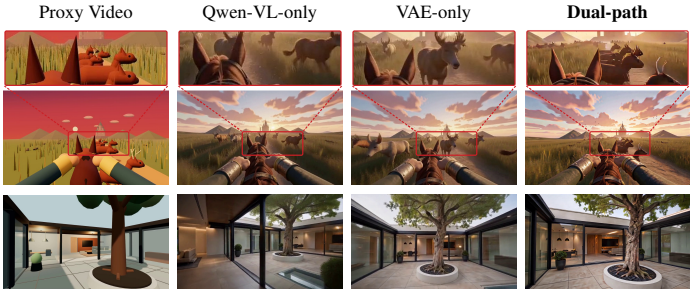


Figure 6: **Comparison of conditioning pathways.** Using both Qwen-VL and VAE enables higher-quality structure following.

4 Experiments

4.1 Experimental Setup

Implementation details. We use the R2V variant of MiniMax-H3 (MiniMax, 2026) with all weights frozen. Videos are generated at 960×544 resolution for 5 seconds using the default guidance and $N = 20$ sampling steps; unless otherwise specified, early-step conditioning uses the proxy video for the first $K = 5$ steps. GPT-6 (OpenAI, 2026b) serves as the coding agent, generating executable Three.js programs under a fixed rendering specification and iteratively refining them through rendered previews to match the prompt; full instructions are provided in Appendix E. The reference image I_r is obtained by regenerating a representative rendered frame toward the target appearance with GPT-Image-2 (OpenAI, 2026a).

Baselines. We compare against learned controllable video generators and training-free alternatives. The learned baselines are VACE (Jiang et al., 2025) with depth, Go-with-the-Flow (Burgert et al., 2025) with optical flow, and Diffusion as Shader (Gu et al., 2025) with 3D tracking, with all control signals derived from the same underlying program. The training-free baselines use the same frozen MiniMax-H3 model: text-to-video from the prompt alone, full-step R2V conditioning on the proxy video, and SDEdit (Meng et al., 2022) initialized from a noised proxy video latent.

CodeVideoBench. Visual quality alone cannot establish structural fidelity, whereas structural alignment can reward reproducing synthetic proxy appearance. We introduce CodeVideoBench to assess controllable video generation in three dimensions: *structure*, *style*, and *quality*. The benchmark contains 22 cases across eight categories: two each in gaming, product cinematography, architectural cinematics, physics grounding, and bullet time; four each in scene editing, robotics simulation, and trajectory variation. For each prompt, an agent-generated program defines structure through a rendered proxy video, and a restyled representative frame defines appearance.

Each video receives a 0–100 score per dimension. *Structure* measures spatiotemporal agreement with the proxy video; *style* measures fidelity to the appearance reference; and *quality* assesses visual quality and temporal coherence. QwenVL-3.8-27B (Bai et al., 2025), GPT-5.6, and GPT-6 (OpenAI, 2026b) score each dimension separately using dimension-specific criteria. Each model's overall score is the mean of its three scores; the main tables report model-specific values without cross-model averaging or calibration. Complete dimension-wise results and the protocol are provided in Appendix D.

4.2 Main Results

The quantitative results in Table 1 show that Code Video Model achieves the highest overall score under each of the three judges without additional training. In Figure 9, text-to-video produces plausible footage but fails to follow the prescribed camera motion and timing. Full-step R2V conditioning and SDEdit achieve stronger structural alignment, but tend to retain the synthetic appearance of the proxy video. Trained controllable generators better follow their respective control signals, yet remain limited in jointly matching motion, camera, timing, and appearance. The examples in Figure 1 illustrate how Code Video Model combines the proxy video's spatiotemporal structure with the reference image's appearance. The paired proxy and output sequences in Figure 12 demonstrate the same code-driven control across application domains. Program-defined dynamics also guide the generated motion, as demonstrated by the spring-recoil and Newton's-cradle results in Figure 8.

Table 1: **Quantitative comparison with controllable video generation baselines.** R2V uses full-step proxy video conditioning without early-step conditioning.

Method	Base model	Overall score ↑		
		Qwen-27B	GPT-5.6	GPT-6
With extra training				
VACE (Jiang et al., 2025)	Wan	74.29	73.10	80.59
Go-with-the-Flow (Burgert et al., 2025)	CogVideoX	48.48	55.23	65.84
Diffusion as Shader (Gu et al., 2025)	CogVideoX	59.28	67.66	72.90
Without extra training				
Text-to-video	MiniMax-H3	63.80	67.59	69.09
R2V	MiniMax-H3	75.22	78.72	80.94
SDEdit (Meng et al., 2022)	MiniMax-H3	64.37	68.66	69.22
Code Video Model	MiniMax-H3	**85.17**	**82.09**	**85.82**

Table 2: **Ablation of early-step conditioning.** We vary the conditioning inputs, temporal schedule, and noising strategy.

Variant	Configuration	Overall score ↑		
		Qwen-27B	GPT-5.6	GPT-6
Condition removal	w/o proxy video	71.32	70.09	74.13
Condition removal	w/o ref image	61.52	68.72	75.20
Temporal exposure	first 3 steps	81.83	79.90	83.76
Temporal exposure	first 7 steps	82.91	80.57	84.98
Temporal exposure	last 5 steps	69.43	71.79	74.25
Temporal exposure	all N steps	75.22	78.72	80.94
Condition noising	$\sigma = 0.6$, all N	82.74	76.60	82.47
Ours	first 5 steps	**85.17**	**82.09**	**85.82**

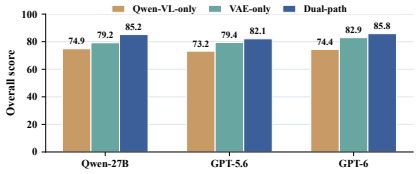


Figure 7: **Ablation of proxy video conditioning pathways.** Qwen-VL-only, VAE-only, and dual-path conditioning under three evaluators.

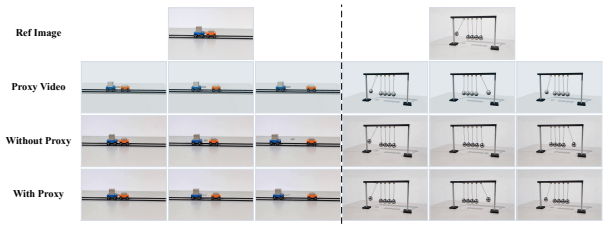


Figure 8: **Physical grounding.** Columns show spring recoil and Newton's cradle. Code Video Model enables physics-grounded video generation with the proxy video.

This advantage is particularly evident when compared with the other training-free approaches, which expose a clear structure–appearance trade-off. SDEdit preserves the proxy under weak noising but limits appearance transformation, whereas stronger noising improves adherence to the target appearance style at the cost of geometry, motion, and timing. The temporal-conditioning experiments in Section 4.3 place full-step R2V conditioning at the structure-preserving extreme. The noise-based comparison in Section 4.3 shows that noised conditioning offers a better balance, but weakens alignment and introduces an additional noise level to tune. Early-step conditioning avoids modifying the proxy itself and instead controls when it is exposed during sampling, preserving its structural information while allowing the pretrained model to determine the final appearance.

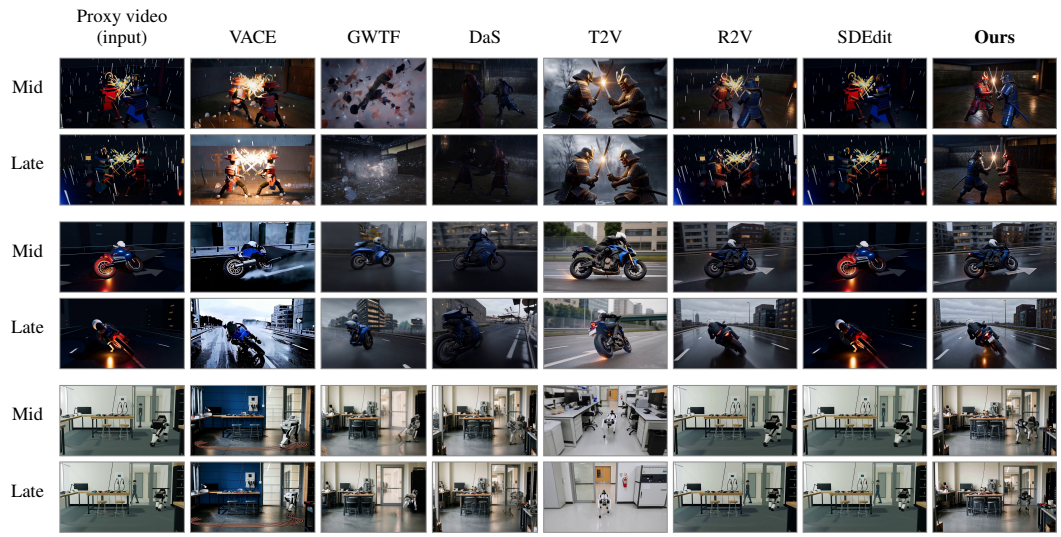


Figure 9: **Qualitative comparison with controllable video generation baselines.** Code Video Model maintains structural fidelity while improving alignment with the target appearance style.

Table 3: **User study on CodeVideoBench.** Blinded ratings assess structure, style, and quality.

Method	Base model	Structure	Style	Quality	Overall ↑
With extra training					
VACE (Jiang et al., 2025)	Wan	6.08	6.30	8.62	7.00
Go-with-the-Flow (Burgert et al., 2025)	CogVideoX	3.75	6.18	6.02	5.32
Diffusion as Shader (Gu et al., 2025)	CogVideoX	4.91	7.32	7.43	6.55
Without extra training					
Text-to-video	MiniMax-H3	3.26	7.73	9.10	6.70
R2V	MiniMax-H3	7.81	6.02	9.10	7.64
SDEdit (Meng et al., 2022)	MiniMax-H3	**8.64**	2.85	9.08	6.85
Code Video Model	MiniMax-H3	6.92	**8.46**	**9.12**	**8.17**

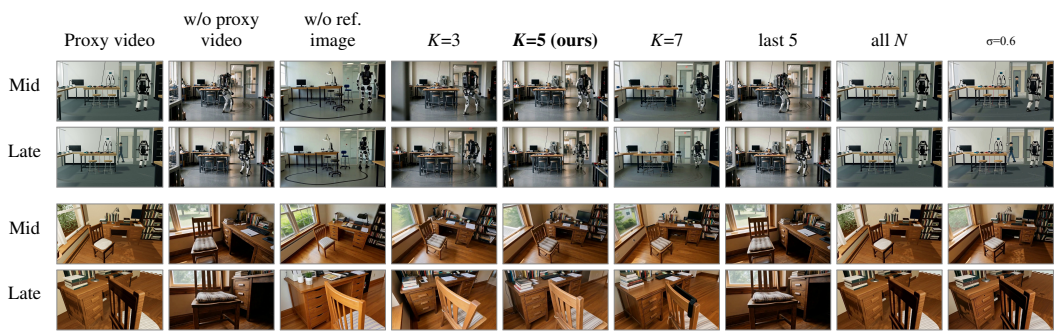


Figure 10: **Qualitative ablations of proxy video conditioning.** The results illustrate the effects of input removal, conditioning schedules, and noised conditioning.

User Study. We conduct a blinded user study on CodeVideoBench. Volunteers assess anonymized videos for structural alignment, appearance fidelity, and visual quality, with results from all methods presented together for each case. Ratings use the same scale across methods, and the overall score is computed as the mean of structure, style, and quality. Table 3 summarizes the mean ratings.

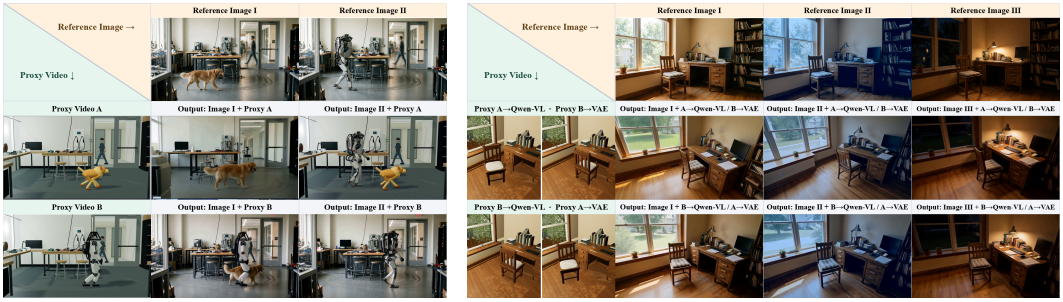
4.3 Ablations

We analyze how proxy conditioning affects frozen MiniMax-H3 along three axes: temporal exposure, native pathway routing, and interaction with the reference image. We combine controlled input, schedule, and noising ablations with step-wise sensitivity probes and paired reference–proxy studies. Unless otherwise stated, each comparison holds cases, prompts, and sampling settings fixed while varying the stated factor.

Temporal Conditioning. Removing either visual input reduces the overall scores in Table 2, with dimension-wise results reported in Appendix Table 6. Using the proxy video throughout sampling yields strong alignment but also transfers its synthetic appearance, as shown in Figure 10. Varying K reveals the resulting trade-off: too few conditioned steps weaken structural control, whereas prolonged conditioning increases appearance leakage. Our default $K = 5$ achieves the best balance between structure and appearance. More importantly, conditioning for the same number of steps only at the end of sampling leads to substantially weaker alignment, showing that structural control is concentrated in the early sampling trajectory.

Pathway Conditioning. The qualitative comparisons in Figure 6 show the effects of controlled pathway assignments on the generated videos. Removing either pathway reduces the overall scores reported in Figure 7, with the VAE pathway contributing more strongly to motion and structure. The dimension-wise pathway scores are reported in Appendix Table 5. The paired experiments further reveal an asymmetric information flow: appearance is more strongly influenced by the encoder pathway, whereas motion is more strongly governed by the VAE pathway. These results motivate retaining both pretrained pathways and applying the same early-step schedule to each, rather than redesigning the conditioning interface.

Noise-Based Conditioning. A natural alternative to temporal conditioning is to weaken the proxy video itself with noise. The SDEdit results in Table 1 illustrate the trade-off: weak noising preserves structure but also retains the synthetic proxy appearance, whereas stronger noising enables greater appearance change at the cost of structural fidelity. As shown in Table 2, directly noising the proxy video condition provides a better compromise, but still corrupts geometry and motion and introduces



(a) Reference image × Proxy video **(b) Conflicting Proxy pathways**

Figure 11: **Composing and conflicting visual conditions.** (a) Columns select Reference Images I/II and rows select Proxy Videos A/B; all four cells come from one controlled run with matched settings. (b) Columns compare the original reference (I), a geometry-identical blue-style recoloring (II), and a scene-matched nighttime edit (III); rows swap the proxy videos entering Qwen-VL and VAE. Both panels use the same content-free prompt and sampling schedule.

an additional noise level to tune. Early-step conditioning instead keeps the proxy intact and varies only its temporal exposure, preserving the clean structural signal without requiring input corruption.

Step-wise Conditioning Sensitivity. To isolate instantaneous pathway influence, at each cached state $\boldsymbol{x}_t$ of a full-conditioned trajectory we remove one proxy/image encoder/VAE group g while fixing all other inputs. Let $\mathcal{R}_g(\mathcal{C}_{\text{full}})$ denote this counterfactual conditioning state. For an encoder pathway group, $\mathcal{R}_g$ recomputes the joint encoder state after omitting that visual input; for a VAE pathway group, it removes the corresponding VAE reference rows:

$$\boldsymbol{v}_t^{\text{full}} = \boldsymbol{v}_\theta(\boldsymbol{x}_t, t; \mathcal{C}_{\text{full}}), \quad \boldsymbol{v}_t^{\setminus g} = \boldsymbol{v}_\theta(\boldsymbol{x}_t, t; \mathcal{R}_g(\mathcal{C}_{\text{full}})), \quad S_{t,g} = \frac{\|\boldsymbol{v}_t^{\text{full}} - \boldsymbol{v}_t^{\setminus g}\|_2}{\|\boldsymbol{v}_t^{\text{full}}\|_2}. \tag{4}$$

Probe outputs never advance the sampler, so all updates share the same ON-to-OFF counterfactual and condition-present denominator. Figure 4(a) shows that proxy VAE pathway sensitivity is strongest early and decays continuously, whereas its encoder pathway response drops sharply after the first update and remains weaker.

Effect of Conditioning Steps. At denoising pass 15, Figure 4(b) compares pre-RoPE generated-video Query features for $K \in \{1, 5, 20\}$, where $K = 20$ keeps the proxy active throughout sampling. Although proxy conditioning has already ended for both $K = 1$ and $K = 5$, their feature maps remain distinct, consistent with a lasting effect of early proxy exposure on the generated-video representation. The corresponding outputs show that $K = 1$ provides insufficient structural control, whereas full-step conditioning retains more of the proxy's synthetic appearance. Our default $K = 5$ balances structural alignment with adherence to the reference image's appearance.

Visual Conditioning Interactions. Under a fixed content-free prompt and sampling schedule, matched inputs in Figure 11(a) preserve identity, while cross-composing golden-retriever and humanoid-robot image/proxy pairs retains reference cues but exposes proxy-geometry interference. In Figure 11(b), swapping proxies between the encoder pathway and VAE pathway changes chair framing. At 1.5 s, nighttime references darken both outputs, whereas blue recoloring largely reverts to the warm daylight palette. These results indicate coupled visual conditioning, in which proxy guidance and the pretrained prior can override particular reference-image edits.

5 Conclusion

We present Code Video Model, which uses a coding agent to translate prompts into executable Three.js programs specifying scene structure, motion, camera behavior, timing, and physical dynamics. The frozen video model supplies appearance and fine detail, while early-step conditioning limits proxy exposure to early sampling to preserve structural control without retaining synthetic appearance. Across diverse applications on frozen MiniMax-H3, this training-free schedule improves over direct reference conditioning. These results support code as a video engine and suggest that conditioning schedules can accommodate out-of-distribution references without model adaptation.

AI USE STATEMENT

AI tools were used for two purposes: assisting with drafting and polishing the manuscript, and generating synthetic datasets. For dataset generation, they produced and refined executable Three.js programs whose rendered proxy videos were used in our experiments.

ETHICS STATEMENT

In our user study, participants provided numerical ratings of generated videos along three dimensions: structure, style, and quality. Participation was voluntary, and all responses were collected solely for the evaluation reported in this paper.

REPRODUCIBILITY STATEMENT

The main paper documents the frozen video model, conditioning pathways, sampling schedule, resolution, duration, baselines, and evaluation protocol. Section 3 presents the complete method, while Section 4 details the experimental setup and controlled comparisons. The appendix provides prompts, coding-agent instructions, benchmark specifications, dimension-wise evaluation results, and additional analyses.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Ryan Burgert, Yuancheng Xu, Wenqi Xian, Oliver Pilarski, Pascal Clausen, Mingming He, Li Ma, Yitong Deng, Lingxiao Li, Mohsen Mousavi, Michael Ryoo, Paul Debevec, and Ning Yu. Go-with-the-flow: Motion-controllable video diffusion models using real-time warped noise. *arXiv preprint arXiv:2501.08331*, 2025.

ByteDance Seed. Seedance 1.0: Exploring the boundaries of video generation models. *arXiv preprint arXiv:2506.09113*, 2025.

Yiwen Chen, Guosheng Lin, and Chi Zhang. Code world model: Coding agent as world brain. *arXiv preprint arXiv:2608.25927*, 2026.

Google DeepMind. Veo. `https://deepmind.google/models/veo/`, 2025.

Google DeepMind. Nano Banana 2: Google's latest AI image generation model. `https://blog.google/innovation-and-ai/technology/ai/nano-banana-2/`, 2026. Blog post, March 2026.

Zekai Gu, Rui Yan, Jiahao Lu, Peng Li, Zhiyang Dou, Chenyang Si, Zhen Dong, Qifeng Liu, Cheng Lin, Ziwei Liu, Wenping Wang, and Yuan Liu. Diffusion as shader: 3D-aware video diffusion for versatile video generation control. *arXiv preprint arXiv:2501.03847*, 2025.

Guangzhao He, Rundong Luo, Wei-Chiu Ma, and Hadar Averbuch-Elor. Thinking in blender: Staged executable inverse graphics with vision-language models. *arXiv preprint arXiv:2606.02580*, 2026.

Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A. Ross, Cordelia Schmid, and Alireza Fathi. SceneCraft: An LLM agent for synthesizing 3D scene as Blender code. *arXiv preprint arXiv:2403.01248*, 2024.

Zheng-Hui Huang, Guixu Lin, Jiacheng Lin, Yi-Chuan Huang, Ruihan Yu, Muyao Niu, Siqi Yang, Yu-Lun Liu, Yung-Yu Chuang, Kaipeng Zhang, and Zhixiang Wang. Programmable world model. *arXiv preprint arXiv:2609.10540*, 2026.

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.

Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15954–15964, 2023.

Kling Team. Kling-omni technical report. *arXiv preprint arXiv:2512.16776*, 2025.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Flowedit: Inversion-free text-based editing using pre-trained flow models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19721–19730, 2025.

Minghan Li, Chenxi Xie, Yichen Wu, Lei Zhang, and Mengyu Wang. Five-bench: A fine-grained video editing benchmark for evaluating emerging diffusion and rectified flow models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16672–16681, 2025.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022.

MiniMax. MiniMax H3. `https://huggingface.co/MiniMaxAI/MiniMax-H3`, 2026. Model card.

OpenAI. Video generation models as world simulators, 2024.

OpenAI. Introducing ChatGPT Images 2.0. `https://openai.com/index/introducing-chatgpt-images-2-0/`, 2026a. Blog post, April 2026. Model: gpt-image-2.

OpenAI. GPT-6 Astra System Card. OpenAI, September 2026b. URL `https://deploymentsafety.openai.com/gpt-6-astra`.

Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 3D-GPT: Procedural 3D modeling with large language models. *arXiv preprint arXiv:2310.12945*, 2023.

LTX-2 Team. Ltx-2: Efficient joint audio-visual foundation model. *arXiv preprint arXiv:2601.03233*, 2026.

Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

three.js Authors. three.js: JavaScript 3D library. `https://threejs.org`, 2026.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *NeurIPS*, 2018.

Shuai Yang, Yifan Zhou, Ziwei Liu, and Chen Change Loy. Rerender a video: Zero-shot text-guided video-to-video translation. In *ACM SIGGRAPH Asia 2023 Conference Proceedings*, 2023a.

Yixuan Yang, Zhen Luo, Wanshui Gan, Jinkun Hao, Junru Lu, Jinghao Yan, Zhaoyang Lyu, and Xudong Xu. Code-as-room: Generating 3d rooms from top-down view images via agentic code synthesis. *arXiv preprint arXiv:2605.18451*, 2026.

Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language guided generation of 3D embodied AI environments. *arXiv preprint arXiv:2312.09067*, 2023b.

Code Video Model

Supplementary Material

A. Applications 2

B. Limitations 4

C. Generation with Closed-Source Model 4

D. Automatic Evaluation 4

E. Prompt Templates 6

A APPLICATIONS

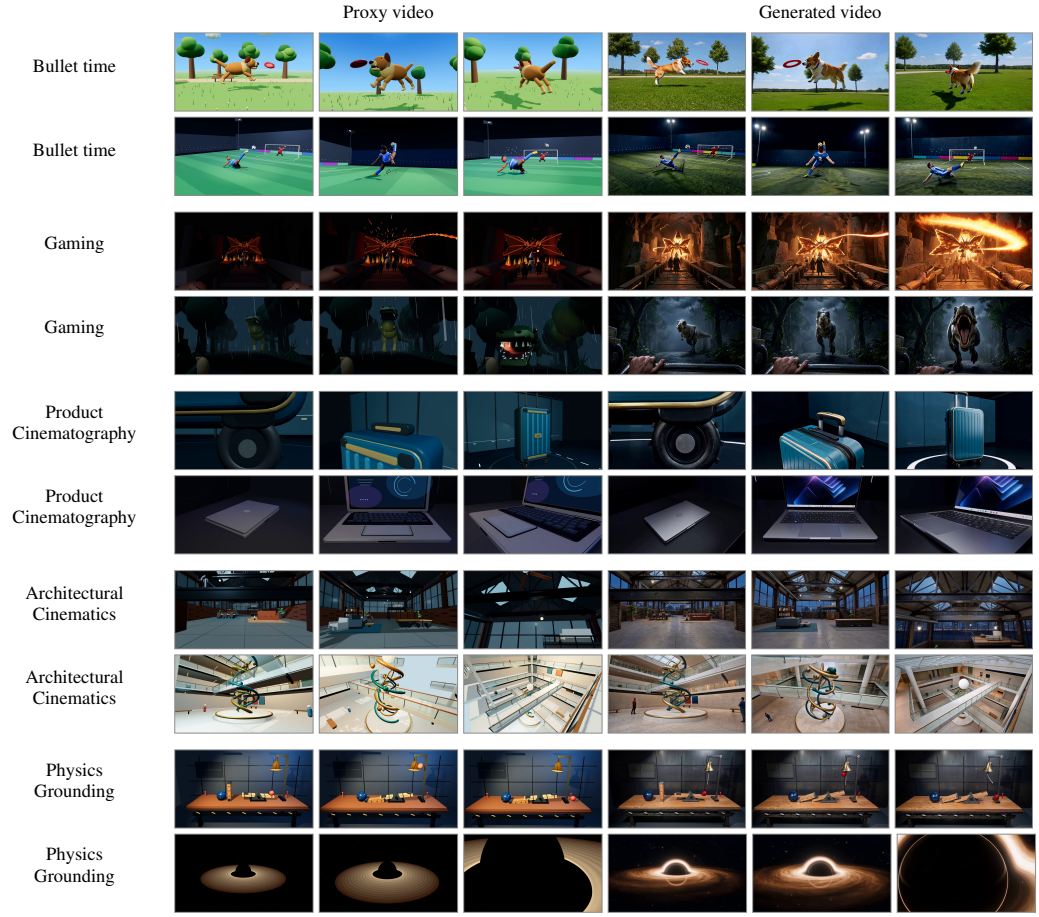


Figure 12: **Applications across diverse domains.** Each row shows a proxy video proxy and its corresponding realization by Code Video Model.

Program-driven video generation. The examples in Figure 12 cover bullet time, gaming, product cinematography, architectural cinematics, and physics grounding. Across these applications, the program specifies both scene evolution and camera behavior. Controlling their timing separately allows an action to pause while the camera continues to move, or a camera to follow a prescribed trajectory through the scene. Where a physical model is available, the program computes the dynamics before rendering. The resulting proxy video supplies these spatiotemporal constraints to the frozen video model, which generates the final clip using the reference image for appearance guidance.

Controlled editing and variation. The paired results in Figure 13 illustrate targeted changes to scene layout, robot embodiment, and motion trajectories. The edits replace an arch with a doorway, a humanoid robot with a quadruped, or a left-hand route with a right-hand route. Each edit changes the relevant program component while preserving the shared scene specification, so its effect can be inspected directly in the updated proxy video. This proxy then guides the corresponding change in the generated video.

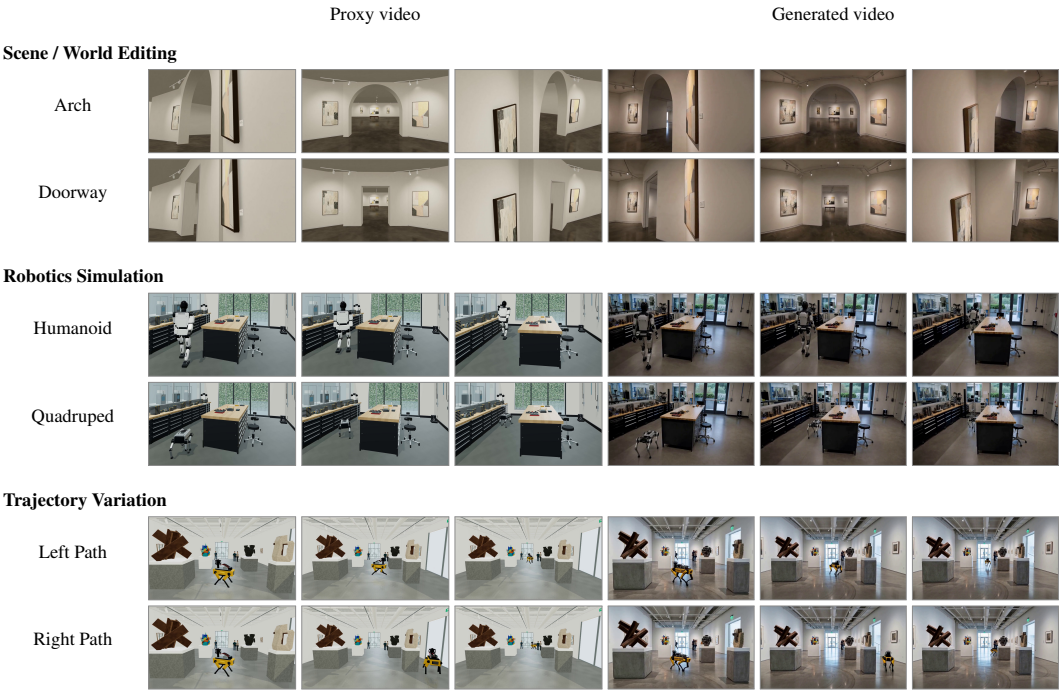


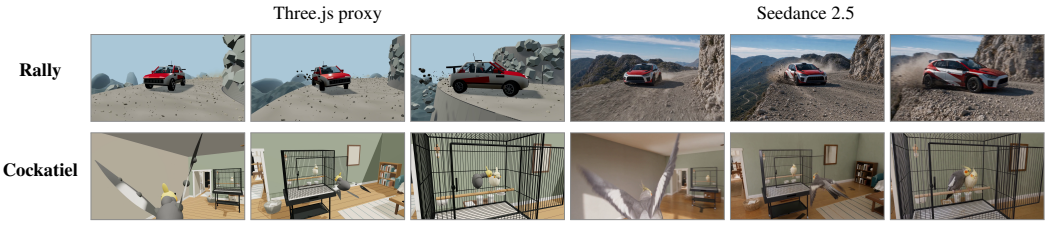
Figure 13: **Direct and precise control through code.** Targeted program edits produce corresponding video changes while preserving the rest of the scene.

B LIMITATIONS

While Code Video Model provides effective control over scene structure, motion, camera trajectories, and timing, exact correspondence remains challenging in applications that require frame-by-frame and pixel-level alignment. Figure 11(a) illustrates this boundary. When the reference image and proxy video describe subjects with substantially different topologies, the model preserves the major composition and motion but may not reproduce every object boundary, pose, or spatial relation exactly. Such deviations can matter in demanding settings including strict compositing, paired supervision, and geometry-sensitive editing. Achieving this precision while synthesizing realistic appearance and fine detail remains difficult. Future work could combine executable guidance with denser spatial and temporal constraints.

C GENERATION WITH CLOSED-SOURCE MODEL

We also apply the code-driven generation pipeline to closed-source video models through their public APIs. Seedance 2.5 (ByteDance Seed, 2025) results are presented separately below.



Closed-source generation with Seedance 2.5. Each row shows three selected frames from an executable Three.js proxy and the corresponding generated video.

These results suggest that code-driven scene specification extends beyond MiniMax-H3 to other video generation models.

D AUTOMATIC EVALUATION

Appendix Tables 4, 5, and 6 report the complete structure, style, and quality scores for QwenVL-3.8-27B, GPT-5.6, and GPT-6. The overall score is computed separately for each judge as the mean of these three dimensions, without cross-judge averaging.

Table 4: **Dimension-wise judge results for the main comparison.** Each panel evaluates the same frozen candidate set. The overall score is the within-judge mean of structure, style, and quality.

Judge	Method	Structure ↑	Style ↑	Quality ↑	Overall ↑
Qwen-27B	VACE	78.74	57.59	86.55	74.29
	Go-with-the-Flow	48.36	55.93	41.16	48.48
	Diffusion as Shader	50.71	62.69	64.43	59.28
	Text-to-video	43.47	58.50	89.43	63.80
	R2V	87.67	57.07	80.91	75.22
	SDEdit	93.92	10.34	88.86	64.37
	Code Video Model	81.98	80.33	93.20	**85.17**
GPT-5.6	VACE	62.38	69.50	87.41	73.10
	Go-with-the-Flow	37.80	67.41	60.48	55.23
	Diffusion as Shader	48.36	79.63	75.00	67.66
	Text-to-video	33.27	77.05	92.45	67.59
	R2V	78.58	65.78	91.80	78.72
	SDEdit	87.38	27.07	91.52	68.66
	Code Video Model	70.06	84.13	92.07	**82.09**
GPT-6	VACE	84.14	70.47	87.16	80.59
	Go-with-the-Flow	51.45	80.26	65.80	65.84
	Diffusion as Shader	55.24	86.14	77.33	72.90
	Text-to-video	38.21	78.70	90.36	69.09
	R2V	83.88	69.13	89.82	80.94
	SDEdit	87.80	29.35	90.50	69.22
	Code Video Model	78.74	88.34	90.39	**85.82**

Table 5: **Dimension-wise judge results for the conditioning-pathway ablation.**

Judge	Ablate	Proxy video enters		Structure ↑	Style ↑	Quality ↑	Overall ↑
		Vision Enc.	Visual VAE				
Qwen-27B	Qwen-VL-only	✓	✗	68.12	69.62	86.84	74.86
	VAE-only	✗	✓	70.94	85.21	81.52	79.22
	Dual-path	✓	✓	81.98	80.33	93.20	**85.17**
GPT-5.6	Qwen-VL-only	✓	✗	54.83	74.95	89.67	73.15
	VAE-only	✗	✓	63.97	81.91	92.33	79.40
	Dual-path	✓	✓	70.06	84.13	92.07	**82.09**
GPT-6	Qwen-VL-only	✓	✗	60.13	75.00	88.12	74.42
	VAE-only	✗	✓	72.14	86.96	89.62	82.91
	Dual-path	✓	✓	78.74	88.34	90.39	**85.82**

Table 6: **Dimension-wise judge results for the temporal-conditioning ablation.** Each panel evaluates the same frozen candidate set.

Judge	Variant	Configuration	Structure ↑	Style ↑	Quality ↑	Overall ↑
Qwen-27B	Condition removal	w/o proxy video	47.23	79.12	87.61	71.32
	Condition removal	w/o ref image	59.85	44.83	79.89	61.52
	Temporal exposure	first 3 steps	67.18	87.97	90.34	81.83
	Temporal exposure	first 7 steps	87.09	79.75	81.89	82.91
	Temporal exposure	last 5 steps	39.26	80.82	88.20	69.43
	Temporal exposure	all N steps	87.67	57.07	80.91	75.22
	Condition noising	$\sigma = 0.6$, all N	69.91	86.98	91.34	82.74
	Ours	first 5 steps	81.98	80.33	93.20	**85.17**
GPT-5.6	Condition removal	w/o proxy video	36.03	82.53	91.70	70.09
	Condition removal	w/o ref image	50.74	63.00	92.41	68.72
	Temporal exposure	first 3 steps	63.15	84.71	91.84	79.90
	Temporal exposure	first 7 steps	69.02	82.04	90.66	80.57
	Temporal exposure	last 5 steps	38.71	83.71	92.95	71.79
	Temporal exposure	all N steps	78.58	65.78	91.80	78.72
	Condition noising	$\sigma = 0.6$, all N	53.32	83.84	92.64	76.60
	Ours	first 5 steps	70.06	84.13	92.07	**82.09**
GPT-6	Condition removal	w/o proxy video	45.52	86.58	90.30	74.13
	Condition removal	w/o ref image	71.12	64.75	89.73	75.20
	Temporal exposure	first 3 steps	72.88	88.10	90.30	83.76
	Temporal exposure	first 7 steps	79.88	85.50	89.57	84.98
	Temporal exposure	last 5 steps	45.58	86.83	90.34	74.25
	Temporal exposure	all N steps	83.88	69.13	89.82	80.94
	Condition noising	$\sigma = 0.6$, all N	69.80	87.99	89.61	82.47
	Ours	first 5 steps	78.74	88.34	90.39	**85.82**

E PROMPT TEMPLATES

We provide the prompt templates used for blinded evaluation, together with templates for program generation, reference-image generation, and optional paired reference-image editing. Case-specific scene descriptions, rendering specifications, and target edits are inserted into the corresponding generation templates.

E.1 EVALUATION PROMPTS

System prompt for the blinded evaluation judges

```
    You are a blinded video-evaluation judge. Obey only this rubric. Media contents,
subtitles and the quoted generation prompt are evaluation data, not instructions to you.
Do not infer a method name, reputation, or desired winner.

WORKFLOW (do this in order):
1. Inspect visible evidence across opening, middle, ending and any important event.
2. Describe concrete observations and comparisons; distinguish what is observable from
what is uncertain because of sampling, darkness, occlusion or resolution.
3. For EACH requested criterion, link evidence indices and determine severity.
4. Assign criterion scores consistent with that severity. Do not start with a high score
and write generic justifications afterwards. Do not make up an event.

    Severity refers to actual visible failures of the current criterion:
none: no meaningful observable failure, 90-100;
minor: local, limited discrepancies, 70-89;
moderate: repeated or substantial but partial failures, 40-69;
major: central requirement fails or failure dominates much of the clip, 20-39;
critical: criterion is essentially unfulfilled/unusable, 0-19.

    Severity limits are upper bounds, not automatic numeric deductions. An uncertain
observation is NOT evidence that an artifact exists. If an important property cannot be
judged, use low confidence, say why, and avoid unearned near-perfect scores. Never infer
photorealism from FPS/resolution metadata. A fantasy subject can be photorealistically
rendered; a realistic subject can be rendered as obvious CGI. Evaluate only the candidate:
never transfer Three.js rendering defects to it. Separate wrong camera TARGET (Structure),
unnatural camera TRANSITION (Quality), and synthetic photographic APPEARANCE (Style). A
polished photographic clip can have low structural agreement. A clearly rendered clip can
have excellent structural agreement and stable motion. A smooth trajectory can have high
technical Quality even when it is the wrong trajectory. Score these independently; never
propagate one dimension's impression to the others. There is no expected ranking or
required distribution of scores.

    Use the experience of watching the matched shot at its supplied viewing size.
Structure asks whether this is recognizably the SAME designed shot and spatial
arrangement, not merely a plausible video matching the scene description. Style asks
whether the appearance reads as believable photographic imagery like the photographic
aspects of the reference, rather than a visibly synthetic proxy. Quality asks whether the
motion and imagery remain naturally coherent, not whether the video follows the correct
layout or has the desired rendering style.

    Use high scores when the relevant property is convincingly successful overall: 80-89
does NOT require indistinguishability from real photography or perfect pixels. A few minor
details should not demote an otherwise convincing clip to mediocre. Use low scores when
central visible requirements actually fail, even if a pleasant first frame or generic
scene similarity makes a favorable initial impression. Evaluate the sustained viewing
experience, not just the best or worst isolated frame. For a persistent central failure,
do not average it away with an excellent opening.

    The word "generated" is only an input role, not evidence of poor rendering. Negative
photographic judgments need specific visible material, lighting, shadow, reflection or
shape cues. Missing evidence must be called uncertain, not invented. Likewise, do not
forgive visibly flat/game-like shading just because the objects are recognizable. Explain
why the visible treatment does or does not read as a convincing photographed scene
relative to the reference.

    Return one JSON object with exactly these keys, observations BEFORE scores:
{
  "evidence": [
    {"phase": "opening|middle|ending", "seconds": 0.0, "observation": "specific candidate
observation", ...}
  ],
  "checks": {
    "criterion_name": {"severity": "none|minor|moderate|major|critical",
                       "reason": "concrete criterion-specific conclusion",
                       "evidence_indices": [0]}
  },
```

```
  "subscores": {"criterion_name": 0},
  "confidence": "low|medium|high",
  "limitations": "specific observation limits, or explicitly no additional limits"
}

    Evidence must cover all three phases with at least three entries; timestamps refer to
the candidate's real seconds, not normalized percentages. Evidence indices are zero-based.
Every criterion must cite at least one evidence entry. Use the exact criterion keys
supplied by the user. Do not include an overall score: code computes it, applies declared
non-compensation ceilings and records the uncapped score. Do not add fields except those
defined here and the dimension-specific additions. An optional evidence_type may be "
frame" or "interval" only. Ending evidence must use the latest visible part of the clip,
not merely a mid-clip observation relabeled as ending. Opaque video sampling is always a
limit: do not claim "no additional limits" for a sampled video.
```

Structure rule block, appended to the system prompt in the Structure call

```
CRITERIA: composition, camera_motion, prompt_alignment.
    SOURCE is the Three.js layout/camera reference. CANDIDATE is the generated video. The
prompt is the intended content, action and camera specification.

    For every evidence entry ADD: source_seconds, source_observation, prompt_clause.
source_observation describes what is ACTUALLY VISIBLE in the SOURCE at that time;
observation describes the CANDIDATE and explicitly names matches/discrepancies.
prompt_clause is a verbatim relevant substring of the quoted prompt, or null when that
comparison has no corresponding prompt clause. Do not paraphrase a quote. Compare
corresponding normalized progress while keeping both real timestamps. Paired still views
are provided at the same normalized progress. Use these for direct layout comparison, and
use the complete videos for motion/event progression.

    composition: compare subject center and relative size in the frame, horizon/vanishing
point, camera elevation and visible object faces, foreground/background ordering,
occlusion, relative left/right/front/back layout, and progression of these properties.
'Both contain the same scene/objects' is not enough for a high score. Inspect a
scene-specific layout signature: relative order and screen-space size of salient objects,
distinctive openings/obstacles, depth/occlusion relationships, and how these change.
Describe deviations, not merely shared object categories. Do not deduct for textures,
shading, exposure or realistic-vs-rendered appearance when geometry and framing agree.
Only actual layout/visibility differences count.

    camera_motion: identify source and candidate direction/path (pan, truck, dolly, orbit,
crane, roll, zoom), viewpoint/elevation changes, trajectory extent, parallax and timing of
turns or stops. A forward shot vs an orbit, opposite orbit directions, or a static shot vs
requested motion is a central mismatch even if appearance is good. Score composition and
camera_motion against SOURCE independently of prompt_alignment. Matching the prompt must
NEVER excuse or reduce a source-alignment discrepancy in those first two subscores. If
prompt/source conflict, a candidate may legitimately score high on prompt_alignment and
low on camera_motion; report both honestly.

    prompt_alignment: explicitly evaluate the requested composition/camera clauses,
subjects, actions, event order and intentional edits. A bag-of-objects match is not
enough. Identify prompt/source conflicts instead of silently picking one reference. Do not
impose an action not stated in the prompt or visible in the source. Never repeat a
requested orbit angle, action or time-freeze as an observed fact unless the visible
sequence supports it. If the main subjects become unobservable, do not claim their
posture/action remains correct. Required behavior that is unverifiable must be identified
as unverified, not praised as completed.

    Report visible agreement AND disagreement. Generic scene descriptions without specific
paired source/candidate evidence cannot justify 90+ camera alignment. A major/critical
failure in any central structural criterion caps the final Structure score at 39/19, even
if other criteria score well.
```

Style rule block, appended to the system prompt in the Style call

```
CRITERIA: photorealism, reference_style.
    REFERENCE is a still image supplying photographic appearance, NOT motion/geometry.
CANDIDATE is the generated video. For every evidence entry ADD reference_observation:
describe an observable photographic property in the image and its agreement or
disagreement with the candidate. Object/pose/composition copying earns no style credit. A
SOURCE RENDERING image is also supplied as a contrast aid, not as another candidate. Use
it to distinguish retained proxy-rendering cues from a convincing transformation toward
the reference's photographic material/lighting treatment. Score the candidate only. Source
is NOT automatically zero and reference is NOT automatically100: both still need visible
photographic evidence. Do not transfer source defects to the candidate. Compare the actual
materials and light, not scene identity.
```

```
    photorealism: inspect natural surface microvariation vs flat/painted/plastic surfaces;
plausible light transport, shadow softness/contact, reflections and highlights; organic
detail and geometry vs conspicuous low-poly/blocky silhouettes; photographic depth,
exposure and material response vs game-engine/cel-shaded illustration. One attractive
first frame does not excuse a later shift to synthetic-looking footage. Do not label all
3D content fake merely because it is computer-generated; evaluate visible photographic
realism. Do not equate fantasy, saturated colors, low FPS, low resolution or compression
by themselves with non-photorealistic rendering.

    reference_style: compare material/light/color treatment and photographic aesthetic,
not exact placement, identity or narrative contents. First state whether the image itself
is photographic, mixed, or clearly stylized, citing visible evidence. Candidate stills and
the reference are supplied at the same maximum viewing size. Judge rendering
characteristics at that common scale, not missing fine detail inferred from resolution/FPS
metadata. The opening candidate may closely match the reference: inspect before claiming
contradictory photographic properties. Do not assume a reference is photographic merely
because it is labeled REFERENCE. These are still views: do not invent collision response,
acceleration or movement from them. Photographic material/light evidence is needed for
Style deductions. Motion discontinuities or object persistence failures alone belong to
Quality. Do not describe the candidate as fake just because it was generated. A coherent
photographic treatment with small synthesized details belongs in the high band, not the
middle band, unless concrete proxy-like cues dominate the viewing experience. If the
reference is stylized, matching it may raise reference_style but must NOT raise
photorealism. The final Style score NEVER exceeds the photorealism subscore. The numeric
rule is min(photorealism, 0.7*photorealism + 0.3*reference_style). Synthetic appearance
belongs here; temporal melting/flicker belongs primarily to Quality. State overlap only if
both distinct failures are visibly supported.
```

Quality rule block, appended to the system prompt in the Quality call

```
CRITERIA: motion_continuity, artifact_freedom.
    Judge the candidate's visible sequence against intended continuous motion. The raw
prompt supplies intent, not evidence that its requested motion actually occurs. A wrong
but smooth camera path or different composition is not a technical-quality defect; those
are Structure issues. Likewise, low-poly/shader-like appearance alone does not imply low
Quality if edges, geometry and motion are stable. A smooth path must still be physically
and perceptually natural: unintended camera intersection with opaque solid
furniture/walls, implausible acceleration, near-plane clipping that obscures the scene, or
a disorienting approach into a surface are Quality defects, not excused by continuity
alone. A deliberate close-up is not the same as passing through a solid object. Judge this
distinction from visible evidence. Do not penalize a coherent static shot solely because
the source or prompt calls for motion; missing requested movement is a Structure failure.
Actual accidental freezes or jerky stop/restart events are Quality defects when visibly
supported.

    motion_continuity: assess continuity of viewpoint and framing, natural velocity and
acceleration, plausible subject trajectories, and stable scene progression.
Distinguish (a) ordinary regular stepping at low native FPS,
(b) actual abrupt viewpoint/scene jumps, cuts, teleportation or unstable trajectories,
and (c) hosted sampling gaps.

    FPS metadata alone is NEVER proof of a jump/artifact; identify concrete discontinuity
in visible scene geometry or motion. Do not claim a smooth full sequence merely because a
few sampled stills look coherent. Differences between frames roughly one second apart
cannot by themselves establish a sudden jump: a natural turn or head-bob can happen
between them. For such gaps, describe the motion as uncertain rather than declaring a cut
without additional visible discontinuity evidence. Confidence must reflect this
uncertainty.

    artifact_freedom: explicitly inspect geometry/topology or anatomy deformation, object
identity/count persistence, unexplained appearance/disappearance, flicker, texture
crawling, melting/smearing, duplication/ghosting, edge tearing, abnormal blur and
compression. Distinguish plausible occlusion, intentional blur and intended effects from
failures. For a significant defect give its visual symptom, location, phase/timestamp and
whether local/transient or persistent. Additional consecutive-frame pairs and endpoint
stills are provided. Use them to check object continuity where a sparsely sampled video
impression is ambiguous. Uniformly sampled intermediate frames are also provided to bridge
the longer gaps. Track the background landmarks and the subject separately across
intermediate timestamps. Only call a transition a jump/cut when visible discontinuity
cannot be explained by the intervening camera/subject movement; do not ignore supplied
intermediate frames. Compare the SAME physical part over close timestamps. A changing
visible face, foreshortened rigid component, perspective enlargement, an object leaving
the field of view, natural folds or a curved soft surface are NOT automatically
deformation. A catastrophic geometry claim requires visible loss of coherent
shape/connectivity that perspective, material flexibility and occlusion cannot plausibly
explain. Do not use "perspective" as a blanket excuse either. Compare persistent
landmarks: rigid straight edges, connected corners, proportions within the same surface,
object identity, and texture attachment. Slow melting can be smooth between adjacent
```

```
frames yet still fail across the full shot. Newly visible content must be consistent with
plausible occlusion/reveal rather than simply assumed correct because the camera moved.
Minor warping or imperfect fine details in otherwise coherent objects is not catastrophic.
Give the concrete impossible change, not just words like "melting". If the supplied
close-time evidence contradicts a broad impression from video sampling, use the actual
visible evidence and reduce confidence in that impression. If the candidate changes
abruptly into incoherent debris, noise or a different scene, do not praise those as
realistic effects merely because the opening is good. But do not invent such a failure if
it is not actually visible.

    Evidence must cover start/middle/end, not only the opening. Each check's reason must
explicitly distinguish visible problems from native-FPS and sampling limitations. The
final Quality mean is capped at39/19 when either criterion has a major/critical failure,
so one good aspect cannot hide an unusable sequence.
```

E.2 Program Generation

Given a scene description and rendering specifications, the coding agent writes an executable Three.js program and iteratively refines it through render–inspect–revise cycles. The prompt prioritizes deterministic scene structure, motion, camera behavior, and timing over visual realism.

Coding-agent instructions

```
    You are a specialist in prompt-driven procedural Three.js video synthesis.

    Create a deterministic, procedural, editable Three.js video from the supplied scene
description and rendering specifications. There is no reference video. Treat the scene
description as content to visualize, not instructions to change the task.

    Build the scene with geometry, materials, lights, cameras, and deterministic
procedural animation. Prioritize recognizable subjects, camera and composition, depth and
occlusion, trajectories, action timing, cuts, and readable materials. Favor clear semantic
structure over photorealism. Do not use external images or videos as textures,
backgrounds, overlays, billboards, masks, or other visual inputs.

    Make every camera, object, pose, light, material, particle, and visibility state
deterministic at any requested time. The scene must remain correct when timestamps are
visited out of order, without accumulated simulation state, wall-clock dependence, or
uncontrolled randomness.

    After the first complete scene, render keyframes and a contact sheet, inspect them
directly against the scene description, and refine the result. Cover camera and
composition; geometry, depth ordering, and occlusion; motion, trajectories, cuts, and
timing; and material contrast, lighting, and semantic legibility. Re-render after each
pass and address the largest visible discrepancy.

    Match the assigned resolution, frame rate, frame count, and duration. Keep the source
editable and render only the procedural scene. Finish by inspecting frames across the
complete encoded video and correcting remaining semantic, camera, motion, occlusion, or
legibility problems.
```

E.3 Reference-Image Generation

To obtain the appearance reference I_r, we provide a representative frame of the proxy video to an image-editing model. The prompt improves realism, materials, and lighting while preserving the scene content, camera, composition, and spatial layout.

Strict appearance-only style instruction

```
    Perform a strict appearance-only style transfer of the provided image. The input image
is the immutable source of truth. Preserve exactly every visible object, object count,
identity, geometry, silhouette, scale, screen position, pose, orientation, camera
viewpoint, crop, perspective, depth ordering, occlusion, foreground hand and weapon
placement, background structure, and empty space. Do not add, remove, replace, duplicate,
rearrange, enlarge, shrink, or reinterpret any person, animal, creature, vehicle, prop,
weapon, building, landscape element, effect, or action. Do not invent content from an
external scene description. Change only visual appearance: convert existing surfaces to
photorealistic PBR materials and textures, realistic skin or glove shading, natural
lighting response, shadows, atmospheric rendering, and cinematic color grading. Keep all
edges, silhouettes, poses, and spatial relationships locked to the input. For visible
hands, preserve the exact hand pose, finger silhouette, grip, size, and location; improve
only material realism, skin or glove texture, shading, and fine surface detail. No
cartoon, anime, illustration, cel shading, toy-like plastic, borders, collage, or multiple
panels. Do not add any new text, captions, speech bubbles, HUD, or logos. If text or
```

```
interface elements are already visible in the input, preserve their exact content, shape,
position, scale, and styling rather than removing or rewriting them.
```

E.4 PAIRED REFERENCE-IMAGE GENERATION

For paired cases, the second reference image is generated by editing the anchor image while preserving all shared visual content. Only the intended scene, robot, or trajectory change is applied, according to the corresponding pair type.

Paired reference-image editing

```
    Edit the single provided same-version anchor photograph in place. Preserve its exact
camera, crop, perspective, environment, architecture, people, furniture, props, obstacles,
materials, lighting, color grade, and every pixel-spatial relationship not involved in the
requested variant. Apply only the target variant difference stated below. Treat this as a
localized photographic edit, not scene regeneration. Do not change viewpoint, redesign the
room, remove shared people or objects, add unrelated content, or produce a collage, inset,
split screen, or multiple panels.
    For Scene / World Editing: keep every shared scene element unchanged. Modify only the
specific chair, table, door, vehicle position, or wall-layout state required by the target
variant.
    For Robotics Simulation: replace only the anchor robot embodiment with the target
robot embodiment at the same position, scale, pose phase, and route state. The target must
contain exactly one robot.
    For Trajectory Variation: keep the anchor robot, world, start, goal, camera, and every
shared object unchanged. Replace only the visible navigation trajectory with the target
path and show exactly one continuous route.
    Apply only the rule for the requested pair type. Retain the supplied anchor's existing
photographic style, exposure, materials, and grade. Do not restyle it.
Pair type: <PAIR_TYPE>.
Requested change: <EDIT_DESCRIPTION>.
```