# WoCA: World Simulation with Compositional Assets

**Anonymous authors**
Paper under double-blind review

## Abstract

World simulation aims to model how environments evolve and predict alternative futures with different possible outcomes. Representing these possibilities calls for a model that retains object appearance, captures how geometric states evolve, and renders the resulting observations. In this paper, we propose World Simulation with Compositional Assets (WoCA), a new object-centric world simulation approach with three core components: a set of reusable appearance assets, a trajectory diffusion model, and an asset-conditioned video renderer. We extract reusable object appearance tokens from reference observations and bind them to time-varying poses. A separate trajectory diffusion model predicts future object states and interactions from history states and visual context. To render these assets under new states, our video renderer combines global token conditioning with localized spatial features, connecting each object's stored appearance to its location in the generated video. This disentangled factorization makes appearance and motion separately addressable: object appearances can be reassigned while pose trajectories are held fixed, and trajectories can be edited while the same appearance assets are maintained. It also provides an explicit object memory that remains available across recursive state predictions for long-horizon world simulation. WoCA therefore brings video generation closer to an editable world representation where persistent entities can participate in multiple possible futures. Experiments on MOVi-D and Waymo Open show improved video quality over Neural Assets under appearance and trajectory control and demonstrate long-horizon simulation with persistent appearance assets.

## 1 Introduction

World simulators aim to represent how an environment evolves, providing a basis for predicting the future and exploring alternative outcomes (Ha & Schmidhuber, 2018). For example, one may wish to imagine a red car taking an alternative path at an intersection, or replace its appearance with that of a blue bus while preserving its route. Such scenarios require a distinction between what should persist and what is allowed to change.

One route to world simulation is to generate future visual observations, as explored by learned simulation environments such as UniSim (Yang et al., 2024) and Genie (Bruce et al., 2024). These models are powered by video generative models (Blattmann et al., 2023; Ma et al., 2025; Brooks et al., 2024; Wan et al., 2025), which provide powerful priors for visual synthesis. However, these models can generate visually plausible videos yet fail to maintain consistent object states or assign attributes and motions to the correct objects (Li et al., 2025; Sun et al., 2025). This motivates an object-centric representation that separates persistent appearance from evolving geometric state (Henderson & Lampert, 2020; Wu et al., 2024), as reusable appearance assets provide a concrete *memory* of the scene's objects, independently of the latest generated frame. Their states can then change while the same visual identities remain available for synthesis.

To address this, we propose WoCA, an object-centric world simulation framework built with compositional assets, comprising three core components: persistent object assets, generative dynamics, and video rendering. Building on Neural Assets (Wu et al., 2024), we finetune a Wan video renderer (Wan et al., 2025) with appearance-pose conditioning and learn a separate diffusion model



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.

over object trajectory dynamics. The renderer can therefore reuse the same appearance memory for predicted or edited motion. Our contributions are threefold:

- We propose an object-centric world simulation approach that separates object appearance from dynamics. We develop an asset-conditioned video renderer that binds persistent appearance to evolving poses through global token conditioning and localized appearance-pose features.
- We design a conditional dynamics model that jointly generates 3D object trajectories with recursive rollouts. These trajectories can be projected into poses as input to our video renderer, providing a route to long-video generation.
- We evaluate WoCA on MOVi-D and Waymo Open datasets, testing video quality and targeted appearance and motion edits, as well as behavior under recursive continuation. Across both datasets and all considered rendering tasks, WoCA improves all reported metrics over Neural Assets; a long-horizon Waymo rollout further demonstrates video generation beyond the training clip length with shared appearance memory.

## 2 BACKGROUND

We first review the two building blocks in our method: Neural Assets and diffusion models. Broader related work discussion can be found in Appendix A.

### 2.1 NEURAL ASSETS

Neural Assets (Wu et al., 2024) use separately encoded object appearance and pose to synthesize *static* scenes. For each object $i$ in a scene, let $A_i$ denote its appearance features and $q_{i,t}$ its target pose at scene time $t$, where $q_{i,t}$ contains the projected image coordinates and depths of selected 3D bounding-box corners, encoding the object's position, orientation and size relative to the target camera. Pairing $A_i$ with $q_{i,t}$ associates each object's visual identity with its desired geometry.

Although Neural Assets synthesize static images, they rely on video clips during training to learn reusable appearance assets from observations of the same objects in different poses. Let $I_s$ and $I_t$ denote source and target image frames sampled from the same training clip at scene times $s$ and $t$, respectively. To make appearance reusable across poses, Neural Assets train with source-to-target reconstruction. Specifically, for each matched object $i$ in the two frames, it learns to combine appearance $A_i$ extracted from the source image $I_s$ with target poses $q_{i,t}$ extracted from the target image $I_t$ to reconstruct the target image $I_t$ with an image diffusion model. As an object's pose can differ across frames, this encourages the model to learn appearance features that can transfer across poses, while the target pose supplies the geometry needed for reconstruction. Such cross-frame reconstruction provides the disentangling signal between the appearance and pose of objects.

While Neural Assets support static scene synthesis, they synthesize individual target images from supplied poses without jointly modeling temporal dynamics or predicting how object states evolve. We seek to extend this appearance-pose factorization to generative world simulation by combining temporally coherent video render with object dynamics prediction, while reusing appearance assets across changing states. Since our video render and trajectory generation components build up on diffusion and flow matching models, we will review these models in the next section.

### 2.2 CONDITIONAL DIFFUSION MODELS

Diffusion and flow matching models learn conditional generation by transforming noise into data (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021; Karras et al., 2022; Lipman et al., 2023). Let $x$ denote clean data and $C$ the corresponding conditioning information. At a sampled diffusion time $\tau$ (distinct from scene time $t$), we corrupt clean data $x$ into noisy data $x_\tau = \alpha_\tau x + \sigma_\tau \epsilon$, where $\epsilon$ is standard Gaussian noise and $\alpha_\tau, \sigma_\tau$ control the amounts of data and noise. Diffusion and flow matching models learn to reverse the corruption process.

More precisely, diffusion commonly predicts the clean sample mean $x_\theta(x_\tau, \tau, C) = \mathbb{E}[x|x_\tau, C]$ or the injected noise $\epsilon_\theta(x_\tau, \tau, C) = (x_\tau - \alpha_\tau \mathbb{E}[x|x_\tau, C])/\sigma_\tau$, whereas flow matching uses a velocity predictor $v_\theta = \mathbb{E}[\partial_\tau x_\tau | x_\tau, C]$ to predict the rate at which $x_\tau$ changes along the data-noise path. At

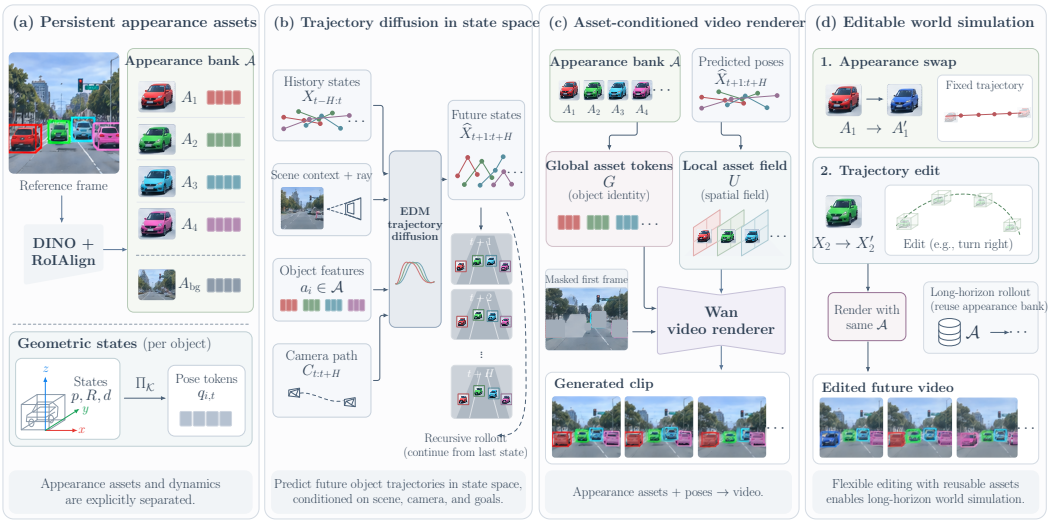


Figure 1: Overview of our proposed World Simulation with Compositional Assets (WoCA).

inference, sampling starts from Gaussian noise and proceeds toward data through iterative denoising by integrating the learned denoiser or velocity field, with conditioning information $C$ held fixed.

Diffusion and flow matching can be applied to different data structures. In latent video diffusion, $x = z_0 = \mathcal{E}(V)$ is the clean latent $z_0$ of a video clip $V$, where $\mathcal{E}$ is an video encoder and $x_\tau = z_\tau$ denotes the noisy counterpart of $z_0$. Conditions $C$ may include text or observed frames, and the sampled latent is decoded into video (Blattmann et al., 2023). In trajectory diffusion, we write $x = X$ for a sequence of states or state-action pairs; $C$ can specify observed states, and sampling produces a trajectory directly in that space (Janner et al., 2022).

## 3 World Simulation with Compositional Assets

In this section, we present our method: *World Simulation with Compositional Assets (WoCA)*. To begin with, we first explain what we want to achieve, and then we present a brief explanation on data and required annotation, followed by a detailed description on each component of our model.

**Goal.** We study generative world simulation with separately controllable object appearance and motion. Given reference appearances and object trajectories, our goal is to generate a video in which the specified objects follow the specified motion. As a result, changing an object's appearance should leave its trajectory unchanged, while changing its trajectory should preserve its appearance. We also seek to predict future trajectories from observed history and render the resulting video; recursively extending these predictions provides a route towards long-horizon simulations.

**Training data.** We use video clips with tracked 3D bounding boxes and calibrated cameras as the training set. The tracks identify the same objects across frames and specify their 3D centers, orientations and box side lengths; the associated 2D image boxes are used for appearance extraction. Object correspondence and geometry are therefore supplied by these annotations.

**Inference.** At test time, we extract object appearance from reference images using annotated 2D boxes and pair them with supplied object trajectories for controlled rendering. For future motion prediction and long-horizon simulation, we predict their continuation from observed 3D object states, conditioned on the initial visual observation and the camera path.

**Method overview.** World Simulation with Compositional Assets (WoCA) comprises three components: an appearance encoder that stores reference objects as reusable assets, a video renderer that combines these assets with per-frame object poses, and a trajectory diffusion model that predicts future motion. Figure 1 provides a high-level overview of our proposed approach. In summary:

- We train the video renderer to reconstruct a target clip by pairing each object's appearance extracted from a temporally separated reference frame in the same video sequence with its target pose sequence, while using the object-masked first frame as scene context.
- We train the trajectory model separately on past-future segments of annotated tracks, conditioned on the initial visual observation and the camera path.
- At test time, supplied or predicted trajectories are converted to its pose representation and paired with the corresponding object appearance before rendering.

In the following, we first describe how this appearance-pose representation is constructed and how the video renderer learns to use it, then introduce trajectory prediction and show how these components support editing and recursive generation for world simulation.

### 3.1 Separating Reference Appearance from Object Motion

To obtain a disentangled representation of object appearance and motion, we represent each object's reference appearance separately from its time-varying geometry.

**Appearance tokens.** Following Neural Assets (Wu et al., 2024), our appearance encoder consists of a DINO backbone (Caron et al., 2021) and a trainable output projection. For each object $i$, it extracts a $2\times2$ RoIAlign grid from the reference-image features, yielding four appearance tokens, collectively denoted by $A_i$. For foreground (object) extraction, pixels outside the selected object boxes are masked. We also encode the reference image with all annotated foreground boxes masked to obtain a background asset $A_{\mathrm{bg}}$. The object and background assets together form the appearance bank $\mathcal{A} = \{A_1, \ldots, A_N, A_{\mathrm{bg}}\}$ for a scene with $N$ foreground slots and one background slot.

**Pose representation.** The geometry input specifies each object's 3D box over time, including its center, orientation, and side lengths (i.e., length, width, height). To express each object's geometry in the target view, we project four selected corners of its 3D bounding box onto the target camera's image plane and retain their depths. Their image coordinates and depths form the 12-dimensional pose $q_{i,t}$ for object $i$ at frame $t$, similar to the pose representation described in Section 2.1. The background instead receives the relative camera transform from the reference view to each target view, so its appearance can be interpreted under camera motion. The detailed corner selection and projection conventions are given in Appendix B.1.

**Paired appearance-pose representation.** Track identities establish the correspondence between appearance assets and pose sequences. Also, the appearance bank remains fixed throughout a generated clip or rollout; missing target poses are marked as unavailable without removing the corresponding appearance assets from the bank. Therefore, recorded, edited, and predicted trajectories can all be paired with the same reference assets without re-extracting appearance from newly generated frames. We next describe how these pairs condition video generation.

### 3.2 Rendering Appearance along a Pose Sequence

Given the appearance bank and target poses, we adapt Wan2.2-TI2V-5B (Wan et al., 2025) to render a video. Our renderer retains Wan's first-frame conditioning mechanism using the first frame image as scene context, with objects being masked, while two added conditioning paths supply object appearance and motion. To match Wan's architecture, we first align the poses to its latent frames, keeping the first frame separate and selecting the last available pose in each subsequent temporal group defined by Wan. We then concatenate each reference appearance token with a learned encoding of the corresponding aligned pose. These joint features associate an object's reference appearance with its geometry at a particular latent frame and serve as the input to both conditioning paths as discussed below.

**Making the appearance-pose pairs available to the video renderer.** We project these features to Wan's context width, add a temporal embedding, and append them to Wan's text context in every cross-attention block. Through cross-attention, the video renderer can access each object's reference appearance and the target pose associated with each latent frame. Background tokens similarly combine background appearance with an encoding of the reference-to-target camera transform and is supplied through the global path. This global path gives the video renderer access to all supplied objects and the reference background throughout generation.

**Assigning object features to their target locations.** Since global cross-attention does not explicitly assign object features to spatial locations, we additionally construct a spatial feature map for each latent frame. For each object, we take the rectangle enclosing its four projected corners, clip it to the image bounds, and divide it into $2 \times 2$ cells. We project each of the four concatenated appearance-pose features into a local feature vector and place it in the corresponding cell, where overlapping assignments are averaged, and uncovered locations receive zero. The spatial map indicates which object's appearance features should guide each region of the video. The video renderer then generates the object's detailed appearance and visible outline, including which parts remain visible when objects overlap. Because the first latent slice is fixed by the conditioning image, we set its local feature contribution to zero. The spatial maps are converted to patch embeddings by a trainable convolution and added to Wan's video patch embeddings. Following ControlNet (Zhang et al., 2023), we zero-initialize this convolution, so the new local path does not initially change the pretrained patch features. Appendix B.2 specifies the temporal alignment, attention masks, and spatial construction.

These conditioning paths make reference appearance and target poses available to the video renderer. We next describe a self-supervised training objective that encourages the renderer to use both when generating the target video.

### 3.3 Learning Appearance-Pose Controlled Video Rendering

A standard image-to-video reconstruction task would not make the appearance bank necessary: the complete first frame already reveals the objects that must appear in the video. The renderer could therefore learn to continue that image while ignoring the reference assets. To remove this alternative source of object appearance, we use a self-supervised learning objective that combines the cross-frame reconstruction principle with foreground masking of the conditioning image.

**Constructing the source-to-target task.** Given a training video sequence, we first sample a target clip from it and then sample a temporally separated reference frame outside the target clip. We select objects observed both in the reference and at least once in the target clip, matching them by track identity. Their reference appearances populate the appearance bank $\mathcal{A}$, while their annotated target boxes supply the pose sequences. Only these selected tracks have independently addressable appearance assets. The reference frame may precede or follow the target clip: this pairing supervises appearance transfer across poses, not causal prediction from the reference frame. We mask all annotated foreground bounding boxes in the target first frame $I_0$ with a constant value to obtain $\widetilde{I}_0$. This masked image supplies the initial scene context without exposing the target objects' appearance. The reconstruction target remains the complete, unmodified target clip. Consequently, the video renderer is trained to recover the objects in target frames from reference appearance paired with target motion and the surrounding context, rather than directly propagate their pixels from the first image. This self-supervised source-to-target reconstruction task does not require paired appearance-edited or motion-edited videos, as the training targets come from the original annotated video sequences.

The source-to-target reconstruction task discussed above is performed in Wan's latent space. Let $z_0$ be the VAE encoding of the complete target clip. Using the flow-matching formulation, for a given noise level $\tau \in [0, 1]$, we can construct its noisy version $z_\tau = (1 - \tau)z_0 + \tau\epsilon$ with $\epsilon \sim \mathcal{N}(0, I)$. We replace the first slice of $z_\tau$ with the VAE encoding of $\widetilde{I}_0$ and denote this input by $\widehat{z}_\tau$. That slice always receives timestep zero, while the remaining slices retain noise level $\tau$. The renderer learns a velocity field $v_\theta(\widehat{z}_\tau, \tau, \mathcal{A}, \{q_{i,t}\})$ to predict their flow velocity from the appearance-pose conditions with a flow matching loss:

$$\mathcal{L}_{\mathrm{ST}} = \mathbb{E}_{\tau,\epsilon}\left[\|v_\theta(\widehat{z}_\tau, \tau, \mathcal{A}, \{q_{i,t}\}) - (\epsilon - z_0)\|^2_{\ell>0}\right], \tag{1}$$

where $\ell$ indexes latent frames. Since the first slice is supplied as a condition, we supervise only the remaining slices $\ell > 0$ as indicated in the subscript. The appearance and pose arguments in $v_\theta(\widehat{z}_\tau, \tau, \mathcal{A}, \{q_{i,t}\})$ are encoded through the global and local paths as described in Section 3.2; text and background camera conditions are omitted here for clarity. Because $z_0$ encodes the unmasked video, the loss supervises object reconstruction in the generated frames even though the conditioning image hides those objects. This encourages reference appearance to remain useful when paired with a different pose sequence. To support asset guidance at test time, we replace the reference boxes and target poses with an encoded empty-track condition for a fraction of training samples, while

retaining the same masked first image and text context. This trains the null prediction used for classifier-free guidance over the joint appearance-pose conditions.

**Retaining ordinary image-to-video generation.** Foreground masking changes the conditioning pattern of the pretrained model. Alongside source-to-target reconstruction, we also train a replay branch that receives the complete first latent slice and disables both asset paths. The replay loss $\mathcal{L}_{\text{replay}}$ uses the same target velocity and the same supervised slices as Equation 1.

**Joint training.** Combining the self-supervised source-to-target reconstruction loss with the replay loss, we train the video renderer by jointly optimizing

$$\mathcal{L}_{\text{render}} = \mathcal{L}_{\text{ST}} + \lambda_{\text{replay}} \mathcal{L}_{\text{replay}}. \tag{2}$$

Both losses use the same target clip, noise level, and noise sample in each update. This joint loss has the advantage of teaching the renderer to use external appearance assets and poses while retaining the original image-to-video capability. The trainable module comprises the appearance encoder's output projection, the global and local conditioning modules (including pose encoders, feature projections, temporal embeddings, and local convolution) and LoRA adapters (Hu et al., 2022) in Wan's attention and feed-forward layers. The DINO backbone, Wan VAE and original WAN transformer remain frozen. Appendix B.3 gives more details about reference-pair selection and training procedure.

We have discussed how the renderer synthesizes videos from appearance assets and supplied pose sequences, and we next describe how to predict object motions when no pose sequences are supplied.

### 3.4 Predicting Object Motion from Observed History

When future motion is unspecified, we jointly predict object states with a conditional trajectory diffusion model (Janner et al., 2022), using their recent history, reference visual context, and a supplied camera path. Predicting geometry separately lets the scene evolve while the renderer reuses the same appearance bank.

**A persistent coordinate frame for object motion.** We represent each object by its 3D center and a continuous 6D rotation representation (Zhou et al., 2019), holding its box dimensions at their observed seed values. The resulting 9D state channels use the fixed coordinate frame of the first target camera, so camera motion alone does not change an object's state and successive predictions share a spatial reference. Let $X_t$ collect the states of all tracked objects at time $t$, standardized using training-set statistics. The model predicts an $H$-frame continuation jointly as

$$\widehat{X}_{t+1:t+H} \sim p_\phi(\cdot \mid \mathcal{H}_t), \tag{3}$$

where $\mathcal{H}_t$ comprises recent object states, reference object appearance, initial scene features, and camera parameters spanning the history and future segment.

**Relating motion to visual context and other objects.** Past states describe motion but omit object appearance and scene context. We extract object appearance from the supplied reference and scene context from the foreground-masked first target frame, using a shared frozen DINO backbone. For each object, we pool its $2 \times 2$ RoI features into an appearance vector; for the background, we retain an $8 \times 8$ feature grid to preserve spatial context. Calibration-based ray encodings and learned grid embeddings distinguish the locations of these background features without requiring scene depth. The feature projections are trained specifically for dynamics.

To let an object's future depend on other objects, the denoiser processes all tracks and prediction times together. Each state is embedded with its object's appearance, track identity, and time index, and camera parameters are provided as additional tokens. Self-attention exchanges information across objects, times, and cameras; cross-attention then incorporates the background features into the object representations. This couples joint motion prediction to both the supplied scene context and the evolving states of the tracked objects.

**Learning to continue observed states.** We train on history-future windows from annotated tracks, keeping the history clean and perturbing only the future states. The elucidated diffusion model (EDM) objective (Karras et al., 2022) trains the denoiser to recover the clean future, with errors evaluated only on available annotations. We optimize the dynamics network and its visual projections separately from the renderer. At inference, denoising produces a future state segment; inverse normalization and calibrated box projection convert it into the same poses $q_{i,t}$ used for prescribed

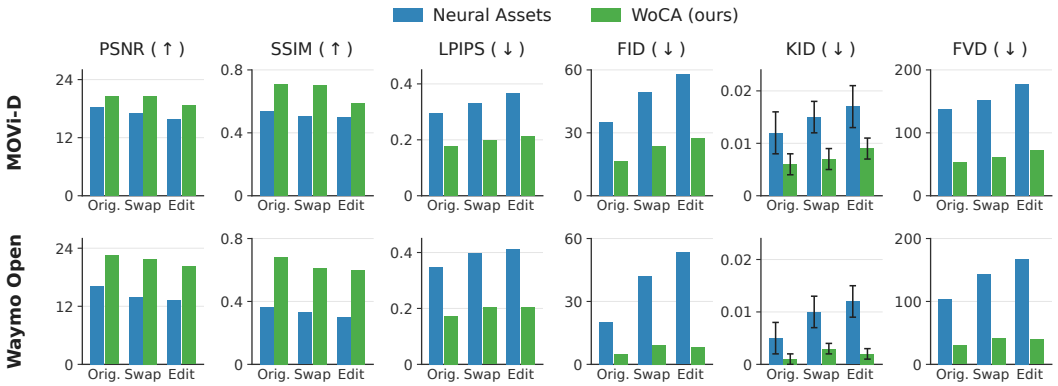


Figure 2: Video metrics on MOVi-D and Waymo Open across three controlled world simulation tasks. *Orig.* uses the original object appearances and pose sequences supplied from each dataset. *Swap* exchanges the appearance assets of two objects while keeping their original pose sequences fixed. *Edit* modifies one object's trajectory while preserving the original object appearances.

motion. These poses connect motion prediction to editing and recursive rendering, as described next. Appendix B.4 provides the model and training details.

### 3.5 World Simulation at Inference Time

At inference time, we first extract the appearance bank and obtain the desired trajectories, either from supplied tracks or from the dynamics model. We then project the 3D boxes under the supplied cameras and build both rendering conditions from the resulting appearance-pose pairs. The renderer uses the same procedure in both cases; it does not need to distinguish recorded, edited, or predicted motion. Specifically, we mask the foreground boxes in the initial scene image and hold its encoded first latent slice fixed during denoising. We generate the remaining slices with classifier-free asset guidance, comparing the asset-conditioned and encoded empty-track predictions under the same masked image condition. The empty-track prediction is the one learned through dropout, not the replay branch, whose first-image condition is different. Because the first latent slice is clamped, this procedure generates the continuation after the conditioning image; it does not reconstruct the masked foreground in that first slice. Appendix B.5 specifies the sampler and guidance.

## 4 Experiments

In this section, we evaluate our method WoCA on a variety of controlled world simulation tasks.

### 4.1 Setup

**Datasets.** We evaluate WoCA on MOVi-D and Waymo Open, covering synthetic object interactions and real-world driving scenes. MOVi-D (Greff et al., 2022) contains Kubric-generated videos of up to 23 objects, with tossed objects interacting with initially stationary ones under a fixed camera. For Waymo Open (Sun et al., 2020), we use front-camera videos and vehicle tracks from its Perception Dataset v1.4.3. Both datasets provide tracked 3D bounding boxes and camera parameters, which supply the object correspondences and geometry required by WoCA. We use the provided training and validation splits for MOVi-D and hold out $5\%$ of Waymo recording segments for validation. The target clips contain $17$ frames at $256\times256$ resolution for MOVi-D and $21$ frames at $576\times384$ for Waymo Open.

**Evaluation metrics.** We assess controlled rendering with original appearances and pose sequences, pairwise appearance swaps with poses held fixed, and trajectory edits with appearances held fixed. We report PSNR, SSIM, LPIPS, FID, KID and FVD for video quality assessment. For each dataset and each task, all six metrics are computed using $2{,}048$ video clips. Evaluation uses generated continuation frames and excludes the input conditioning frame. We report KID as the mean and standard deviation across randomly sampled feature subsets.



(a) MOVi-D (b) Waymo Open

Figure 3: Object appearance swapping examples with WoCA. In (a) MOVi-D, we exchange the appearance assets of the yellow flowerpot and the colorful box. In (b) Waymo Open, we exchange the black and white cars ahead. In both examples, their respective pose sequences remain unchanged.



(a) MOVi-D (b) Waymo Open

Figure 4: Trajectory editing examples with WoCA. In (a) MOVi-D, we alter the falling trajectory of the white cup. In (b) Waymo Open, we modify the trajectory of the white car directly ahead so that it changes into the left lane and collides with the white car in that lane.

**Implementation details.** We build the renderer on Wan2.2-TI2V-5B (Wan et al., 2025) and extract appearance features with a frozen DINO ViT-B/8 encoder (Caron et al., 2021). We train the asset-conditioning modules and LoRA adapters while keeping the pretrained backbone and VAE frozen; the trajectory diffusion model is optimized separately on annotated object tracks. For our Neural Assets (Wu et al., 2024) baseline, we use the same model backbone as WoCA with its temporal components removed to ensure a fair comparison. Full data-processing, architecture, training, and sampling details are provided in Appendix B.

### 4.2 Main Results

**Controlled video rendering.** Figure 2 compares WoCA with Neural Assets across three tasks on both datasets: *Orig.* uses unmodified object appearances and pose sequences supplied from the original datasets; *Swap* exchanges two objects' appearance assets while keeping their poses fixed; and *Edit* changes one object's trajectory while retaining the original appearances. WoCA improves all six reported metrics on both datasets across all three tasks relative to Neural Assets. The improvements under Swap and Edit show that its video-quality advantage extends to controlled changes in appearance and motion.

**Examples of appearance and motion editing.** Figure 3 illustrates appearance swaps between the yellow flowerpot and colorful box in MOVi-D, and between the black and white cars in Waymo Open. The exchanged appearances follow the recipient pose sequences, while the surrounding scene remains largely preserved. Figure 4 illustrates the complementary intervention: the white cup follows a modified falling path, while the white car ahead follows an edited lane-change trajectory that brings it into collision with the white car on its left. These motion edits reuse the original appearance assets, allowing a different simulated outcome without requiring new reference appearances. Together, the examples demonstrate separate control of appearance and motion through the same video renderer. Appendix C.1 contains additional examples of object replacement.

**Extending the simulation over time.** To extend simulation beyond the training clip length, we recursively predict trajectories and render successive windows with the same appearance bank. Figure 5 shows a highway simulation example selected out of 200 generated frames, in which a vehicle gradually overtakes the ego car on the right; their similar speeds produce subtle relative motion



Figure 5: Long-horizon highway driving simulation example with WoCA on Waymo Open. The ego vehicle travels straight at high speed, while a slightly faster car gradually overtakes it on the right with subtle relative motion.

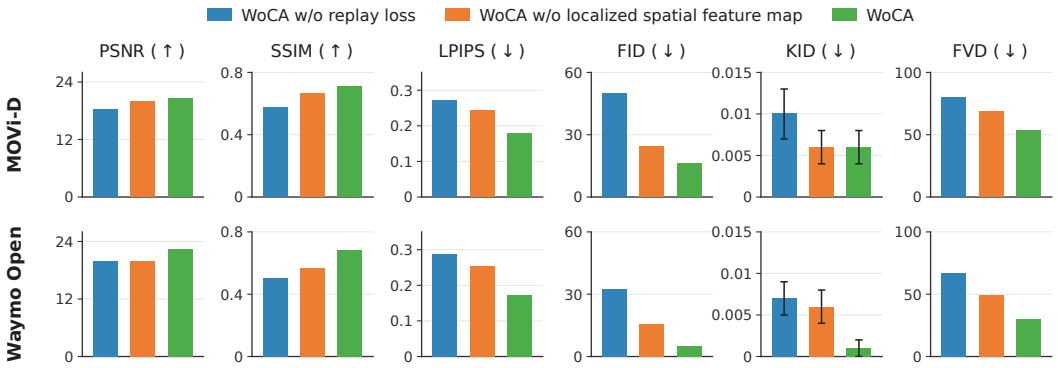


Figure 6: Video metrics on MOVi-D and Waymo Open for ablation study.

despite rapid forward travel. The example illustrates how persistent object assets can support an evolving scene over an extended horizon while the dynamics model advances its geometric states.

### 4.3 Ablation Study

Figure 6 evaluates the effects of replay loss and localized spatial feature map by removing each component separately under the original-appearance rendering setting in WoCA. The full WoCA model achieves the best or joint-best scores across all six metrics on both datasets, showing that these two modules contribute to better rendering quality.

## 5 Conclusion

We presented WoCA, a new controllable world simulation approach that disentangles object appearance and motion representations. WoCA combines a set of persistent appearance assets, a conditional diffusion model over 3D object trajectories, and a video renderer with global and localized conditioning, enabling separate control of appearance and motion. On MOVi-D and Waymo Open datasets, our method improves all reported video quality metrics over Neural Assets across controlled rendering, appearance swapping and trajectory editing; a long-horizon Waymo rollout further demonstrates generation beyond the training clip length. Ablation study supports the contributions of replay training and localized spatial conditioning in WoCA.

**Limitations and future work.** Although WoCA achieved good world simulation results, we observed that appearance preservation could be imperfect under object swapping and trajectory editing in some cases, as illustrated in Appendix C.2. Alongside larger and more diverse video datasets and higher-capacity models, explicit appearance-consistency supervision across frames and edited poses could help address the appearance failures. Also, long-horizon generation remains unstable in some complex scenes. Training on longer sequences and multi-step rollouts is a complementary direction for improving stability during recursive prediction. Furthermore, learning to infer object tracks and geometric states directly from video could further reduce dependence on annotated data.

### AI USE STATEMENT

In this work, we used generative AI tools for aiding writing and aiding implementing baselines and our methods. We have not used generative AI tools for retrieval, discovery, or drafting sections of the paper, and generating synthetic datasets and proving mathematical claims are not applicable to this work. Additionally, we used generative AI tools for modifying scientific figures. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by 2 authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

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*, pp. 22563–22575, 2023.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. OpenAI technical report, 2024.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C.Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623. PMLR, 2024.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J. Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Tianpei Gu, Guangyi Chen, Junlong Li, Chunze Lin, Yongming Rao, Jie Zhou, and Jiwen Lu. Stochastic trajectory prediction via motion indeterminacy diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17113–17122, 2022.

Yuchao Gu, Yipin Zhou, Bichen Wu, Licheng Yu, Jia-Wei Liu, Rui Zhao, Jay Zhangjie Wu, David Junhao Zhang, Mike Zheng Shou, and Kevin Tang. VideoSwap: Customized video subject swapping with interactive semantic point correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7621–7630, 2024.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Paul Henderson and Christoph H. Lampert. Unsupervised object-centric video generation and decomposition in 3D. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Roberto Henschel, Levon Khachatryan, Hayk Poghosyan, Daniil Hayrapetyan, Vahram Tadevosyan, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. StreamingT2V: Consistent, dynamic, and extendable long video generation from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2568–2577, 2025.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. GAIA-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915. PMLR, 2022.

Chiyu "Max" Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, and Dragomir Anguelov. MotionDiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9644–9653, 2023.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Dacheng Li, Yunhao Fang, Yukang Chen, Shuo Yang, Shiyi Cao, Justin Wong, Michael Luo, Xiaolong Wang, Hongxu Yin, Joseph E. Gonzalez, Ion Stoica, Song Han, and Yao Lu. WorldModelBench: Judging video generation models as world models. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with Slot Attention. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Xin Ma, Yaohui Wang, Xinyuan Chen, Gengyun Jia, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and Yu Qiao. Latte: Latent diffusion transformer for video generation. *Transactions on Machine Learning Research*, 2025.

Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11453–11464, 2021.

Mingzhi Sheng, Zekai Gu, Peng Li, Cheng Lin, Hao-Xiang Guo, Ying-Cong Chen, and Yuan Liu. FlexAM: Flexible appearance-motion decomposition for versatile video generation control. In *Computer Vision – ECCV 2026*, volume 17062 of *Lecture Notes in Computer Science*, pp. 661–677. Springer, 2026. doi: 10.1007/978-3-032-37098-3_36.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265. PMLR, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. T2V-CompBench: A comprehensive benchmark for compositional text-to-video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8406–8416, 2025.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo Open Dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2446–2454, 2020.

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing motion and content for video generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1526–1535, 2018.

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.

Xiaofeng Wang, Zheng Zhu, Guan Huang, Xinze Chen, Jiagang Zhu, and Jiwen Lu. DriveDreamer: Towards real-world-driven world models for autonomous driving. In *European Conference on Computer Vision*, 2024a.

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*, 2024b.

Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. SlotFormer: Unsupervised visual dynamics simulation with object-centric models. In *International Conference on Learning Representations*, 2023.

Ziyi Wu, Yulia Rubanova, Rishabh Kabra, Drew A. Hudson, Igor Gilitschenski, Yusuf Aytar, Sjoerd van Steenkiste, Kelsey R. Allen, and Thomas Kipf. Neural Assets: 3D-aware multi-object scene synthesis with image diffusion models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Zeqi Xiao, Yushi Lan, Yifan Zhou, Wenqi Ouyang, Shuai Yang, Yanhong Zeng, and Xingang Pan. WorldMem: Long-term consistent world simulation with memory. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Sherry Yang, Yilun Du, Seyed Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *International Conference on Learning Representations*, 2024.

Zhuoran Yang and Yanyong Zhang. ConsisDrive: Identity-preserving driving world models for video generation by instance mask. In *International Conference on Learning Representations*, 2026.

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*, pp. 3836–3847, 2023.

Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. UniPC: A unified predictor-corrector framework for fast sampling of diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5745–5753, 2019.

# A Related Work

**Generative world models and video simulation.** World models learn environment dynamics to support planning, control, and the exploration of possible futures (Ha & Schmidhuber, 2018; Hafner et al., 2020). UniSim (Yang et al., 2024) and Genie (Bruce et al., 2024) develop this idea into interactive visual simulation, while driving models such as GAIA-1 (Hu et al., 2023) and DriveDreamer (Wang et al., 2024a) incorporate action or structured scene conditions. In parallel, video diffusion models have advanced temporal synthesis (Ho et al., 2022), latent video modeling (Blattmann et al., 2023; Ma et al., 2025), and large-scale generation (Brooks et al., 2024; Wan et al., 2025). Yet evaluations of physical consistency and compositional object–attribute–motion binding reveal that visual quality alone is insufficient for reliable simulation (Li et al., 2025; Sun et al., 2025). WoCA makes individual objects directly controllable within this simulation process by representing their reference appearance separately from their evolving 3D states. This lets the same scene objects be reused under different motion hypotheses, or selected appearances be replaced while the prescribed geometry is held fixed, using a shared video renderer.

**Memory and long-video generation.** Extending a simulation beyond a short temporal context requires retaining visual information across generation windows. StreamingT2V (Henschel et al., 2025) combines recent-frame conditioning with long-term appearance preservation, while WorldMem (Xiao et al., 2025) stores frames with poses and timestamps and retrieves them to maintain consistency when revisiting a scene. WoCA supports long-video generation by recursively predicting 3D object trajectories and rendering successive video windows with a shared reference appearance bank. Predicted states become history for subsequent motion predictions, extending the simulation beyond a single video window while retaining the original object assets. This organization also makes long-video generation editable at the object level: the appearance bank is indexed by track identity and maintained separately from the evolving poses, so the same objects can be rendered along alternative motion sequences. Each video window has access to the same reference appearances, including after an object's pose condition has been temporarily absent. Long-horizon synthesis thus retains a common source of object appearance without repeatedly rebuilding assets from generated frames.

**Object-centric representations and controllable generation.** Object-centric models factor a scene into entities, from object discovery with Slot Attention (Locatello et al., 2020) to generative decomposition of moving 3D scenes (Henderson & Lampert, 2020) and autoregressive prediction of learned object slots in SlotFormer (Wu et al., 2023). Separating appearance from geometry or motion also has a substantial history: MoCoGAN decomposes video-level content and motion (Tulyakov et al., 2018), and GIRAFFE represents images through compositional 3D feature fields with separate appearance and pose controls (Niemeyer & Geiger, 2021). Neural Assets (Wu et al., 2024) are particularly close to our representation, learning reusable reference appearance through cross-frame image reconstruction with supplied target poses. Video generation and editing methods provide complementary controls. MotionCtrl separates camera and object motion conditioning (Wang et al., 2024b), and VideoSwap replaces subject identity while retaining source-video motion through semantic point correspondences (Gu et al., 2024). More recently, FlexAM separates appearance conditions from depth-aware 3D point-trajectory controls for video editing and spatial object manipulation (Sheng et al., 2026). ConsisDrive targets identity consistency in driving videos through instance conditions and attention masks that associate features with object trajectories (Yang & Zhang, 2026).

WoCA extends the reference-appearance factorization of Neural Assets from image synthesis to video simulation with learned object dynamics. Keeping reference assets separate from explicit 3D states provides both a reusable visual identity and directly editable position and orientation parameters for each tracked object. Tracked annotations establish these correspondences, while cross-frame video reconstruction with a foreground-masked initial image trains the renderer to recover object appearance from the assets under changing poses. This training design supports reuse of the same appearance bank across motion sequences, even when the conditioning image cannot supply the objects' appearance. Coupling that bank to a dynamics model further extends control from following supplied trajectories to generating and recursively continuing object motion, with the same appearance inputs available throughout rendering.

**Diffusion models for trajectories and object dynamics.** Diffusion models represent distributions over future motion by iteratively denoising trajectories. Diffuser applies this approach to state-action sequences for planning (Janner et al., 2022), while MID models stochastic pedestrian futures (Gu et al., 2022) and MotionDiffuser jointly predicts multiple agents with controllable sampling (Jiang et al., 2023). Diffusion Policy further demonstrates visual conditioning and receding-horizon execution of generated action sequences (Chi et al., 2023). These works establish diffusion as a framework for multimodal motion prediction and control; their principal outputs are trajectories or actions.

WoCA jointly predicts tracked objects' 3D positions and orientations from observed states, reference object appearance, initial scene context, and a supplied camera path. Projecting predicted 3D boxes into the same pose representation as supplied or edited trajectories makes these motion sources interchangeable inputs to the asset-conditioned video renderer. The resulting separation lets a user render multiple sampled futures with the same object appearances, or change those appearances for a chosen trajectory, using the same trained renderer. Recursive state rollout supplies the motion for long-video synthesis while retaining the original appearance assets as a common visual reference.

# B Model Details and Configurations

We first specify the appearance and geometry representations, then explain how they enter the video renderer and its training objective. We next detail the dynamics model that supplies this geometry, before describing editing, sampling, and recursive state continuation. Tables 1 and 2 respectively summarize the configurations of the renderer and trajectory model for both datasets.

## B.1 Persistent Assets and Geometric Interfaces

**Notation and track correspondence.** For a video clip $V = (I_0, \dots, I_{T-1})$, $t$ indexes image frames, $\ell$ indexes VAE latent frames, and $\tau$ indexes the renderer's data-noise path. The reference image is $I_s$; its scene time need not lie within the target clip. An object slot $i \in \{1, \dots, N\}$ maintains the same track identity across frames, with $v_{i,t} \in \{0, 1\}$ recording whether its annotated pose is available. Unused slots are padded, and an unavailable renderer pose is represented by $q_{i,t} = 0$. Thus, the absence of a pose affects the rendering condition without erasing its stored appearance. These slots come from tracked annotations or supplied tracks; the source-to-target reconstruction objective learns to use their appearance and geometry rather than discovering the correspondences.

**From a 3D box to a renderer pose.** The projection interface must preserve enough geometry to distinguish translation, rotation and size. We retain the four-corner representation of Neural Assets (Wu et al., 2024). For center $p_{i,t}$, object-to-world rotation $R_{i,t}$, and side lengths $d_{i,t}$, the selected world-space corners are

$$\begin{aligned} P_{i,t,c} &= p_{i,t} + R_{i,t}\,\mathrm{diag}(d_{i,t}/2)s_c, \\ (s_1, s_2, s_3, s_4) &= \big((-1,-1,-1),(-1,-1,1),(-1,1,-1),(1,-1,-1)\big). \end{aligned} \tag{4}$$

Here $c$ indexes corners, and the sign vectors select one corner and its three adjacent corners, corresponding to indices $0, 1, 2, 4$ in the eight-corner box ordering. To project them in Waymo's native camera coordinates, let $C_t^{\mathrm{W}}$ be the camera-to-world rotation and $o_t$ the camera center. For $y_{i,t,c} = (C_t^{\mathrm{W}})^\top (P_{i,t,c} - o_t) = (X, Y, Z)^\top$, the axes point forward, left, and up, respectively. Writing $f_{x,t}, f_{y,t}, c_{x,t}, c_{y,t}$ for focal lengths and principal-point coordinates normalized by image width and height gives

$$K_t^{\mathrm{W}} = \begin{bmatrix} c_{x,t} & -f_{x,t} & 0 \\ c_{y,t} & 0 & -f_{y,t} \\ 1 & 0 & 0 \end{bmatrix}, \tag{5}$$

$$(\xi_{i,t,c}, \eta_{i,t,c}, \zeta_{i,t,c}) = \left( c_{x,t} - f_{x,t}\frac{Y}{X}, c_{y,t} - f_{y,t}\frac{Z}{X}, X \right). \tag{6}$$

The image coordinates $(\xi, \eta)$ are dimensionless and the depth $\zeta$ uses the annotation's spatial units. The implementation stabilizes the homogeneous division using with a small-magnitude guard $X + 10^{-6}$; the displayed expression gives the underlying pinhole projection. Concatenating $(\xi, \eta, \zeta)$ for the four corners produces the pose $q_{i,t} \in \mathbb{R}^{12}$. Projected coordinates may fall outside the image, which preserves off-screen geometry; clipping is applied only when constructing the local conditioning field.

**Reference appearance extraction.** To keep the object memory tied to reference evidence, appearance is encoded once and reused as the target pose changes. Let $B_{i,s}$ be object $i$'s normalized 2D reference box, stored as $(y_{\min}, x_{\min}, y_{\max}, x_{\max})$. Let $M_s^{\text{sel}}$ cover the union of selected object boxes and $M_s^{\text{all}}$ cover all annotated foreground boxes. The two reference inputs are

$$I_s^{\text{fg}} = M_s^{\text{sel}} \odot I_s + (1 - M_s^{\text{sel}})\gamma, \qquad I_s^{\text{bg}} = (1 - M_s^{\text{all}}) \odot I_s + M_s^{\text{all}}\gamma, \quad \gamma = 0.5. \tag{7}$$

Masks broadcast over RGB channels, and pixel values lie in $[0, 1]$. The foreground input removes unrelated background pixels, while the background input prevents annotated foreground appearances from being copied into the background asset. These are box masks, so they do not assume instance segmentation.

Let $D$ denote the frozen DINO ViT-B/8 feature extractor, and let $\mathcal{R}_2(F, B)$ return a $2 \times 2$ RoIAlign grid from feature map $F$ at box $B$. For row-major RoI cell $k \in \{1, \dots, 4\}$,

$$a_{i,k} = W_{\text{app}}\, \mathcal{R}_2(D(I_s^{\text{fg}}), B_{i,s})_k + b_{\text{app}}, \qquad a_{\text{bg},k} = W_{\text{app}}\, \mathcal{R}_2(D(I_s^{\text{bg}}), B_{\text{full}})_k + b_{\text{app}}. \tag{8}$$

Here $B_{\text{full}} = (0, 0, 1, 1)$ spans the image, DINO features have dimension 768, and the shared trainable projection maps them to 512 dimensions. The four foreground tokens retain coarse within-box layout rather than reducing appearance to a single pooled vector. For empty boxes, RoI features are zeroed before projection; subsequent pose-validity masks determine which foreground tokens may be attended to.

**Background geometry.** An object asset is paired with object geometry, whereas background appearance must be interpreted under a change of viewpoint. Its geometric condition therefore encodes the transform that takes a point in the source camera frame into the target camera frame:

$$q_{\text{bg},t} = \left[\text{vec}_{\text{row}}\left((C_t^{\text{W}})^\top C_s^{\text{W}}\right); (C_t^{\text{W}})^\top (o_s - o_t)\right] \in \mathbb{R}^{12}. \tag{9}$$

The operator $\text{vec}_{\text{row}}$ stacks the three rotation-matrix rows. This token expresses camera motion relative to the stored background reference, so its reference time must remain consistent with the image used to extract $A_{\text{bg}}$.

### B.2 Conditioning the Video Renderer

**Aligning geometry with latent time.** Object annotations are given at image-frame resolution, while the renderer operates on temporally compressed video. Wan2.2's VAE has strides $(4, 16, 16)$ in time, height, and width. Its causal encoding treats frame zero separately and groups the remaining frames in fours, giving $L = 1 + (T - 1)/4$ latent frames for our clip lengths. MOVi-D uses $T = 17$ frames at $256 \times 256$ resolution, yielding a $48 \times 5 \times 16 \times 16$ VAE latent; Waymo uses $T = 21$ frames at $576 \times 384$, yielding $48 \times 6 \times 24 \times 36$. Image resolutions are written as width $\times$ height, and latent shapes as channel $\times$ time $\times$ height $\times$ width. The corresponding temporal groups are

$$\mathcal{T}_0 = \{0\}, \qquad \mathcal{T}_\ell = \{4(\ell - 1) + 1, \dots, \min(4\ell, T - 1)\}, \quad \ell \geq 1. \tag{10}$$

For each object, let $t^\star_{i,\ell} = \max\{t \in \mathcal{T}_\ell : v_{i,t} = 1\}$. When this set is nonempty, $q_{i,\ell} = q_{i,t^\star_{i,\ell}}$; otherwise $q_{i,\ell} = 0$. Using the last available pose avoids discarding an object solely because it disappears before the group's endpoint. The background transform is available at every frame and therefore uses the group endpoint.

**Global access to appearance-pose pairs.** At latent frame $\ell$, we concatenate each reference appearance token with an encoding of its aligned target pose, then project the result into a global context token:

$$j_{\ell ik} = [a_{i,k}; E_{\text{pose}}(q_{i,\ell})], \qquad g_{\ell ik} = F_{\text{g}}(j_{\ell ik}) + e_\ell. \tag{11}$$

Here $i$ indexes object slots, $k \in \{1, \dots, 4\}$ indexes the RoI cells, $E_{\text{pose}}$ is a learned pose encoder, and $[\,;\,]$ denotes concatenation. The intermediate feature $j_{\ell ik}$ is shared by the global and local paths; $F_{\text{g}}$ projects it to the global token $g_{\ell ik}$, and $e_\ell$ identifies the latent frame. The foreground pose encoder in Equation 11 is a $12 \to 1024 \to 512$ MLP with a GELU activation, whose output is repeated for the four RoI cells. Unavailable foreground poses receive a zero pose embedding. Because relative camera transforms have a different meaning from projected corners, a separate $12 \to 24 \to 12$ GELU MLP first maps $q_{\text{bg},\ell}$ into the shared pose-encoder input space. Both foreground and background

then concatenate their 512-dimensional appearance and pose features. The affine map $F_{\mathrm{g}} : \mathbb{R}^{1024} \to \mathbb{R}^{3072}$ and a learned temporal embedding form the global tokens. The temporal embeddings are initialized from $\mathcal{N}(0, 0.02^2 I)$.

Flattening these tokens over latent frames, slots, and RoI cells yields $G$, with at most $4L(N+1)$ tokens. Let $C_{\mathrm{text}}$ denote Wan's projected text context, and let $h$ denote video tokens at a native cross-attention block. The context update has the form

$$h \leftarrow h + \mathrm{CrossAttn}(\mathrm{LN}(h), [C_{\mathrm{text}}; G]; M_G)\,, \tag{12}$$

where the second argument supplies keys and values, LN is the block's layer normalization, and $M_G$ excludes foreground tokens with unavailable target poses. Background tokens remain valid. All valid asset tokens can be accessed by every video query; the temporal embedding identifies their associated latent frame without imposing frame-local attention.

**An explicit spatial assignment.** Global attention provides access to the entire appearance bank, but does not explicitly assign its contents to image locations. The local field supplies this missing assignment using the same joint features. For object $i$ at latent frame $\ell$, define the clipped rectangle $\mathcal{R}_{\ell i} = [\xi^-_{\ell i}, \xi^+_{\ell i}] \times [\eta^-_{\ell i}, \eta^+_{\ell i}]$ by taking the coordinate-wise extrema of its four projected corners and clipping them to $[0, 1]$. For a latent grid of height $H_z$ and width $W_z$, integer location $(u, v)$ has normalized center $(\bar{u}, \bar{v}) = ((u + \frac{1}{2})/W_z, (v + \frac{1}{2})/H_z)$. Inside a nondegenerate rectangle, its column and row bins are

$$\begin{aligned} c_{\ell i}(u) &= \min\left(1, \left\lfloor \frac{2(\bar{u} - \xi^-_{\ell i})}{\max(\xi^+_{\ell i} - \xi^-_{\ell i}, 10^{-6})} \right\rfloor\right), \\ r_{\ell i}(v) &= \min\left(1, \left\lfloor \frac{2(\bar{v} - \eta^-_{\ell i})}{\max(\eta^+_{\ell i} - \eta^-_{\ell i}, 10^{-6})} \right\rfloor\right). \end{aligned} \tag{13}$$

The binary mask $m_{\ell i k}(u, v)$ equals one when the pose is available, the center lies inside $\mathcal{R}_{\ell i}$, and $k = 1 + 2r_{\ell i}(v) + c_{\ell i}(u)$. It is zero outside the rectangle and for empty or degenerate rectangles. We use these masks to assign projected appearance-pose features to latent-grid locations:

$$U_\ell(u, v) = \frac{\sum_{i=1}^{N} \sum_{k=1}^{4} m_{\ell i k}(u, v) F_{\mathrm{l}}(j_{\ell i k})}{\max\left(1, \sum_{i=1}^{N} \sum_{k=1}^{4} m_{\ell i k}(u, v)\right)}, \tag{14}$$

Here $F_{\mathrm{l}}$ is the learned local projection of the shared feature $j_{\ell i k}$. Uncovered locations receive zero, and the denominator averages overlapping assignments rather than selecting a frontmost object. This is deliberately a coarse feature-placement rule: the renderer resolves actual silhouettes, occlusion, and appearance under the requested pose.

The local projection $F_{\mathrm{l}} : \mathbb{R}^{1024} \to \mathbb{R}^{512}$ is initialized as $[I,\ 0.1I]$, with zero bias. It initially retains the appearance coordinates while allowing the pose embedding to contribute. The resulting field $U \in \mathbb{R}^{512 \times L \times H_z \times W_z}$ is passed through a bias-free 3D convolution with kernel and stride $(1, 2, 2)$ and 3072 output channels. This convolution, $P_{\mathrm{asset}}$, is zero-initialized, so adding the local branch does not initially perturb Wan's video patch embeddings. The transformer input is

$$h_{\mathrm{in}} = P_{\mathrm{Wan}}(z_\tau) + P_{\mathrm{asset}}(U), \tag{15}$$

where $z_\tau$ is the noisy video latent, $P_{\mathrm{Wan}}$ is Wan's original patch-embedding layer, and $h_{\mathrm{in}}$ denotes the resulting patch embeddings. The first temporal slice of the local field is set to zero, $U_0 = 0$, because that video latent slice is supplied as an image condition.

**Adapting the pretrained video prior.** The renderer retains Wan2.2-TI2V-5B's 30 transformer blocks, hidden width 3072, 24 attention heads, and feed-forward width 14336. To learn the new conditioning interface without updating all pretrained weights, each adapted linear map uses LoRA (Hu et al., 2022):

$$W_{\mathrm{eff}} = W_0 + \frac{\alpha}{r} B_{\mathrm{L}} A_{\mathrm{L}}, \qquad A_{\mathrm{L}} \in \mathbb{R}^{r \times d_{\mathrm{in}}}, \quad B_{\mathrm{L}} \in \mathbb{R}^{d_{\mathrm{out}} \times r}. \tag{16}$$

Here $W_0$ is frozen, $r$ is the adapter rank, and $\alpha$ sets its scale. The adapters are attached to query, key, value, and output projections in self-attention and cross-attention and to both feed-forward linear maps. $A_{\mathrm{L}}$ uses Kaiming initialization and $B_{\mathrm{L}}$ starts at zero. On both datasets, the rank and scale are 128, with zero adapter dropout. Together with the trainable conditioning modules, these adapters learn how the persistent appearance bank should influence video synthesis.

Table 1: Video renderer configurations. Resolutions use width × height; batch sizes are per device. Learning rates remain constant after warmup.

| **Renderer setting** | **MOVi-D** | **Waymo** |
|---|---|---|
| Clip length | 17 frames | 21 frames |
| Video resolution | $256 \times 256$ | $576 \times 384$ |
| DINO input | $256 \times 256$ | $384 \times 256$ |
| Output token width | 512 | 512 |
| Foreground capacity | 23 objects | 23 objects |
| Background capacity | one background asset | one background asset |
| Appearance extraction | $2 \times 2$ RoIAlign | $2 \times 2$ RoIAlign |
| LoRA rank / scale / dropout | 128 / 128 / 0 | 128 / 128 / 0 |
| Optimizer / weight decay | AdamW / 0.01 | AdamW / 0.01 |
| LoRA / conditioning learning rate | $10^{-4}$ / $5 \times 10^{-4}$ | $10^{-4}$ / $5 \times 10^{-4}$ |
| GPU / batch size per GPU / gradient accumulation | 8 H20s / 32 / 1 | 8 H200s / 32 / 1 |
| Training updates / warmup | 100,000 / 100 | 20,000 / 50 |
| Gradient-norm clipping | 1.0 | 1.0 |
| ST / replay weight; asset dropout | 1 / 1; 0.1 | 1 / 1; 0.1 |
| Training / split seed | 42 / provided split | 42 / 42 |
| Video sampling (inference) | 40 steps; shift 5; asset guidance 3.5 | 40 steps; shift 5; asset guidance 3.5 |

### B.3 Training the Asset-Conditioned Renderer

**Pairing observations across time.** Source-to-target training must preserve identity correspondence while allowing appearance and target pose to come from different observations. For a target window with original-sequence indices $a, \dots, a+T-1$, renderer training on both datasets selects a source index satisfying

$$s \le a - g \quad \text{or} \quad s \ge a + T - 1 + g, \qquad g = 5. \tag{17}$$

Both images belong to the same recorded sequence; target times are reindexed to $0, \dots, T-1$ after selecting the window. The source may lie after the target window because this task supervises appearance transfer, not causal prediction from that source frame. The slot roster is formed from tracks visible in the source and in at least one target frame, retaining source boxes with normalized area at least $0.001$. A training pair requires at least one shared track and retains at most 23 foreground slots; unused slots are padded. Slot order is shuffled during training, with the same permutation applied to source features and every target pose. This preserves correspondence while avoiding a fixed ordering of particular objects across samples.

We use 17-frame target windows and the provided training and validation splits for MOVi-D. For Waymo, we use 21-frame windows from the front camera with vehicle annotations, holding out $5\%$ of recording segments for validation with split seed $42$. This segment-level split separates the recordings used to construct Waymo training and validation pairs. The video and DINO input resolutions are $256 \times 256$ on MOVi-D, and $576 \times 384$ and $384 \times 256$, respectively, on Waymo. The first-frame mask covers *all* annotated foreground boxes, including tracks omitted from the asset roster. Those omitted objects remain in the clean video target, but they have no independently addressable asset in that sample.

**Making the source assets necessary.** Let $M_0^{\text{all}}$ cover foreground boxes in the target first frame. The image condition and its latent are

$$\widetilde{I}_0 = (1 - M_0^{\text{all}}) \odot I_0 + M_0^{\text{all}} \gamma, \qquad \widetilde{c}_0 = \mathcal{E}(\widetilde{I}_0), \quad \gamma = 0.5. \tag{18}$$

The clean target latent $z_0 = \mathcal{E}(V)$ is encoded from the complete clip. Consequently, replacing its first noisy slice by $\widetilde{c}_0$ removes target foreground pixels from the conditioning image while retaining supervision for their appearance in subsequent frames. The renderer must predict the flow velocity on those subsequent slices using the available scene context, source assets, and target geometry. As in Neural Assets, this is a reconstruction-based signal for appearance-pose disentanglement; we do not add a separate statistical-independence penalty.

For the implementation-level form of Equation 1, let $\mathcal{B} = (G, U)$ collect the encoded global and local conditions, including the background camera transform in $G$. The input $\widehat{z}_\tau$ replaces the first slice of the noisy complete-video latent with $\widetilde{c}_0$. Writing $v^\star = \epsilon - z_0$ for the flow target gives

$$\mathcal{L}_{\mathrm{ST}} = \mathbb{E}_{\tau,\epsilon}\left[\|[v_\theta(\widehat{z}_\tau, \tau, \mathcal{B}) - v^\star]\|^2_{\ell>0}\right], \tag{19}$$

The notation $v_\theta(\widehat{z}_\tau, \tau, \mathcal{A}, \{q_{i,t}\})$ in the main text makes the editable inputs explicit; the implemented predictor receives their encoded condition $\mathcal{B}$, with text omitted in both expressions.

In the configured TI2V training path, an integer noise index $n$ is sampled uniformly from $\{0, \ldots, 999\}$ and $\tau = n/1000$ is used in the linear data-noise path defined in Section 3.3. All generated latent elements in a sample share this noise level. The transformer receives timestep $n$ for their video patch tokens and timestep zero for the first-slice tokens. The loss averages squared error over the generated slices $\ell > 0$ and excludes the conditioning slice $\ell = 0$, as in Equation 1. The text context is shared by the two training branches and is suppressed in the equations.

**Empty-track dropout and replay serve different purposes.** For asset guidance to measure the effect of the reference bank, training must also expose the model to a matching condition without asset evidence. With probability $p_{\mathrm{drop}} = 0.1$ per source-to-target sample, we set *all* reference boxes, including the background box, and all target poses to zero before encoding. Denote the resulting condition by $\mathcal{B}_{\mathrm{null}}$. Empty-box RoI features are zeroed before their output projection, so no reference appearance enters this branch. However, projection biases, the background pose MLP, and temporal embeddings can produce nonzero tokens. Foreground tokens are masked out, background tokens remain valid, and the local field is zero. Thus $\mathcal{B}_{\mathrm{null}}$ is the encoded empty-track condition, not an array of zero context tokens.

To express the two training branches with a shared notation, write $\mathcal{L}(c, \mathcal{B}')$ for the objective in Equation 19 with first-slice condition $c$ and asset condition $\mathcal{B}'$. Replay uses the complete first latent $c_0 = z_0^{(\ell=0)}$ and disables both asset paths, denoted by $\emptyset$, giving $\mathcal{L}_{\mathrm{replay}} = \mathcal{L}(c_0, \emptyset)$. This preserves an ordinary image-to-video training task alongside the task that relies on external appearance memory. Let $\delta \sim \mathrm{Bernoulli}(1 - p_{\mathrm{drop}})$ select the source-to-target condition, with $\mathcal{B}_\delta = \mathcal{B}$ if $\delta = 1$ and $\mathcal{B}_\delta = \mathcal{B}_{\mathrm{null}}$ otherwise. The implemented minibatch objective averages Equation 2 over training pairs and $\delta$, using $\mathcal{L}_{\mathrm{ST}} = \mathcal{L}(\widetilde{c}_0, \mathcal{B}_\delta)$. The masked image condition is unchanged by dropout; the complete image condition is used only by replay.

**Renderer training procedure.** The two objectives act on the same target clip, making their relative influence explicit. One optimizer update proceeds as follows:

1. Sample a minibatch of paired references and target clips; construct the track slots, reference masks, target first-frame masks, and projected poses.
2. Encode the complete clips into $z_0$ and the masked first frames into $\widetilde{c}_0$ with the frozen VAE.
3. Build $(G, U)$ from source appearances and aligned target poses, replacing the condition by $\mathcal{B}_{\mathrm{null}}$ for samples selected by dropout.
4. Draw one Gaussian noise tensor and one noise level per clip. Evaluate source-to-target reconstruction with $\widetilde{c}_0$, then replay with $c_0$ and both asset paths disabled, using the same draws.
5. Average the two losses with weights 1 and $\lambda_{\mathrm{replay}}$, divide by their sum, and update the conditioning modules and LoRA parameters.

Both branches contribute their weighted gradients to the same optimizer update. DINO, the VAE, and the original Wan weights remain frozen. This makes the reconstruction task responsible for learning the appearance-geometry connection while replay continues to train the adapted video model on its original conditioning pattern.

We use AdamW with weight decay $0.01$, learning rates $10^{-4}$ for LoRA and $5 \times 10^{-4}$ for the conditioning modules, and gradient-norm clipping at $1$. Both datasets use $32$ examples per device. MOVi-D training runs for $100{,}000$ updates, including $100$ linear warmup steps; Waymo training runs for $20{,}000$ updates, including $50$ warmup steps. Both use one minibatch per optimizer update, equal source-to-target and replay weights, and asset dropout probability $0.1$. Learning rates remain constant after warmup.

### B.4 Conditional Dynamics in Object-State Space

Section 3.4 predicts object motion independently of its visual rendering. Here we first define the states to be predicted, then explain how reference appearance, scene context, and the camera path condition their joint evolution. The resulting representation determines both the denoiser architecture and the future-state training objective.

Each training example contains $F$ frames of tracked 3D boxes, a calibrated camera path, and a reference image $I_s$ with object boxes. Track identities associate each object's state sequence with its reference features and renderer slot. The reference supplies visual evidence, while a history segment of the tracks supplies the observed motion; the future segment provides supervision. On both datasets, we choose a reference at least five frames before the target clip, so its appearance is available before motion prediction begins. The first target frame provides the initial scene context, as described below.

**A fixed coordinate frame for continuation.** To extend a trajectory consistently, successive predictions must use the same spatial reference. We therefore express all object states relative to the first target camera and keep this coordinate frame fixed throughout the rollout. For object $i$ at target frame $t$, let $p_{i,t}$ and $R_{i,t}$ denote its world-space center and object-to-world rotation, and let $C_0$ and $o_0$ denote the first target camera's camera-to-world rotation and center. The normalized state is

$$x_{i,t} = \mathrm{Std}\big(\big[C_0^\top (p_{i,t} - o_0); \rho_6(C_0^\top R_{i,t})\big]\big) \in \mathbb{R}^9, \tag{20}$$

Here $\rho_6$ concatenates the first two columns of a rotation matrix (Zhou et al., 2019), and $\mathrm{Std}$ standardizes each channel using training-state statistics. The resulting nine coordinates encode position and orientation; box dimensions $d_i$ remain fixed at their observed seed values. Because $C_0$ and $o_0$ do not change, camera motion alone cannot alter an object's state.

We use camera axes pointing right, up, and backward, matching MOVi-D's native convention. Waymo's forward/left/up coordinates in Appendix B.1 are related to this basis by

$$S = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix}, \qquad y^{\mathrm{P}} = S y^{\mathrm{W}}, \qquad C_t = C_t^{\mathrm{W}} S^\top. \tag{21}$$

Here W and P distinguish the Waymo and right/up/backward camera bases; world coordinates and camera centers $o_t$ are unchanged. Thus, object states and camera conditions share a consistent coordinate convention across the two datasets.

**Balancing the state channels.** Position and rotation have different numerical scales, so we standardize them before adding isotropic diffusion noise. Write the unstandardized initial-frame state as $\bar{x}_{i,t} = [\bar{p}_{i,t}; \bar{r}_{i,t}]$, where $\bar{p}_{i,t} = C_0^\top (p_{i,t} - o_0)$ and $\bar{r}_{i,t} = \rho_6(C_0^\top R_{i,t})$. For channel $j \in \{1, \dots, 9\}$, standardization is

$$\begin{aligned} x_{i,t,j} &= \frac{\bar{x}_{i,t,j} - \mu_j}{s_j}, \qquad \mu_j = \frac{1}{M} \sum_{m=1}^{M} \bar{x}_j^{(m)}, \\ s_j &= \max\left( \sqrt{\frac{1}{M-1} \sum_{m=1}^{M} (\bar{x}_j^{(m)} - \mu_j)^2}, 10^{-3} \right). \end{aligned} \tag{22}$$

The index $m$ runs over valid training states used to fit the statistics, and $M$ is their count. On both datasets, we estimate these moments from 500 training minibatches before optimization and keep the resulting statistics fixed during training and sampling. Using the same statistics for every future chunk preserves the meaning of a state when it becomes history for the next prediction.

**Separating identity from observation availability.** An object's identity should persist even when an annotated state is unavailable at a particular frame. Let $\nu_i \in \{0, 1\}$ indicate membership in the selected object roster, and let $v_{i,t}$ indicate whether its 3D state is annotated at frame $t$. We form the roster from reference-matched tracks with valid initial states, setting $\nu_i = v_{i,0}$ after source-target selection. The roster mask determines which object slots participate in attention, while $\nu_i v_{i,t}$ determines which states contribute to supervision. These masks describe track identity and annotation availability, independently of whether the projected object lies inside the image. Missing

Table 2: Trajectory diffusion model configurations. Image and DINO resolutions use width × height. Both models predict four frames per recursive step, conditioned on the latest eight observed or generated frames; sampling begins with eight observed frames.

| **Trajectory model setting** | **MOVi-D** | **Waymo** |
|---|---|---|
| Target clip | 17 frames | 21 frames |
| Foreground slots | 23 slots | 23 slots |
| Image / DINO resolution | $256 \times 256$ / $256 \times 256$ | $576 \times 384$ / $384 \times 256$ |
| Reference gap | 5 frames | 5 frames |
| Minimum box area | 0.001 | 0.001 |
| Normalization batches | 500 | 500 |
| Predicted state | Position 3 + rotation 6; standardized | Position 3 + rotation 6; standardized |
| Future / history length | 4 / 1-8 frames (training) | 4 / 1-8 frames (training) |
| Hidden width / depth / heads | 256 / 6 / 8 | 256 / 6 / 8 |
| Feed-forward width / dropout | 1024 / 0 | 1024 / 0 |
| Object appearance | Mean of $2 \times 2$ DINO RoI features; width 128 | Mean of $2 \times 2$ DINO RoI features; width 128 |
| Scene memory | $8 \times 8$ visual and ray tokens; width 256 | $8 \times 8$ visual and ray tokens; width 256 |
| Camera descriptor | 13 dimensions | 13 dimensions |
| EDM $\sigma_{\mathrm{data}}$ | 1 | 1 |
| Training $\log \sigma$ | $\mathcal{N}(-0.5, 1.2^2)$ | $\mathcal{N}(-0.5, 1.2^2)$ |
| Sampling steps | 50 | 50 |
| $\sigma_{\min}$ / $\sigma_{\max}$ / $\rho$ | 0.002 / 80 / 7 | 0.002 / 80 / 7 |
| Optimizer / weight decay | AdamW / 0 | AdamW / 0 |
| Learning rate / gradient clipping | $10^{-4}$ / 1 | $10^{-4}$ / 1 |
| Batch size / gradient accumulation | 64 per device / 1 | 64 per device / 1 |
| Training updates | 100,000 | 100,000 |
| Observed seed frames | 8 | 8 |
| Reference selection | Before clip | Before clip |
| Scene observation | Masked first target frame | Masked first target frame |

observations of a known object are represented, before standardization, by zero position and the identity-rotation code $(1, 0, 0, 0, 1, 0)$. The separate validity mask excludes these placeholders from supervision without removing the object from the roster. During training, we permute foreground slots consistently across reference features and state sequences; during inference, their ordering remains fixed throughout prediction and rendering.

**Reference appearance and spatial scene context.** Past states constrain motion, while reference appearance and background features provide complementary evidence about the objects and their surroundings. For a history of length $h$ and a future chunk of length $H$, we collect these inputs as

$$\mathcal{H}_t = \left( X_{t-h+1:t}, \{a_i^{\mathrm{dyn}}\}, \{b_j\}, \{\chi_{t'}\}_{t'=t-h+1}^{t+H}, \{\nu_i\} \right). \tag{23}$$

Here $a_i^{\mathrm{dyn}} \in \mathbb{R}^{128}$ is an object appearance vector, $b_j \in \mathbb{R}^{256}$ is a scene token, and $\chi_{t'} \in \mathbb{R}^{13}$ is a camera descriptor; $\nu_i$ indicates membership in the fixed object roster. Both observed and previously generated states can supply $X_{t-h+1:t}$ at inference. We extract object features once from $I_s$ and scene features from the foreground-masked first target frame $\widetilde{I}_0$, retaining both as the state history advances.

The masking operations in Appendix B.1 provide object-focused and background inputs to a shared frozen DINO ViT-B/8 backbone. These inputs separate reference object appearance from the scene in which future motion begins. The DINO input resolution is $256 \times 256$ for MOVi-D and $384 \times 256$ for Waymo. For each object, we average its $2 \times 2$ RoI grid and project the resulting 768-dimensional feature to $a_i^{\mathrm{dyn}}$. For the scene, an $8 \times 8$ RoI grid over the foreground-masked image retains information about different background regions. These projections are trained separately from the renderer's appearance projection, allowing the same visual backbone to support motion prediction and image synthesis.

To make the spatial background features sensitive to camera calibration, we augment each grid feature with a directional encoding:

$$r_j = \frac{\mathrm{Unproj}_{K_0}(u_j, v_j)}{\left\|\mathrm{Unproj}_{K_0}(u_j, v_j)\right\|_2} \quad \text{and} \quad b_j = F_{\mathrm{vis}}(f_j) + F_{\mathrm{ray}}(r_j) + e_j^{\mathrm{grid}}, \tag{24}$$

Here $f_j$ is the DINO feature of the scene observation at normalized grid center $(u_j, v_j)$, and $K_0$ contains the first target camera's normalized intrinsics. The map $\mathrm{Unproj}_{K_0}$ converts a grid location to a camera-axis direction, which is normalized to obtain $r_j$. The projections $F_{\mathrm{vis}}$ and $F_{\mathrm{ray}}$ each consist of an affine map followed by layer normalization, and $e_j^{\mathrm{grid}}$ is a learned grid-position embedding. In the right/up/backward convention, the unnormalized direction is

$$d_j = \left[\frac{u_j - c_{x,0}}{f_{x,0}}, \frac{c_{y,0} - v_j}{f_{y,0}}, -1\right]^\top, \qquad r_j = \frac{d_j}{\|d_j\|_2}. \tag{25}$$

For the full-image grid, cell centers are $(u_j, v_j) = ((c + \frac{1}{2})/8, (r + \frac{1}{2})/8)$ with column and row indices $c, r \in \{0, \ldots, 7\}$. The ray encodes image location under the supplied calibration, while the grid embedding identifies its discrete spatial cell. Together, these features let the denoiser distinguish background regions without estimating their depth or reconstructing a 3D scene.

**Representing the camera path separately.** Since object states use a fixed coordinate frame, the denoiser also needs to know how the requested views change over time. For each history and future frame, we supply

$$\chi_t = \left[C_0^\top (o_t - o_0); \rho_6(C_0^\top C_t); \log f_{x,t}; \log f_{y,t}; c_{x,t} - \tfrac{1}{2}; c_{y,t} - \tfrac{1}{2}\right] \in \mathbb{R}^{13}. \tag{26}$$

Here $C_t$ and $o_t$ are the target camera's camera-to-world rotation and center, and $f_{x,t}, f_{y,t}, c_{x,t}, c_{y,t}$ are its normalized intrinsics. The first nine entries describe the camera pose in the same fixed frame as the object states. The remaining four encode normalized calibration; logarithms encode the positive focal lengths, while centering the principal point expresses its offset from the image center. This descriptor supplies time-varying viewpoint information alongside the fixed reference visual features.

**Jointly denoising interacting objects.** An object's future can depend on the motion of other objects, so the denoiser processes the history and future states of all selected tracks together. We first associate each state with its object's reference appearance and its position in the prediction window. For relative window index $j \in \{0, \ldots, h + H - 1\}$, an object token is

$$o_{i,j} = W_x \check{x}_{i,j} + b_x + e_i^{\mathrm{obj}} + e_j^{\mathrm{frame}} + (W_a a_i^{\mathrm{dyn}} + b_a) + e_{\mathrm{role}(j)}^{\mathrm{role}} + e_\sigma. \tag{27}$$

The input $\check{x}_{i,j}$ is the clean state for a history token and the preconditioned noisy state $c_{\mathrm{in}}(\sigma) x_{\sigma,i,j}$ for a future token. Here $x_{\sigma,i,j} = x_{i,j} + \sigma\epsilon_{i,j}$, with $\epsilon_{i,j} \sim \mathcal{N}(0, I)$; the noise scale $\sigma$ and coefficient $c_{\mathrm{in}}$ are specified below. The pairs $(W_x, b_x)$ and $(W_a, b_a)$ define learned affine maps for state and appearance, respectively. All summands have dimension 256: $e_i^{\mathrm{obj}}$ identifies the slot, $e_j^{\mathrm{frame}}$ identifies its position within the current window, and the role embedding distinguishes history from prediction. The noise embedding $e_\sigma$ is a sinusoidal encoding of $\frac{1}{4}\log\sigma$ followed by a two-layer SiLU MLP. Relative frame indices restart at zero for each window, whereas the state coordinate frame remains fixed.

Each frame also contributes a camera token. An affine projection of $\chi_t$'s first nine entries is added to an MLP encoding of its four intrinsic entries, followed by the same frame embedding, a camera-role embedding, and $e_\sigma$. The intrinsic MLP has widths $4 \to 256 \to 256$, with SiLU and final layer normalization.

The window contributes $(h + H)N$ object-state tokens and $h + H$ camera tokens, allowing the denoiser to relate each object's motion to the supplied viewpoint. Each of six layers first applies a pre-normalized transformer encoder block, with unrestricted self-attention across the window and a feed-forward update. It then lets the history and future object-state tokens query the fixed scene memory:

$$O^{(m+1)} = \widetilde{O}^{(m)} + \mathrm{MHA}_m\left(\mathrm{LN}_m^q(\widetilde{O}^{(m)}), \mathrm{LN}_m^b(B_{\mathrm{scene}}), \mathrm{LN}_m^b(B_{\mathrm{scene}})\right). \tag{28}$$

Here $\widetilde{O}^{(m)}$ collects the history and future object-state tokens after layer $m$'s self-attention and feed-forward block, $B_{\text{scene}} = (b_1, \ldots, b_{64})$, and the three arguments of multihead attention supply queries, keys, and values. The operators $\mathrm{LN}_m^q$ and $\mathrm{LN}_m^b$ apply layer normalization to the object and scene features, respectively. Camera tokens bypass this direct scene query, but interact with its updated outputs in the next self-attention block. The scene tokens themselves remain fixed. The model uses 8 heads, hidden width 256, feed-forward width 1024, GELU in the feed-forward layers, and zero dropout. Padded object slots are masked as attention keys, and their scene-attention updates are suppressed. Only future tokens are read out through a $256 \to 9$ affine map to form the residual prediction $F_\phi$.

**Learning future-state continuation.** Training windows mirror the history-to-future structure used during recursive generation. For a sequence of $F$ frames, we sample the future start index $a$ uniformly from $\{h_{\min}, \ldots, F - H\}$ and use $h = \min(h_{\max}, a)$ history frames. The clean history is $X_{a-h:a-1}$, the prediction target is $X_{a:a+H-1}$, and the camera condition spans both segments. We use $H = 4$, $h_{\min} = 1$, and $h_{\max} = 8$, so the model learns to continue from different amounts of history. The start index is shared across sequences in a minibatch; the reference features and object roster remain fixed as the training window changes.

Within each window, we keep the history clean and add noise only to the future. This makes observed states the boundary condition for the denoising task, whose objective follows EDM (Karras et al., 2022). Let $X$ collect a future chunk of normalized 9D object states, and let $\mathcal{H}$ contain the history, appearance, scene, camera, and object-roster conditions in Equation 23. At noise scale $\sigma > 0$, the preconditioned network predicts clean states as

$$\begin{aligned} X_\sigma &= X + \sigma\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \\ \widehat{X} &= D_\phi(X_\sigma, \sigma, \mathcal{H}) = c_{\text{skip}}(\sigma) X_\sigma + c_{\text{out}}(\sigma) F_\phi(c_{\text{in}}(\sigma) X_\sigma, c_{\text{noise}}(\sigma), \mathcal{H}) \,. \end{aligned} \tag{29}$$

For sample $b$ in a minibatch, let $\mathcal{V}_b$ contain the valid scalar coordinates of its future states, and let $\widehat{X}_b$ be their prediction. To supervise annotated states without penalizing padding or missing observations, we pool noise-weighted errors over these coordinates:

$$\mathcal{L}_{\text{EDM}} = \mathbb{E}\left[ \frac{\sum_b w(\sigma_b) \sum_{v \in \mathcal{V}_b} (\widehat{X}_{b,v} - X_{b,v})^2}{\max(1, \sum_b |\mathcal{V}_b|)} \right], \tag{30}$$

where the expectation is over sampled minibatches, noise levels, and Gaussian noise. Each state with $\nu_i v_{i,t} = 1$ contributes its nine coordinates to $\mathcal{V}_b$; the denominator guard gives zero loss when no target coordinates are valid. We independently sample one noise level per sequence, shared across its future chunk, with $\log \sigma \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$, $P_{\text{mean}} = -0.5$, and $P_{\text{std}} = 1.2$.

The standard preconditioning coefficients are

$$\begin{aligned} c_{\text{skip}}(\sigma) &= \frac{\sigma_{\text{data}}^2}{\sigma^2 + \sigma_{\text{data}}^2}, \quad c_{\text{out}}(\sigma) = \frac{\sigma \sigma_{\text{data}}}{\sqrt{\sigma^2 + \sigma_{\text{data}}^2}}, \quad c_{\text{in}}(\sigma) = \frac{1}{\sqrt{\sigma^2 + \sigma_{\text{data}}^2}}, \\ c_{\text{noise}}(\sigma) &= \tfrac{1}{4} \log \sigma, \quad w(\sigma) = \frac{\sigma^2 + \sigma_{\text{data}}^2}{(\sigma \sigma_{\text{data}})^2}. \end{aligned} \tag{31}$$

We set $\sigma_{\text{data}} = 1$ to match the scale of the standardized state channels. The input and output scalings balance the signal presented to the residual network across noise levels, and the skip term retains the corresponding contribution of the noisy input. These coefficients and the noise-weighted objective follow EDM, applied here to joint object-state sequences with visual and camera conditioning.

**Optimization and configuration.** We optimize the dynamics network and its visual projections using Equation 30, with the pretrained DINO backbone and state-normalization statistics held fixed. The dynamics objective uses annotated states directly and is optimized separately from the video-rendering objective. On both datasets, we use AdamW with a constant learning rate of $10^{-4}$, zero weight decay, gradient-norm clipping at 1, and 100,000 updates. Both datasets use 64 examples per device, with one minibatch per optimizer update. Table 2 summarizes the shared architecture and the dataset-specific data, training, and sampling settings.

### B.5 Editing, Sampling, and Recursive Continuation

**Editing the appearance-geometry pairing.** Because appearance is stored independently of geometry, an edit can be expressed before running the video sampler. For an appearance permutation $\pi$ over foreground slots, the edited bank is $A'_i = A_{\pi(i)}$, while $q'_{i,t} = q_{i,t}$ and $A'_{\mathrm{bg}} = A_{\mathrm{bg}}$. All four RoI tokens move together, preserving their within-asset ordering. A donor reference can similarly replace the four tokens of one recipient slot. Recomputing both $G$ and $U$ from these pairings applies the intervention consistently to the global and local paths. This is reference-based appearance control; color changes are supplied through the chosen reference appearance.

For a geometric intervention, the user can instead supply new $p_{i,t}$ and $R_{i,t}$ and reproject them, or directly edit the projected pose sequence. For example, a gradual image-plane displacement changes an object's projected path while retaining its reference appearance. If its first and last available frames are $t_a < t_b$, the four corners at an available frame $t$ are changed as

$$\beta_t = \mathrm{clip}\left(\frac{t - t_a}{t_b - t_a}, 0, 1\right), \tag{32}$$

$$(\xi'_{i,t,c}, \eta'_{i,t,c}, \zeta'_{i,t,c}) = (\xi_{i,t,c} + \beta_t \Delta_x,\ \eta_{i,t,c} + \beta_t \Delta_y,\ \zeta_{i,t,c}). \tag{33}$$

Here $(\Delta_x, \Delta_y)$ is the requested final displacement in normalized image coordinates. Unavailable poses are left empty, depths are unchanged, and other object slots and the background transform are held fixed. The edited coordinates are not clipped before conditioning, allowing partial off-screen motion. This projected edit specifies an image-space intervention; a metric 3D intervention instead uses the world-state projection in Appendix B.1.

**Video sampling with asset guidance.** The asset-conditioned and empty-track predictions share the same masked image condition, text, noise level, and current video latent. Their difference therefore isolates the effect of the asset condition at the model-input level. Using classifier-free guidance (Ho & Salimans, 2022), the sampled velocity is

$$\widehat{v}_\theta(z, \tau) = v_\theta(z, \tau, \mathcal{B}_{\mathrm{null}}) + s_{\mathrm{A}} \left[v_\theta(z, \tau, \mathcal{B}) - v_\theta(z, \tau, \mathcal{B}_{\mathrm{null}})\right], \tag{34}$$

where $s_{\mathrm{A}}$ is the asset-guidance scale and both predictions use $z^{(\ell=0)} = \widetilde{c}_0$. The empty-track branch is the one trained by dropout, not the replay branch. Thus guidance changes the influence of external appearance and geometry while preserving the conditioning-image pattern used during training.

Sampling starts with independent standard Gaussian noise in the generated latent slices and inserts $\widetilde{c}_0$ into the first slice. Wan's flow-matching UniPC scheduler (Zhao et al., 2023) advances from high to low noise using Equation 34. Its time shift maps an unshifted noise level $\bar{\tau}$ to

$$\tau = \frac{s_{\mathrm{shift}} \bar{\tau}}{1 + (s_{\mathrm{shift}} - 1)\bar{\tau}}. \tag{35}$$

On both datasets, we use $40$ sampling steps, $s_{\mathrm{shift}} = 5$, and $s_{\mathrm{A}} = 3.5$. Before each model evaluation and after each solver update, the first latent slice is reset to $\widetilde{c}_0$; its timestep remains zero. The frozen VAE then decodes the final latent. Since this slice is fixed, the procedure synthesizes the continuation after the masked conditioning image; it does not denoise the masked foreground of that first slice.

**Sampling a state chunk.** Trajectory sampling operates on normalized states, with the conditioning information $\mathcal{H}$ held fixed throughout each chunk. Let $Y \in \mathbb{R}^{H \times N \times 9}$ denote the current noisy future chunk. We use the denoiser $D_\phi$ from Equation 29 with EDM's deterministic second-order sampler (Karras et al., 2022). For $J$ steps, its decreasing noise schedule is

$$\sigma_j = \left(\sigma_{\max}^{1/\rho} + \frac{j}{J-1}\left(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho}\right)\right)^\rho, \quad j = 0, \ldots, J-1, \qquad \sigma_J = 0. \tag{36}$$

On both datasets, we set $J = 50$, $\sigma_{\min} = 0.002$, $\sigma_{\max} = 80$, and $\rho = 7$. Starting from $Y_0 = \sigma_0 \epsilon$, with standard Gaussian $\epsilon$, each step first estimates the derivative toward the denoised states and takes an Euler proposal:

$$d_j = \frac{Y_j - D_\phi(Y_j, \sigma_j, \mathcal{H})}{\sigma_j}, \qquad \Delta\sigma_j = \sigma_{j+1} - \sigma_j, \qquad Y_{\mathrm{E}} = Y_j + \Delta\sigma_j d_j. \tag{37}$$

For $\sigma_{j+1} > 0$, a second denoiser evaluation corrects the proposal:

$$d_{\mathrm{E}} = \frac{Y_{\mathrm{E}} - D_\phi(Y_{\mathrm{E}}, \sigma_{j+1}, \mathcal{H})}{\sigma_{j+1}}, \qquad Y_{j+1} = Y_j + \frac{\Delta\sigma_j}{2}(d_j + d_{\mathrm{E}}). \tag{38}$$

The correction averages the derivative at the current state and the proposal, reducing integration error relative to the Euler step. At the terminal step $\sigma_{j+1} = 0$, we use $Y_{j+1} = Y_{\mathrm{E}}$ without another division by the noise level. The final $Y_J$ is the predicted future chunk; no additional stochastic noise is injected between solver steps.

**Recovering 3D boxes and projected poses.** The sampled states must be converted to rigid boxes before they can condition video rendering. We first undo standardization to recover the position $\bar{p}$ and rotation code $\bar{r}$ in the fixed initial-camera frame. For its rotation part $\bar{r} = [a_1; a_2]$, the 6D rotation representation (Zhou et al., 2019) is decoded by

$$b_1 = \frac{a_1}{\|a_1\|_2}, \qquad \widetilde{b}_2 = a_2 - (b_1^\top a_2)b_1, \qquad b_2 = \frac{\widetilde{b}_2}{\|\widetilde{b}_2\|_2}, \qquad \bar{R} = [b_1, b_2, b_1 \times b_2]. \tag{39}$$

We floor the norm denominators at $10^{-8}$ for numerical stability. For nondegenerate inputs this constructs an orthonormal rotation from the two predicted column vectors. The world state is then $p = o_0 + C_0\bar{p}$ and $R = C_0\bar{R}$. Combining it with the object's seed dimensions $d_i$ recovers a 3D box whose size remains fixed as its position and orientation change.

Equation 4 then constructs the four selected world-space corners of this box. For a corner $P$, write its target-camera coordinates as $C_t^\top(P - o_t) = (x, y, z)^\top$. In the right/up/backward convention, the image coordinates and depth are

$$(\xi, \eta, \zeta) = \left( c_{x,t} + f_{x,t}\frac{x}{-z},\ c_{y,t} - f_{y,t}\frac{y}{-z},\ -z \right). \tag{40}$$

The intrinsics are normalized by image dimensions. Under the basis conversion in Equation 21, this expression is equivalent to the Waymo projection in Equation 6. We stabilize division by depth with a $10^{-6}$ offset and a small-magnitude guard. Concatenating these triples in corner order $0, 1, 2, 4$ yields the renderer pose $q_{i,t}$. Calibrated projection therefore connects predicted 3D motion to the same appearance-pose conditioning used for supplied trajectories.

**Recursive state rollout.** A short-chunk model can generate a longer state sequence by feeding each sampled chunk back as history. The state history advances at each step, while the reference visual evidence and the first target camera's coordinate frame remain fixed. The state sampling and conversion to renderer poses can be written as

$$\widehat{X}_{t+1:t+H} \sim p_\phi(\cdot \mid \mathcal{H}_t), \qquad \widehat{Q}_{t+1:t+H} = \Pi_{\mathcal{K}}(\widehat{X}_{t+1:t+H}; \{d_i\}), \tag{41}$$

Here $p_\phi$ is the conditional trajectory distribution represented by the EDM denoiser, $\mathcal{H}_t$ is the condition in Equation 23, and $\widehat{Q}$ collects the projected poses of all object slots. The map $\Pi_{\mathcal{K}}$ uses the supplied camera parameters $\mathcal{K}$ and includes inverse standardization, conversion from the initial camera frame to world coordinates, and box projection with fixed seed dimensions $d_i$. On both datasets, we initialize the rollout with $h_0 = 8$ observed frames and retain the latest eight observed or generated states as history for each subsequent prediction. Given these seed states and a camera path covering the requested $F_{\mathrm{out}}$ frames, we generate the continuation as follows:

1. Establish the object roster from the valid seed tracks and match their identities to the reference objects. Retain each object's dimensions from its first valid seed state and extract the fixed object and scene features specified in Appendix B.4. Convert the observed states to the initial camera frame and standardize them.
2. Let $a$ be the current sequence length and $h = \min(h_{\max}, a)$. Select the latest $h$ normalized states as history and the supplied camera descriptors for these frames and the next $H$ frames.
3. Sample a full $H$-frame state chunk using Equations 36-38, with fresh initial noise, the latest state history, the corresponding camera window, and the fixed reference features. Append the predicted normalized states to the history.



(a) MOVi-D (b) Waymo Open

Figure 7: Object replacement examples with WoCA. In (a) MOVi-D, we replace a shoe with a box; in (b) Waymo Open, we replace a silver car with a black car. Each edit replaces the selected object's appearance asset while keeping its original pose sequence and the remaining scene conditions fixed.

4. Repeat until at least $F_{\text{out}}$ frames are available, then discard excess frames from the last chunk. Sampling retains the training chunk length even when fewer output frames remain. Repeat the final camera descriptor to condition temporary excess frames in the last chunk; these frames are discarded after sampling.
5. Undo standardization, decode rotations, convert states to world coordinates, and combine them with the fixed box dimensions. Project the resulting boxes into the 12-dimensional renderer poses, and compute background transforms from the camera of the renderer's reference image to each target camera.

Predictions are fed back in normalized 9D form, so each new chunk is conditioned in the same representation as its history. Rotation decoding and world-space conversion produce the geometric trajectory used for rendering. Object identities and dimensions remain those of the seed roster throughout this process.

**Rendering the extended state sequence with shared appearance memory.** We pair each projected trajectory with its original appearance asset using the fixed track correspondence. This yields $N$ foreground appearance-pose pairs per frame; the background condition additionally accounts for the camera of its stored reference image. For reference camera rotation $C_s$ and center $o_s$, the relative transform is

$$q_{\text{bg},t} = \left[\text{vec}_{\text{row}}(C_t^\top C_s);\ C_t^\top (o_s - o_t)\right], \tag{42}$$

where $\text{vec}_{\text{row}}$ flattens the rotation by rows. This transform remains relative to the original appearance reference even when the prediction window advances. The reference camera thus defines how the background asset is viewed, while the first target camera defines the fixed coordinates in which motion is predicted.

For a requested video window, we construct its global and local conditions from the corresponding projected poses and the stored bank $\mathcal{A}$. These conditions enter the video sampler exactly as supplied or edited poses do, together with the window's foreground-masked image condition. Recursive prediction extends the geometric sequence, while reuse of $\mathcal{A}$ preserves the source of appearance across rendering windows. The method therefore carries object appearance forward explicitly, without requiring it to be re-estimated from each newly generated frame.

# C More Examples of Controlled Video Rendering

## C.1 Object Replacement

The same appearance-pose representation also supports replacing a selected object's appearance with that of a donor reference. For object $i$, we replace all four tokens of its appearance asset $A_i$ with the donor's tokens, preserving their within-asset order while keeping the recipient's pose sequence $q_{i,t}$ fixed. The other object assets, background asset, and camera path remain unchanged. We then recompute the global and localized conditioning features from the edited appearance bank and the original poses, so both conditioning paths use the replacement appearance. Figure 7 shows a shoe replaced by a box on MOVi-D and a silver car replaced by a black car on Waymo Open. These



Figure 8: Object swapping example on Waymo Open, in which the appearances of the two cars are not preserved.



Figure 9: Trajectory editing example on Waymo Open, in which the appearances of the two cars are not preserved.

examples further illustrate how modifying the appearance bank allows a prescribed motion sequence to be reused with different object appearances through the same video renderer.

### C.2 Failure Cases

Successful editing requires the generated objects to follow the supplied poses while retaining the appearances assigned to them. Figures 8 and 9 show two Waymo Open examples in which appearance preservation fails. In the appearance-swapping example, the generated cars do not faithfully retain the intended exchanged appearances. In the trajectory-editing example, the two cars' appearances are not preserved even though their original appearance assets are retained as conditions. These failures show that separate control of appearance and geometry at the input does not always guarantee faithful appearance reproduction in the generated video. This is because although the stored assets provide a persistent reference, the renderer can still deviate from that reference when synthesizing an edited scene.