TRAJLOOM: DENSE FUTURE TRAJECTORY GENERATION FROM VIDEO

Anonymous authors
Paper under double-blind review

ABSTRACT

Predicting future motion is crucial in video understanding and controllable video generation. Dense point trajectories are a compact, expressive motion representation, but modeling their future evolution from observed video remains challenging. We propose a framework that predicts future trajectories and visibility from past trajectories and video context. Our method has three components: (1) Grid-Anchor Offset Encoding, which reduces location-dependent bias by representing each point as an offset from its pixel-center anchor; (2) TrajLoom-VAE, which learns a compact spatiotemporal latent space for dense trajectories with masked reconstruction and a spatiotemporal consistency regularizer; and (3) TrajLoom-Flow, which generates future trajectories in latent space via flow matching, with boundary cues and on-policy K-step fine-tuning for stable sampling. We also introduce TrajLoomBench, a unified benchmark spanning real and synthetic videos with a standardized setup aligned with video-generation benchmarks. Compared with state-of-the-art methods, our approach extends the prediction horizon from 24 to 81 frames while improving motion realism and stability across datasets. The predicted trajectories directly support downstream video generation and editing. Code, model checkpoints, and datasets will be released.

1 INTRODUCTION

Motion is central to video and carries information beyond static appearance (Simonyan & Zisserman, 2014). Recent video generation and editing systems rely on motion cues—including camera control, optical flow, and trajectory guidance—to shape temporal dynamics (Geng et al., 2025; Burgert et al., 2025; Wang et al., 2024; Chu et al., 2025; Deng et al., 2024). Point trajectories are a flexible motion representation. Modern trackers can recover dense trajectories with long-range correspondences and occlusion patterns (Harley et al., 2025; Karaev et al., 2025; Wang et al., 2023a; Doersch et al., 2022). This motivates a key question: given trajectories in a fixed history window, how can we predict their future positions and visibility over a future horizon?

Trajectory forecasting methods model future motion directly in trajectory space (Walker et al., 2016; Wen et al., 2023; Bharadhwaj et al., 2024; Yang et al., 2025). However, future motion is inherently uncertain and multimodal, making deterministic prediction insufficient. A recent method, *What Happens Next?* (WHN) (Boduljak et al., 2026), formulates trajectory anticipation as a generative task and is primarily conditioned on appearance cues in a given image and possibly text prompts. However, appearance-only conditioning overlooks explicit motion history. Observed trajectories already encode current dynamics and strongly constrain plausible futures. This motivates future-trajectory generation conditioned on trajectory and video history. The main challenges are preserving temporal stability and local coherence across forecast windows in diverse real-world videos.

Our goal is motion continuation: generating futures consistent with dynamics already observed in a video, rather than inferring motion from a still image alone. TrajLoom conditions on both video context and explicit trajectory history, using boundary cues to connect observed motion with the generated future. This conditioning captures ongoing dynamics and differs from WHN-style appearance-driven generation, which mainly depends on image content (Boduljak et al., 2026). A central design question is how to represent dense trajectories for learning. Most methods use *absolute* image coordinates (Boduljak et al., 2026), which couple motion with global position and induce location-dependent statistics. We instead propose Grid-Anchor Offset Encoding, which represents

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.

each trajectory as a displacement from a fixed pixel-center anchor. Absolute coordinates are recovered by adding anchors back. This offset-based parameterization emphasizes motion rather than location and provides a stable foundation for latent generative modeling.

Even with Grid-Anchor Offset Encoding, forecasting dense trajectory fields remains high-dimensional. We first learn TrajLoom-VAE, a variational autoencoder (VAE) (Kingma & Welling, 2013) that maps trajectory segments to compact spatiotemporal tokens and reconstructs dense tracks. To preserve motion structure, TrajLoom-VAE applies a spatiotemporal consistency regularizer that aligns velocities with local neighbors. We then generate future motion in this latent space using TrajLoom-Flow, a rectified-flow model conditioned on observed trajectories and video that predicts the full future window (Lipman et al., 2023; Liu et al., 2023). Lightweight boundary cues enforce continuity with observed history. Because training uses constructed interpolation states whereas inference queries self-visited ODE states, we further use on-policy K-step fine-tuning to reduce this mismatch.

For evaluation, we introduce TrajLoomBench, a unified benchmark spanning real and synthetic videos with standardized setups (*e.g.*, resolution and horizon) as common video generation benchmarks (Doersch et al., 2022; Vecerik et al., 2024; Greff et al., 2022; Li et al., 2025). Compared with WHN (Boduljak et al., 2026), our method improves motion realism, temporal consistency, and stability in both quantitative and qualitative evaluations. We also show that the predicted trajectories effectively guide motion-controlled video generation and editing (Wan et al., 2025; Chu et al., 2025).

We summarize our main contributions as follows.

1. **Trajectory encoding:** Grid-Anchor Offset Encoding, which represents each point as an offset from a fixed grid anchor to reduce location-dependent bias in dense trajectory prediction.
2. **Latent trajectory generation:** A generative framework that combines (i) TrajLoom-VAE, a VAE with masked reconstruction and spatiotemporal regularization for compact, structured trajectory latents, and (ii) TrajLoom-Flow, a rectified-flow generator conditioned on observed trajectories and video, with boundary cues and on-policy K-step fine-tuning for stable sampling over extended forecast windows.
3. **Benchmark and results:** TrajLoomBench, a unified benchmark for dense trajectory forecasting in natural videos. Our approach achieves state-of-the-art performance and provides a strong foundation for downstream applications such as motion-controlled video generation and editing.

2 Related Works

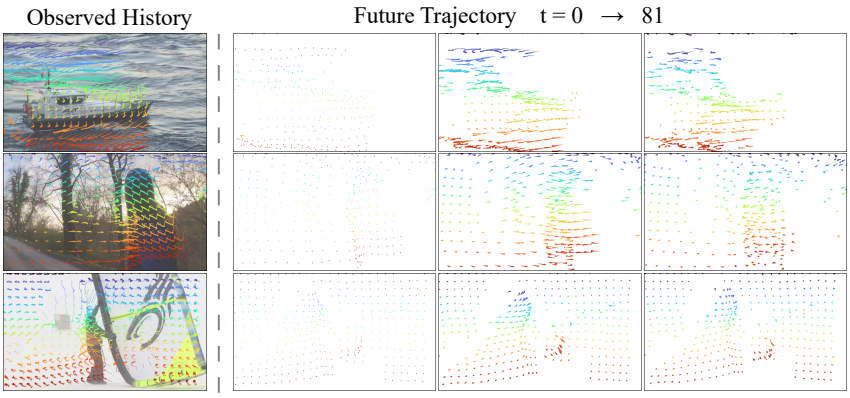


Figure 1: For each sequence, the model observes an 81-frame history (left) and predicts future trajectories for the next 81 frames (right). Predicted trajectories are shown at three times: early, middle, and final. Colors show the spatial order of query points.

Trajectories for motion anticipation. Tracking-any-point methods recover long-range trajectories and visibility from videos (Doersch et al., 2022; 2023; 2024; Karaev et al., 2025; Wang et al., 2023a; Harley et al., 2025; 2022), supported by synthetic and real-video training pipelines (Zheng et al., 2023;

Doersch et al., 2024). Beyond tracking observed motion, point trajectories provide a representation for forecasting, planning, and policy learning (Walker et al., 2016; Vecerik et al., 2024; Wen et al., 2023; Bharadhwaj et al., 2024; Yang et al., 2025). WHN generates dense future trajectories from image and text cues (Boduljak et al., 2026). ZipMo (Stracke et al., 2026) learns a motion autoencoder and uses conditional flow matching to generate trajectories from a start image and spatial or textual goals. Its emphasis is goal-conditioned motion synthesis, whereas ours is continuing the dynamics captured by an observed video and its trajectory history. Related work also extends trajectory generation to 3D. MoGe4D (Zhang et al., 2026b) infers an initial geometric representation from a single image and generates dense 3D trajectories for dynamic scene and novel-view synthesis. MolmoMotion (Zhang et al., 2026a) instead forecasts the 3D motion of queried object points from visual history and a language-specified goal. These formulations target geometry-aware scene synthesis or object-grounded 3D forecasting, rather than the continuation of a scene-wide image-plane trajectory field. Closer to our forecasting setting, Thakkar et al. (2026) predict animal motion from an image and short trajectory history, directly denoising velocities and occlusion states with one token per track. We share the use of observed motion to constrain future uncertainty, but model a different representation: a dense, grid-anchored offset field encoded into spatiotemporal latents, rather than a set of animal tracks modeled directly in trajectory space. Our consistency-regularized VAE preserves temporal and spatial relations in this representation, while boundary cues and on-policy fine-tuning target coherent continuation during latent sampling.

Motion-guided generation and editing. Controllable video generation often incorporates explicit motion controls such as optical flow, camera trajectories, or point tracks to guide temporal dynamics in diffusion-based models (Wang et al., 2024; Burgert et al., 2025; Geng et al., 2025; Wang et al., 2023b). Trajectory-conditioned methods use sparse or dense tracks as a low-level interface for directing object motion, as exemplified by DragNUWA, MagicMotion, Tora, and SG-I2V (Yin et al., 2023; Li et al., 2025; Zhang et al., 2025b; Namekata et al., 2025). Wan-Move is particularly relevant to our applications. Built on the Wan image-to-video backbone, it employs latent trajectory guidance that propagates information along dense point trajectories, enabling direct point-level motion control (Chu et al., 2025; Wan et al., 2025). Interactive editing similarly uses sparse point constraints to manipulate deformation and motion. These range from DragGAN and DragDiffusion to video drag methods such as DragVideo (Pan et al., 2023; Shi et al., 2024; Zhang et al., 2025a; Deng et al., 2024). Our future-trajectory generator complements these controllers. We adopt Wan-Move because it consumes dense trajectories *directly*, allowing our predicted tracks to integrate without additional motion representations (Chu et al., 2025).

3 TRAJLOOM: DENSE FUTURE TRAJECTORY GENERATION

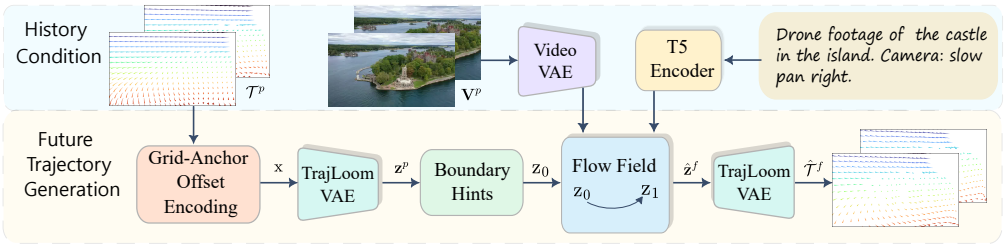


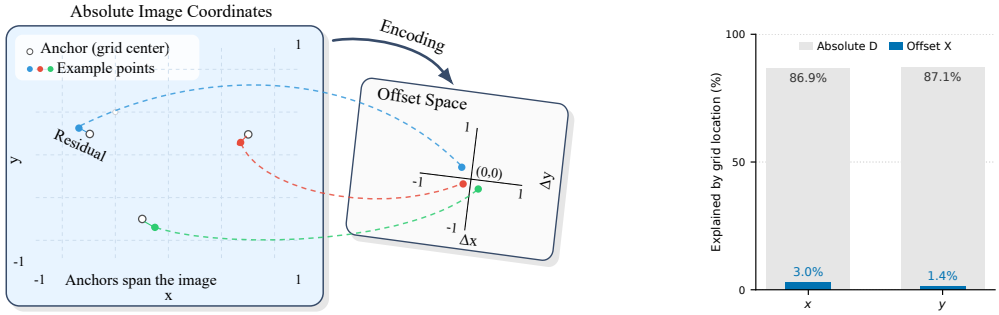
Figure 2: **Overview of our pipeline.** Given observed trajectories $\mathcal{T}^p$, we rasterize and encode them with Grid-Anchor Offset Encoding into a dense offset field, then compress with TrajLoom-VAE into history latents $\mathbf{z}^p$. Conditioned on $\mathbf{z}^p$ and video features, TrajLoom-Flow generates future latents via rectified-flow integration with boundary hints, which are decoded by TrajLoom-VAE into future trajectories $\hat{\mathcal{T}}^f$.

We study future-motion generation from observed history in a video clip. A video is denoted as $\mathbf{V} \in \mathbb{R}^{T\times H\times W\times 3}$, where T is the total number of frames, and each frame has a spatial resolution of $H \times W$. Motion is represented as a set of N trajectories, $\mathcal{T} = \{\tau_n\}_{n=1}^{N}$, where each trajectory $\tau_n = \{(x_{n,t}, y_{n,t})\}_{t=0}^{T-1}$ tracks one reference 2D point through time. At each frame t, the location of the point is $(x_{n,t}, y_{n,t})$, accompanied by a visibility indicator $v_{n,t} \in \{0, 1\}$. For clarity, we split each clip into a past history window of length T_p and a future window of length T_f, where $T_p + T_f = T$. Thus, $\mathbf{V}$ is divided as $\mathbf{V} = (\mathbf{V}^p, \mathbf{V}^f)$. Each trajectory τ_n is similarly partitioned into a

history segment, $\tau_n^p = \{(x_{n,t}, y_{n,t})\}_{t=0}^{T_p-1}$, and a future segment, $\tau_n^f = \{(x_{n,t}, y_{n,t})\}_{t=T_p}^{T-1}$. Visibility indicators are split in the same manner. Given the observed history trajectories $\mathcal{T}^p = \{\tau_n^p\}_{n=1}^N$, their corresponding visibility indicators, the history video clip $\mathbf{V}^p$, and a text caption, our goal is to generate the corresponding future trajectories $\hat{\mathcal{T}}^f$.

Our pipeline has three stages: Grid-Anchor Offset Encoding densifies sparse trajectories into grid-anchored offsets (Section 3.1); TrajLoom-VAE compresses dense fields into compact spatiotemporal latents with masked reconstruction and spatiotemporal regularization (Section 3.2); and TrajLoom-Flow jointly predicts future latents via a history-conditioned rectified flow, then decodes them into trajectories (Section 3.3). To reduce train-test mismatch from ODE integration (Chen et al., 2018), we further apply on-policy K-step fine-tuning.

3.1 Grid-Anchor Offset Encoding



(a) **Grid-Anchored Offset Encoding.** We encode trajectories as displacements from pixel-center anchors: instead of absolute coordinates (left), each point is given by its offset $(\Delta x, \Delta y)$ from its local anchor (right), producing a zero-centered, location-consistent representation.

(b) Variance explained by grid location for absolute and offset coordinates. We measure coordinate variance attributable to grid position using time-averaged coordinates at each grid cell.

Figure 3: Grid-Anchor Offset Encoding converts absolute trajectories into offset space, reducing the bias of absolute coordinates.

Starting from trajectories $\mathcal{T} = \{\tau_n\}_{n=1}^N$, Grid-Anchor Offset Encoding constructs a dense trajectory representation on the video grid. Grid-Anchor Offset Encoding represents each pixel by its displacement from a local pixel-center anchor, rather than by absolute coordinates. This yields offsets that are consistent and comparable across grid locations.

Concretely, trajectories are extracted from a stride-s grid. Let $H_c = H/s$ and $W_c = W/s$ so that $N = H_c W_c$. Rasterization produces (i) a dense absolute coordinate field $\mathbf{D} \in \mathbb{R}^{T\times H\times W\times 2}$ and (ii) a dense visibility mask $\mathbf{M} \in \{0,1\}^{T\times H\times W}$. For a pixel location $p = (h, w)$, the corresponding coarse-grid trajectory index is $\pi(h, w) = \lfloor h/s \rfloor W_c + \lfloor w/s \rfloor + 1$ and the dense fields are defined by $\mathbf{D}(t, h, w) = \left(x_{\pi(h,w),t}, y_{\pi(h,w),t}\right)$ and the mask is $\mathbf{M}(t, h, w) = v_{\pi(h,w),t}$. By construction, $\mathbf{D}$ and $\mathbf{M}$ are piecewise constant within each $s \times s$ stride cell. All coordinates are represented in a normalized image coordinate system.

For each pixel p at location (h, w) in the dense field, we define its normalized pixel-center anchor $\mathbf{G}(p) \in \mathbb{R}^2$, where $\mathbf{G}(p) = \left[2\frac{w+\frac{1}{2}}{W} - 1,\ 2\frac{h+\frac{1}{2}}{H} - 1\right]^\top$. The offset-encoded trajectory field is then $\mathbf{X}(t, p) = \mathbf{D}(t, p) - \mathbf{G}(p)$. From now on, the offset field $\mathbf{X}$, together with the visibility mask $\mathbf{M}$, serves as the trajectory representation. Absolute coordinates can be recovered from this representation.

We validate Grid-Anchor Offset Encoding by comparing coordinate variance under absolute and relative representations. With absolute coordinates $\mathbf{D}$, trajectory variance is dominated by grid location: points from different grid cells are centered at different image positions, so the overall variance is large even when local motion is similar. To quantify this effect, we compute the fraction of coordinate variance explained by grid location, using a visibility-weighted time-averaged coordinate at each grid position as the location baseline. Figure 3b shows that this explained variance is high for absolute coordinates but much lower for relative offsets. Using offsets $\mathbf{X} = \mathbf{D} - \mathbf{G}$ removes most

location-driven variance and yields a more uniform representation focused on local displacement. More details can be found in Appendix F.1.

3.2 TRAJLOOM-VAE

Modeling future motion directly in dense trajectory-field space is high-dimensional. To obtain a compact representation for generative modeling, we learn a variational autoencoder (VAE) in the latent space. TrajLoom-VAE is trained on temporal segments $\mathbf{x}$ from the offset-encoded trajectory field $\mathbf{X}$ (Section 3.1) and the corresponding visibility mask $\mathbf{m}$. Given a segment $\mathbf{x}$, the encoder defines an approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$, and the decoder reconstructs it as $\hat{\mathbf{x}} = \psi(\mathbf{z})$.

A masked pointwise reconstruction loss encourages $\hat{\mathbf{x}}$ to match $\mathbf{x}$ at visible locations, but it does not directly model temporal evolution or local relative motion. As a result, reconstructions with reconstruction error alone still show temporal jitter or local spatial inconsistency (see Appendix F.2). To enforce temporal smoothness and local coherence, we propose a *spatiotemporal consistency regularizer* that matches (i) temporal velocities and (ii) multiscale spatial neighbor relations between the target segment $\mathbf{x}$ and reconstruction $\hat{\mathbf{x}}$.

3.2.1 Spatiotemporal consistency regularizer. The regularizer combines a temporal velocity term and a multiscale spatial neighbor term. Let Ω denote the set of spacetime indices (t, p) within a segment window. The trajectory value at (t, p) is $\mathbf{x}(t, p) \in \mathbb{R}^2$, and the corresponding visibility is $\mathbf{m}(t, p) \in \{0, 1\}$. All consistency terms are computed only on valid, visible pairs and are normalized by the number of such pairs so that the loss scale does not depend on how many points are visible.

We discourage the frame-to-frame jittering by the following loss,

$$L_{\text{temporal}} = \frac{1}{\sum_{(t,p)\in\Omega} \mathbf{m}_{\text{pair}}(t,p)} \sum\nolimits_{(t,p)\in\Omega} \mathbf{m}_{\text{pair}}(t,p) \; \|\Delta_t \hat{\mathbf{x}}(t,p) - \Delta_t \mathbf{x}(t,p)\|_1 \,, \tag{1}$$

where $\Delta_t \mathbf{x}(t, p) = \mathbf{x}(t, p) - \mathbf{x}(t-1, p)$, $\Delta_t \hat{\mathbf{x}}(t, p) = \hat{\mathbf{x}}(t, p) - \hat{\mathbf{x}}(t-1, p)$, and $\mathbf{m}_{\text{pair}}(t, p) = \mathbf{m}(t, p)\mathbf{m}(t-1, p)$. Basically, it matches the temporal consistency between the reconstruction and the ground truth for locations that are visible.

To preserve spatial consistency, we additionally match relative motion among neighboring locations. Let $\mathcal{S}$ be a set of horizontal/vertical offsets at multi-hop distances. For each neighboring location $\delta \in \mathcal{S}$, we define $\Delta_\delta \mathbf{x}(t, p) = \mathbf{x}(t, p + \delta) - \mathbf{x}(t, p)$ and $\Delta_\delta \hat{\mathbf{x}}(t, p) = \hat{\mathbf{x}}(t, p + \delta) - \hat{\mathbf{x}}(t, p)$, and introduce the following loss,

$$L_{\text{spatial}} = \frac{1}{\sum_{\delta\in\mathcal{S}} \alpha_\delta} \sum\nolimits_{\delta\in\mathcal{S}} \alpha_\delta \, \frac{\sum_{(t,p)\in\Omega} \mathbf{m}_\delta(t,p) \; \|\Delta_\delta \hat{\mathbf{x}}(t,p) - \Delta_\delta \mathbf{x}(t,p)\|_1}{\sum_{(t,p)\in\Omega} \mathbf{m}_\delta(t,p)}. \tag{2}$$

The loss is only activated when both neighboring locations are visible since $\mathbf{m}_\delta(t, p) = \mathbf{m}(t, p)\mathbf{m}(t, p + \delta)$. Each offset $\delta \in \mathcal{S}$ has hop length $h_\delta = \|\delta\|_1 \in \{1, 2, 4\}$ and weight $\alpha_\delta = 1/h_\delta$, giving weights $1, 0.5, 0.25$. The outer sum is normalized by $\sum_{\delta\in\mathcal{S}} \alpha_\delta$. Larger offsets receive smaller weights, emphasizing local motion, while the reconstruction loss captures global motion.

Therefore, the full spatiotemporal consistency regularizer is,

$$L_{\text{st}} = \lambda_{\text{temporal}} L_{\text{temporal}} + \lambda_{\text{spatial}} L_{\text{spatial}}, \tag{3}$$

where $\lambda_{\text{temporal}}$ and λ_{spatial} are weighting coefficients.

Appendix F.2 and Figure 9 provide a toy example showing why pointwise reconstruction alone is insufficient and how the consistency regularizer separates smooth from jittery solutions.

3.2.2 Training objective. The reconstruction loss of our VAE is as follows,

$$L_{\text{rec}} = \sum\nolimits_{(t,p)\in\Omega} w(t,p)\, \rho\big(\hat{\mathbf{x}}(t,p) - \mathbf{x}(t,p)\big), \tag{4}$$

where the normalized mask $w(t, p) = \mathbf{m}(t, p) \big/ \sum_{(j,q)\in\Omega} \mathbf{m}(j, q)$ ensures that we only consider visible locations. ρ is the Huber loss (Huber, 1992).

We train TrajLoom-VAE by minimizing reconstruction error, the KL divergence, and the spatiotemporal consistency regularizer,

$$L_{\text{vae}} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[L_{\text{rec}} + L_{\text{st}}] + \beta \, D_{\text{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}) \,\|\, \mathcal{N}(\mathbf{0}, \mathbf{I})) \,, \tag{5}$$

where β is the weighting of the KL term. In practice, we set $\beta = 5{\times}10^{-5}$, $\lambda_{\text{temporal}}$ as 0.1, and λ_{spatial} as 0.2 for the spatiotemporal consistency regularizer.

3.3 TRAJLOOM-FLOW

We generate future motion in the latent space learned by TrajLoom-VAE. Given a history segment $\mathbf{x}^p$ and a future segment $\mathbf{x}^f$ (both from the offset field $\mathbf{X}$), we obtain their latent representations with the frozen VAE encoder. We use the posterior mean as a deterministic encoding:

$$\mathbf{z}^p = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^p)}[\mathbf{z}], \qquad \mathbf{z}^f = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^f)}[\mathbf{z}]. \tag{6}$$

TrajLoom-Flow models the conditional distribution of future latents given observed history and predicts the full future window jointly.

To keep predictions consistent with observed motion, we summarize all conditioning signals as $\mathbf{c}$. In our setting, $\mathbf{c}$ includes history trajectory latents $\mathbf{z}^p$, history visibility, history-video features, and a pooled text embedding. The generator is a latent flow matching model, parameterized by a conditional velocity field $v_\theta(\mathbf{z}_t, t, \mathbf{c})$.

3.3.1 Boundary hints. Because we generate the entire future window jointly, we provide explicit boundary information so the model can align future predictions with the observed past. We use two lightweight mechanisms: (i) a boundary-anchored initialization of $\mathbf{z}_0$, and (ii) token-aligned fusion of history latents into the query stream.

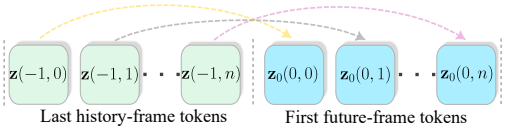


Figure 4: The first future latent slice is initialized from the last history slice plus Gaussian noise. Remaining slices use standard Gaussian noise.

Let $\Lambda = \{(k, n)\}$ index latent tokens, where k denotes a latent time index and n denotes a spatial token index, and let $\mathbf{z}(k, n) \in \mathbb{R}^C$ denote a token. Denoting by $\mathbf{z}(-1, n)$ the latent at the last history time step, we initialize a future slice selected for boundary anchoring as

$$\mathbf{z}_0(k, n) = \mathbf{z}(-1, n) + \sigma_0 \, \boldsymbol{\eta}(k, n), \qquad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{7}$$

where σ_0 controls the noise scale. We apply this rule only at $k = 0$; Appendix G.1 specifies the initialization of the remaining slices.

Beyond conditioning through $\mathbf{c}$, we inject history latents $\mathbf{z}^p$ into the velocity network through a small token-aligned fusion module, providing a direct boundary cue. More details are in Appendix G.1 and the ablation study in Appendix G.2.

3.3.2 Flow matching. To model a distribution over future latents without autoregressive rollout, we adopt rectified flow (Lipman et al., 2023; Liu et al., 2023) and learn a conditional latent velocity field. Denote the future target as $\mathbf{z}_1 = \mathbf{z}^f$, and let $\mathbf{z}_0$ be the history-conditioned source state. For flow time $t \in [0, 1]$, an intermediate state is

$$\mathbf{z}_t = (1 - t)\, \mathbf{z}_0 + t\, \mathbf{z}_1 + \sigma\, \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{8}$$

and the model predicts a conditional velocity field $v_\theta(\mathbf{z}_t, t, \mathbf{c})$. Under linear interpolation, the target velocity is $u_t = \mathbf{z}_1 - \mathbf{z}_0$, and training matches v_θ to u_t.

To emphasize visible future regions, we derive effective token weights $\mathbf{m}^{\text{tok}}(k, n) \in [0, 1]$ from future visibility pooled onto the VAE token grid, with the treatment of invisible tokens specified in Appendix A. We define normalized token weights as $w_\Lambda(k, n) = \mathbf{m}^{\text{tok}}(k, n) / \sum_{(j,q)\in\Lambda} \mathbf{m}^{\text{tok}}(j, q)$. The resulting visibility-weighted flow-matching loss is

$$L_{\text{fm}} = \mathbb{E}_{t,\boldsymbol{\epsilon}} \left[\frac{1}{C} \sum\nolimits_{(k,n)\in\Lambda} w_\Lambda(k, n) \, \| v_\theta(\mathbf{z}_t, t, \mathbf{c})(k, n) - u_t(k, n) \|_2^2 \right], \tag{9}$$

where C denotes the number of latent channels in $\mathbf{z}$.

3.3.3 On-policy fine-tuning. Flow matching trains v_θ using interpolated states $\mathbf{z}_t$, while sampling evaluates v_θ on states produced by integrating the learned ODE. This mismatch can cause drift because the model is queried off the training path. We therefore apply an on-policy K-step rollout loss to fine-tune v_θ on its own visited states.

Let $\varepsilon = t_0 < t_1 < \cdots < t_K = 1 - \varepsilon$ be an increasing time grid and set $\tilde{\mathbf{z}}_0 = \mathbf{z}_0$. A detached forward-Euler rollout generates visited states

Table 1: Future-trajectory generation at 24 and 81 frames. Top: comparison with WHN (L). Bottom: ablation of Grid-Anchor Offset Encoding. Lower is better; metric scales are shown in parentheses.

Frames	Metric	Method	Kinetics	RoboTAP	Kubric	MagicData (E)
24	FVMD ↓	WHN (L)	8999	7587	4872	10383
		Ours	**3626**	**2467**	**1338**	**2968**
	FlowTV (10^2) ↓	WHN (L)	24.71	27.26	12.56	26.74
		Ours	**7.53**	**6.03**	**6.03**	**7.11**
	DivCurlE (10^3) ↓	WHN (L)	34.69	37.39	9.89	40.04
		Ours	**3.97**	**2.03**	**2.09**	**2.99**
81	FVMD-Long (10^{-3}) ↓	w/o offset	46.61	36.43	31.66	41.97
		Ours	**35.06**	**21.75**	**7.24**	**28.27**
	FlowTV (10^2) ↓	w/o offset	15.92	15.26	15.54	15.01
		Ours	**7.65**	**5.96**	**5.03**	**7.11**
	DivCurlE (10^3) ↓	w/o offset	19.10	16.27	17.66	16.13
		Ours	**4.21**	**1.89**	**1.45**	**3.09**

$$\tilde{\mathbf{z}}_{i+1} = \tilde{\mathbf{z}}_i + (t_{i+1} - t_i)\,\mathrm{sg}[v_\theta(\tilde{\mathbf{z}}_i, t_i, \mathbf{c})], \qquad i = 0, \ldots, K-1. \tag{10}$$

Denote $\mathbf{v}_i = v_\theta(\tilde{\mathbf{z}}_i, t_i, \mathbf{c})$. Endpoint-consistent velocity targets are

$$\mathbf{v}_i^{(1)} = \frac{\mathbf{z}_1 - \tilde{\mathbf{z}}_i}{1 - t_i}, \qquad \mathbf{v}_i^{(0)} = \frac{\tilde{\mathbf{z}}_i - \mathbf{z}_0}{t_i}. \tag{11}$$

The on-policy rollout loss is defined as,

$$L_{\text{k-step}} = \frac{1}{K}\sum_{i=0}^{K-1}\Big(w_1 \left\|\mathbf{v}_i - \mathbf{v}_i^{(1)}\right\|_{w_\Lambda}^2 + w_0 \left\|\mathbf{v}_i - \mathbf{v}_i^{(0)}\right\|_{w_\Lambda}^2\Big), \tag{12}$$

where $\|f\|_{w_\Lambda}^2 = \frac{1}{C}\sum_{(k,n)\in\Lambda} w_\Lambda(k,n)\ \|f(k,n)\|_2^2$. We further introduce endpoint consistency, using uniform token weights in its norms for this term only,

$$L_{\text{cons}} = \frac{1}{K-1}\sum_{i=1}^{K-1}\Big(\|\hat{\mathbf{z}}_i^{(1)} - \mathrm{sg}[\hat{\mathbf{z}}_{i-1}^{(1)}]\|_{w_\Lambda}^2 + \|\hat{\mathbf{z}}_i^{(0)} - \mathrm{sg}[\hat{\mathbf{z}}_{i-1}^{(0)}]\|_{w_\Lambda}^2\Big), \tag{13}$$

where $\hat{\mathbf{z}}_i^{(1)} = \tilde{\mathbf{z}}_i + (1 - t_i)\mathbf{v}_i$ and $\hat{\mathbf{z}}_i^{(0)} = \tilde{\mathbf{z}}_i - t_i\mathbf{v}_i$. The final loss is,

$$L = L_{\text{fm}} + \lambda_{\text{k-step}}\Big(L_{\text{k-step}} + \gamma\, L_{\text{cons}}\Big). \tag{14}$$

In practice, we apply this loss on a small sub-batch to limit overhead, and use small $\lambda_{\text{k-step}}$ and γ to stabilize training. More details are in Appendix A.4.

3.3.4 Sampling. At inference, we obtain future latents by integrating the learned rectified-flow ODE from the history-conditioned source state. We take the final state as the generated future latent $\hat{\mathbf{z}}^f = \mathbf{z}(1)$. Finally, $\hat{\mathbf{z}}^f$ is decoded using the frozen decoder to obtain future dense trajectories.

4 Experiments

4.1 Implementation Details

Baseline and trajectory extraction. We compare against WHN (L) (Boduljak et al., 2026) in its single-image-conditioned setting, matching its 24-frame generation window. Our model additionally supports joint generation of 81-frame futures. For training and evaluation, we extract trajectories with AllTracker (Harley et al., 2025) on a stride-32 grid using the first frame as reference, then rasterize them into dense trajectory fields and visibility masks (Section 3).

Data and benchmark. After filtering, MagicData (Li et al., 2025) provides 16k videos; we use the first 162 frames at 480p and a 90%/10% training/validation split. TrajLoomBench evaluates on TAP-Vid-Kinetics (Doersch et al., 2022), RoboTAP (Vecerik et al., 2024), Kubric (MOVi-A) (Greff et al., 2022), and held-out MagicData, with unified resolution, temporal length, and preprocessing for each evaluation setting. Kubric follows the WHN (L) MOVi-A configuration, with longer videos re-rendered as needed. Data preprocessing is detailed in Appendix A.1.

Model and training. Both TrajLoom-VAE and TrajLoom-Flow use a Latte/DiT backbone (Peebles & Xie, 2023; Ma et al., 2025), with TrajLoom-Flow matching the WHN (L) model scale. Full architectures, optimization settings, and on-policy fine-tuning details are provided in Appendix A.

Table 2: Fidelity of trajectory VAE reconstruction. Visibility-masked endpoint error (VEPE, ↓).

Frames	Method	Kinetics	RoboTAP	Kubric	MagicData (E)
24	WHN (L)-VAE	63.09	29.35	15.97	69.88
	Ours	**1.99**	**1.59**	**0.75**	**1.72**
81	WHN (L)-VAE	63.31	31.52	16.27	67.97
	Ours	**2.04**	**1.61**	**0.77**	**1.72**

4.1.1 Evaluation Metrics.

We assess motion quality and reconstruction fidelity.

(i) *Fréchet Video Motion Distance (FVMD).* FVMD (Liu et al., 2024) compares trajectory-derived motion-feature distributions using Fréchet distance. By default, we use 16-frame feature windows (stride 1) and visibility masking; FVMD-Long instead evaluates the full 81-frame future window as a single clip.

(ii) *FlowSmoothTV (FlowTV)* measures spatial variation of per-frame flow (Zach et al., 2007). For a T-frame window, let $\mathbf{f}_{t,i,j} = (u_{t,i,j}, v_{t,i,j}) = \hat{\mathbf{p}}_{t,i,j} - \hat{\mathbf{p}}_{t-1,i,j}$ be the predicted pixel displacement. For grid spacing s, define $\Delta_x \mathbf{f}_{t,i,j} = (\mathbf{f}_{t,i,j+1} - \mathbf{f}_{t,i,j})/s$ and Δ_y analogously. We compute

$$\mathrm{FlowTV} = \frac{1}{|\mathcal{I}_{\mathrm{TV}}|} \sum\nolimits_{t \in \mathcal{I}_{\mathrm{TV}}} \sum\nolimits_{a \in \{x,y\}} \mathbb{E}^a_{i,j} \left[\|\Delta_a \mathbf{f}_{t,i,j}\|_1 \right], \tag{15}$$

where the directional means use valid neighboring flows, with empty directions contributing zero; $\mathcal{I}_{\mathrm{TV}}$ contains intervals with at least one valid pair.

(iii) *DivCurlEnergy (DivCurlE)* measures expansion/contraction and rotation (Corpetti et al., 2006):

$$\mathrm{DivCurlE} = \frac{1}{|\mathcal{I}_{\mathrm{DC}}|} \sum\nolimits_{t \in \mathcal{I}_{\mathrm{DC}}} \mathbb{E}_{i,j} \left[\mathrm{div}(\mathbf{f}_t)^2_{i,j} + \mathrm{curl}(\mathbf{f}_t)^2_{i,j} \right], \tag{16}$$

Here $\mathbb{E}_{i,j}$ averages over valid cells and $\mathcal{I}_{\mathrm{DC}}$ contains intervals with at least one such cell. Appendix A.5 specifies operators and masks. These coherence diagnostics do not establish prediction accuracy.

(iv) *Visibility-masked Endpoint Error (VEPE).* For ground-truth and reconstructed pixel positions $\mathbf{p}_{t,n}, \hat{\mathbf{p}}_{t,n} \in \mathbb{R}^2$ and visibility $v_{t,n} \in \{0, 1\}$, we report

$$\mathrm{VEPE} = \frac{\sum_{t,n} v_{t,n} \|\hat{\mathbf{p}}_{t,n} - \mathbf{p}_{t,n}\|_2}{\sum_{t,n} v_{t,n}}. \tag{17}$$

4.2 Quantitative Results.

Future-trajectory generation. Table 1 compares our generator and WHN (L) on real data (Kinetics, RoboTAP), synthetic data (Kubric), and MagicData (E). Kubric uses the same MOVi-A setup as WHN (L); we re-render the dataset for direct comparison. Our method consistently improves motion quality, reducing FVMD by 2.5–3.6× (e.g., 4872 to 1338 on Kubric). It also lowers FlowTV and DivCurlE, indicating fewer spatial discontinuities and more stable motion.

Track-VAE reconstruction. Table 2 reports trajectory VAE reconstruction fidelity. TrajLoom-VAE achieves low VEPE and outperforms WHN (L)-VAE by a large margin on all datasets. Performance remains similar from 24 to 81 frames, indicating stable reconstruction across segment lengths.

4.3 Qualitative Results

Figure 5 compares future trajectories across all benchmark sources. Conditioning on observed motion history produces futures that better continue current dynamics and remain spatially coherent, whereas WHN (L) more often shows drift and spatial tearing. More examples are provided in Appendix E.1.

4.4 Ablation Study

Table 1 shows the effect of Grid-Anchor Offset Encoding for 81-frame future generation. Removing offsets degrades long-horizon motion, increasing FVMD-Long and worsening FlowTV and DivCurlE, indicating more drift and tearing. We also ablate TrajLoom-VAE offset encoding and the spatiotemporal consistency regularizer, both of which improve reconstruction (Appendix F.3). Ablations of TrajLoom-Flow boundary hints are in Appendix G.2.

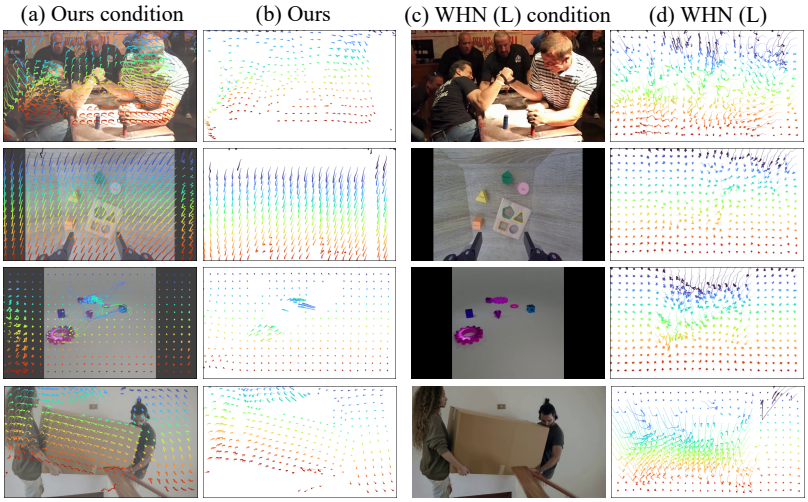


Figure 5: Comparison with WHN (L). Each row shows a dataset: Kinetics, RoboTAP, Kubric, and MagicData (E). Our method yields smoother and more coherent motion.



Figure 6: Trajectory-guided video generation with Wan-Move (Chu et al., 2025). Wan-Move uses an image and trajectories predicted by TrajLoom-Flow to generate an 81-frame video.

4.5 Downstream Applications

Predicted future trajectories can serve as motion-control signals for video generation with Wan-Move (Chu et al., 2025). As shown in Figure 6, we use observed history trajectories to generate future trajectory fields with TrajLoom-Flow, then feed them to Wan-Move with a single input image to synthesize motion-consistent videos. This demonstrates that trajectory generation enables controllable motion continuation in downstream video generation.

5 Conclusion

In this work, we present a framework for dense future-trajectory generation from observed history. Our approach combines Grid-Anchor Offset Encoding, a consistency-regularized TrajLoom-VAE, and the rectified-flow generator TrajLoom-Flow with explicit boundary cues and on-policy fine-tuning. Together, these components produce stable long-horizon motion and outperform state-of-the-art methods on TrajLoomBench in both quantitative and qualitative evaluations. For future work, we plan to improve trajectory controllability (*e.g.*, user-driven trajectory editing) and further integrate our model with motion-guided generation and editing methods to improve versatility and accuracy.

AI USE STATEMENT

We used generative AI tools to assist with manuscript editing and formatting, literature search, and code development and debugging. The authors reviewed the AI-assisted material and take full responsibility for the methods, results, and final manuscript.

ETHICS STATEMENT

This work models possible future motion as point trajectories for video understanding and generation. We encourage responsible downstream use that respects privacy and accounts for prediction uncertainty.

REPRODUCIBILITY STATEMENT

Implementation, training, and evaluation details are provided in Appendix A. Code, pretrained models, configurations, and data-preparation scripts will be made publicly available with the paper to support reproducibility.

REFERENCES

Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision*, pp. 306–324. Springer, 2024.

Gabrijel Boduljak, Laurynas Karazija, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. What happens next? anticipating future motion by generating point trajectories. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=t1vMYl1yhe.

Ryan Burgert, Yuancheng Xu, Wenqi Xian, Oliver Pilarski, Pascal Clausen, Mingming He, Li Ma, Yitong Deng, Lingxiao Li, Mohsen Mousavi, et al. Go-with-the-flow: Motion-controllable video diffusion models using real-time warped noise. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 13–23, 2025.

João Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Ruihang Chu, Yefei He, Zhekai Chen, Shiwei Zhang, Xiaogang Xu, Bin Xia, Dingdong WANG, Hongwei Yi, Xihui Liu, Hengshuang Zhao, Yu Liu, Yingya Zhang, and Yujiu Yang. Wan-move: Motion-controllable video generation via latent trajectory guidance. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=lHW93LKaUk.

Thomas Corpetti, Dominique Heitz, Georges Arroyo, Etienne Mémin, and Alina Santa-Cruz. Fluid experimental flow estimation based on an optical-flow scheme. *Experiments in fluids*, 40(1):80–97, 2006.

Yufan Deng, Ruida Wang, Yuhao Zhang, Yu-Wing Tai, and Chi-Keung Tang. Dragvideo: Interactive drag-style video editing. In *European conference on computer vision*, pp. 183–199. Springer, 2024.

Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens, Lucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems*, 35:13610–13626, 2022.

Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10061–10072, 2023.

Carl Doersch, Pauline Luc, Yi Yang, Dilara Gokay, Skanda Koppula, Ankush Gupta, Joseph Heyward, Ignacio Rocco, Ross Goroshin, Joao Carreira, et al. Bootstap: Bootstrapped training for tracking-any-point. In *Proceedings of the Asian Conference on Computer Vision*, pp. 3257–3274, 2024.

John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

Daniel Geng, Charles Herrmann, Junhwa Hur, Forrester Cole, Serena Zhang, Tobias Pfaff, Tatiana Lopez-Guevara, Yusuf Aytar, Michael Rubinstein, Chen Sun, et al. Motion prompting: Controlling video generation with motion trajectories. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 1–12, 2025.

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3749–3761, 2022.

Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *European Conference on Computer Vision*, pp. 59–75. Springer, 2022.

Adam W. Harley, Yang You, Xinglong Sun, Yang Zheng, Nikhil Raghuraman, Yunqi Gu, Sheldon Liang, Wen-Hsuan Chu, Achal Dave, Pavel Tokmakov, Suya You, Rares Ambrus, Katerina Fragkiadaki, and Leonidas J. Guibas. AllTracker: Efficient dense point tracking at high resolution. In *ICCV*, 2025.

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518. Springer, 1992.

Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6013–6022, 2025.

Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017. URL http://arxiv.org/abs/1705.06950.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Quanhao Li, Zhen Xing, Rui Wang, Hui Zhang, Qi Dai, and Zuxuan Wu. Magicmotion: Controllable video generation with dense-to-sparse trajectory guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12112–12123, October 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Jiahe Liu, Youran Qu, Qi Yan, Xiaohui Zeng, Lele Wang, and Renjie Liao. Fréchet video motion distance: A metric for evaluating motion consistency in videos. *arXiv preprint arXiv:2407.16124*, 2024.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

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.

Koichi Namekata, Sherwin Bahmani, Ziyi Wu, Yash Kant, Igor Gilitschenski, and David B. Lindell. SG-i2v: Self-guided trajectory control in image-to-video generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=uQjySppU9x.

Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your gan: Interactive point-based manipulation on the generative image manifold. In *ACM SIGGRAPH 2023 conference proceedings*, pp. 1–11, 2023.

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

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Novin Shahroudi, Mihkel Lepson, and Meelis Kull. Evaluation of trajectory distribution predictions with energy score. In *Forty-first International Conference on Machine Learning*, 2024.

Yujun Shi, Chuhui Xue, Jun Hao Liew, Jiachun Pan, Hanshu Yan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8839–8849, 2024.

Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, 27, 2014.

Nick Stracke, Kolja Bauer, Stefan Andreas Baumann, Miguel Ángel Bautista, Josh Susskind, and Björn Ommer. Learning long-term motion embeddings for efficient kinematics generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 42581–42591, June 2026.

Neerja Thakkar, Shiry Ginosar, Jacob C Walker, Jitendra Malik, João Carreira, and Carl Doersch. Forecasting motion in the wild. In *European Conference on Computer Vision (ECCV)*, 2026.

Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=CD9Snc73AW. Expert Certification.

Mel Vecerik, Carl Doersch, Yi Yang, Todor Davchev, Yusuf Aytar, Guangyao Zhou, Raia Hadsell, Lourdes Agapito, and Jon Scholz. Robotap: Tracking arbitrary points for few-shot visual imitation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5397–5403. IEEE, 2024.

Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *European conference on computer vision*, pp. 835–851. Springer, 2016.

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.

Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19795–19806, 2023a.

Xiang Wang, Hangjie Yuan, Shiwei Zhang, Dayou Chen, Jiuniu Wang, Yingya Zhang, Yujun Shen, Deli Zhao, and Jingren Zhou. Videocomposer: Compositional video synthesis with motion controllability. *Advances in Neural Information Processing Systems*, 36:7594–7611, 2023b.

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*, pp. 1–11, 2024.

Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning, 2023.

Jiange Yang, Haoyi Zhu, Yating Wang, Gangshan Wu, Tong He, and Limin Wang. Tra-moe: Learning trajectory prediction model from multiple domains for adaptive policy conditioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6960–6970, 2025.

Shengming Yin, Chenfei Wu, Jian Liang, Jie Shi, Houqiang Li, Gong Ming, and Nan Duan. Dragnuwa: Fine-grained control in video generation by integrating text, image, and trajectory. *arXiv preprint arXiv:2308.08089*, 2023.

Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint pattern recognition symposium*, pp. 214–223. Springer, 2007.

Jianing Zhang, Chenhao Zheng, Yajun Yang, Max Argus, Rustin Soraki, Winson Han, Taira Anderson, Chun-Liang Li, Shuo Liu, Jiafei Duan, Zhongzheng Ren, Jieyu Zhang, and Ranjay Krishna. Molmomotion: Forecasting point trajectories in 3d with language instruction, 2026a. URL https://arxiv.org/abs/2606.18558.

Yanran Zhang, Ziyi Wang, Wenzhao Zheng, Zheng Zhu, Jie Zhou, and Jiwen Lu. Geometry-aware single-image 4d synthesis via dense trajectory generation. In *European Conference on Computer Vision*, pp. 95–114. Springer, 2026b.

Zewei Zhang, Huan Liu, Jun Chen, and Xiangyu Xu. Gooddrag: Towards good practices for drag editing with diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Zhenghao Zhang, Junchao Liao, Menghao Li, Zuozhuo Dai, Bingxue Qiu, Siyu Zhu, Long Qin, and Weizhi Wang. Tora: Trajectory-oriented diffusion transformer for video generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 2063–2073, 2025b.

Yang Zheng, Adam W Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19855–19865, 2023.

APPENDIX CONTENTS

Section		*Page*
A	**Model, Training, and Evaluation Details**	15
A.1	Data Preprocessing and Benchmark Construction	15
A.2	TrajLoom-VAE	15
A.3	TrajLoom-Flow	15
A.4	On-Policy K-Step Fine-Tuning	16
A.5	Metric Implementation Details	16
B	**Additional Baseline Comparison**	17
C	**Computational Efficiency**	19
C.1	Runtime and Peak Memory	19
C.2	Sampler Choice and Generation Quality	19
D	**Robustness to Noisy Trajectory History**	20
E	**Qualitative Evaluation and Downstream Applications**	20
E.1	Additional Qualitative Comparisons	20
E.2	User Study	20
E.3	Downstream Video Generation	22
F	**Trajectory Representation and Reconstruction Analysis**	23
F.1	Location-Bias Analysis	23
F.2	Role of Spatiotemporal Consistency Regularization	23
F.3	TrajLoom-VAE Ablations	23
F.4	VAE Training Dynamics	24
G	**TrajLoom-Flow Details and Ablations**	25
G.1	Boundary Hints	25
G.2	TrajLoom-Flow Ablations	25
G.3	Visibility Prediction	26
H	**Limitations and Failure Cases**	27

A Model, Training, and Evaluation Details

Backbone. Following WHN (Boduljak et al., 2026), both TrajLoom-VAE and TrajLoom-Flow use a DiT-style transformer (Peebles & Xie, 2023) with the Latte design (Ma et al., 2025). We operate on trajectory fields at 480×832 with patch size 32, yielding $15\times26{=}390$ spatial tokens per frame.

A.1 Data Preprocessing and Benchmark Construction

Training data. Training uses MagicData (Li et al., 2025), a motion-focused text–video dataset with about 23k video–caption pairs. We apply standard filtering by aspect ratio, resolution, and clip length for consistency. Following WAN (Wan et al., 2025), videos are processed at 480p, and clips shorter than 162 frames are removed. After filtering, 16k videos remain; we use the first 162 frames of each video to match our forecasting window. We split these samples 90%/10% for training and validation.

Real-world evaluation sources. TAP-Vid evaluation sources (Doersch et al., 2022) are reconstructed from raw videos. Specifically, TAP-Vid-Kinetics is constructed from YouTube IDs and temporal segments from the Kinetics-700 validation set (Kay et al., 2017; Carreira et al., 2019), and RoboTAP (Vecerik et al., 2024) is provided in the same point-track annotation format as other TAP-Vid-style datasets. Instead of using TAP-Vid resized videos, we re-extract all videos at the target resolution and convert them into fixed-length temporal windows to match the forecasting horizon.

Synthetic evaluation source. In its original work, WHN (L) (Boduljak et al., 2026) is trained and evaluated on a Kubric variant (MOVi-A) (Greff et al., 2022), enabling direct comparison on this dataset. Kubric is also a primary synthetic source in TAP-Vid (Doersch et al., 2022). For comparability, we follow the same MOVi-A configuration as WHN and re-render longer videos when needed.

Camera-Motion Caption Augmentation. To enrich text conditioning with explicit camera cues, we augment MagicData captions with short camera-motion phrases estimated from the extracted point tracks. For each clip, we robustly estimate global camera translation from track displacements and further decompose residual motion into radial (zoom) and tangential (roll) components around the image center; remaining residual jitter is summarized as a handheld/shake score. We then map these statistics to a small vocabulary of camera primitives (e.g., pan/tilt, zoom in/out, static/handheld) with coarse speed buckets, and append the resulting phrase to the original caption during training.

A.2 TrajLoom-VAE

Architecture. TrajLoom-VAE encodes 81-frame trajectory segments (2D offsets; Section 3.1) into a compact spatiotemporal latent tensor with $C{=}16$ channels. We use a Latte encoder/decoder with 16 blocks each, 8 attention heads, and hidden size 512. A temporal convolutional compression module with stride 4 reduces the temporal length from 81 to 21 latent steps.

Objective. We train TrajLoom-VAE with a visibility-masked Huber reconstruction loss, a KL penalty with $\beta = 5\times10^{-5}$, and the spatiotemporal consistency regularizer from Section 3.2. The regularizer weights are $\lambda_{\text{temporal}} = 0.1$ and $\lambda_{\text{spatial}} = 0.2$. For the multiscale spatial term, $\mathcal{S}$ contains horizontal/vertical offsets with hop lengths $h_\delta = \|\delta\|_1 \in \{1, 2, 4\}$. Their weights are $\alpha_\delta = 1/h_\delta$, giving $1, 0.5, 0.25$.

Optimization. We optimize with AdamW (Kingma & Ba, 2015; Loshchilov & Hutter, 2019) using a learning rate of 2×10^{-5}, $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay of 0, batch size 4, and gradient accumulation 2 on 2 RTX PRO 6000 (effective batch size 16), using bf16 mixed precision and clipping gradients to keep the norm at 1.0, training for a total of 30k steps.

A.3 TrajLoom-Flow

Architecture and conditioning. TrajLoom-Flow predicts future TrajLoom-VAE latents with a Latte (Ma et al., 2025) generator of 16 blocks, 12 heads, and hidden size 768 (matching WHN (L)

scale). In our setting, $\mathbf{c}$ includes history trajectory latents $\mathbf{z}^p$, tokenized history visibility, history-video features, and a pooled text embedding (Raffel et al., 2020).

Latent normalization. Before training TrajLoom-Flow, we compute per-channel mean and std statistics of TrajLoom-VAE latents on the training set and apply channel-wise normalization. The generator is trained in the normalized latent space; decoding uses the inverse transform.

Flow-matching pretraining. We train with rectified flow matching in latent space (Section 3.3.2) using tube noise $\sigma = 0.05$. We sample flow time $t \in (0, 1)$ from a mixture: with probability 0.2, $t \sim \mathrm{Uniform}(0, 0.1)$; otherwise, t is the sigmoid of a standard normal sample, clamped to $[10^{-5}, 1 - 10^{-5}]$ (Polyak et al., 2024; Boduljak et al., 2026). We use visibility-weighted MSE with a base weight of 0.01 for fully invisible tokens before normalization, retaining weak supervision rather than excluding them entirely. The effective weights $\mathbf{m}^{\mathrm{tok}}$ include this adjustment and are normalized to obtain w_Λ as described in Section 3.3.2.

Source distribution (z_0). We initialize the first future latent slice from the last history latent plus Gaussian noise of standard deviation 0.1. The remaining future slices use independent standard Gaussian noise, as detailed in Appendix G.1.

Optimization. We optimize with AdamW (weight decay 0) at a learning rate of 6×10^{-5}, batch size 32 on 4 H100 with an effective batch size of 128, bf16 mixed precision, and gradient clipping at 1.0. We enable checkpointing for activation to reduce memory usage. And we trained 100k steps in total.

A.4 On-Policy K-Step Fine-Tuning

To reduce the train–test mismatch from ODE sampling (Section 3.3.3), we fine-tune TrajLoom-Flow with the on-policy K-step rollout objective. We use $K = 8$ Euler rollout steps on a logit-spaced time grid in $[10^{-5}, 1 - 10^{-5}]$ and floor the endpoint-target denominators at 10^{-3}. The rollout loss uses weights $w_1 = 1.0$ (toward $\mathbf{z}_1$), $w_0 = 0.5$ (toward $\mathbf{z}_0$), and endpoint-consistency weight γ as 0.1. For the endpoint-consistency term in Eq. 13, we use uniform weights $w_\Lambda(k, n) = 1/|\Lambda|$. Under the weighted-norm definition, this gives an unmasked mean over all tokens and channels. This choice applies only to L_{cons}; L_{fm} and $L_{\text{k-step}}$ retain their visibility-based weights. We apply the rollout loss on a small sub-batch of 8 samples per iteration, with overall weight $\lambda_{\text{k-step}} = 0.1$. And fine-tune using learning rate 1×10^{-5} while keeping other settings unchanged.

A.5 Metric Implementation Details

We specify the validity masks, discrete operators, and averaging conventions for Eqs. 15–17.

Valid positions, flows, and stencils. For a T-frame evaluated window, let $q_{t,i,j} \in \{0, 1\}$ indicate that the predicted position is finite and passes the visibility mask used for evaluation. When visibility masking is disabled, only finiteness is required. Define

$$\mathbf{f}_{t,i,j} = (u_{t,i,j}, v_{t,i,j}) = \hat{\mathbf{p}}_{t,i,j} - \hat{\mathbf{p}}_{t-1,i,j}, \qquad r_{t,i,j} = q_{t,i,j} q_{t-1,i,j},$$

for $t = 1, \ldots, T - 1$. A valid flow therefore requires valid positions at both times. The directional masks are

$$m^x_{t,i,j} = r_{t,i,j} r_{t,i,j+1}, \qquad m^y_{t,i,j} = r_{t,i,j} r_{t,i+1,j}.$$

For divergence and curl, use $m^c_{t,i,j} = m^x_{t,i,j} m^y_{t,i,j}$ on the common interior grid. This stencil requires the origin, right neighbor, and lower neighbor, not the diagonally opposite corner.

FlowTV. With grid spacing s in pixels, define

$$\Delta_x u_{t,i,j} = \frac{u_{t,i,j+1} - u_{t,i,j}}{s}, \qquad \Delta_y u_{t,i,j} = \frac{u_{t,i+1,j} - u_{t,i,j}}{s},$$

and likewise for v. Let $N_a(t) = \sum_{i,j} m^a_{t,i,j}$ for $a \in \{x, y\}$. For $N_a(t) > 0$, the directional mean is

$$\mathrm{TV}_a(t) = \frac{1}{N_a(t)} \sum_{(i,j):m^a_{t,i,j}=1} \left(|\Delta_a u_{t,i,j}| + |\Delta_a v_{t,i,j}|\right).$$

Set $\mathrm{TV}_a(t) = 0$ when $N_a(t) = 0$. The notation $\mathbb{E}^a_{i,j}$ in Eq. 15 denotes this directional mean, with zero contribution for an empty direction. Since $\|\Delta_a \mathbf{f}\|_1 = |\Delta_a u| + |\Delta_a v|$, the vector and component-wise expressions are equivalent. Horizontal and vertical pairs are averaged separately, not pooled. Define the supported intervals by

$$\mathcal{I}_{\mathrm{TV}} = \{t \in \{1, \dots, T-1\} : N_x(t) + N_y(t) > 0\}.$$

Then

$$\mathrm{FlowTV} = \frac{1}{|\mathcal{I}_{\mathrm{TV}}|} \sum_{t \in \mathcal{I}_{\mathrm{TV}}} \left(\mathrm{TV}_x(t) + \mathrm{TV}_y(t)\right).$$

An interval with no valid pairs is omitted from the temporal mean.

DivCurlE. The differences above already include the grid-spacing normalization. On the common interior grid, define

$$\mathrm{div}(\mathbf{f}_t)_{i,j} = \Delta_x u_{t,i,j} + \Delta_y v_{t,i,j}, \qquad \mathrm{curl}(\mathbf{f}_t)_{i,j} = \Delta_x v_{t,i,j} - \Delta_y u_{t,i,j}.$$

There is no additional division by s in these operators. Let $N_c(t) = \sum_{i,j} m^c_{t,i,j}$ and $\mathcal{I}_{\mathrm{DC}} = \{t \in \{1, \dots, T-1\} : N_c(t) > 0\}$. We compute

$$\mathrm{DivCurlE} = \frac{1}{|\mathcal{I}_{\mathrm{DC}}|} \sum_{t \in \mathcal{I}_{\mathrm{DC}}} \frac{1}{N_c(t)} \sum_{(i,j): m^c_{t,i,j}=1} \left[\mathrm{div}(\mathbf{f}_t)^2_{i,j} + \mathrm{curl}(\mathbf{f}_t)^2_{i,j}\right].$$

Thus $\mathbb{E}_{i,j}$ in Eq. 16 averages over valid cells, and intervals without valid cells are omitted. For either diagnostic, the temporal denominator reduces to $T-1$ when every interval has support. An entirely unsupported video has an undefined score and is excluded from that diagnostic's dataset mean.

Dataset aggregation and VEPE. FlowTV and DivCurlE are averaged over videos with defined scores, with equal weight per video. VEPE instead pools finite, visibility-valid point-error sums and counts across videos; it is not an unweighted mean of per-video VEPE values. All sums in Eq. 17 are restricted to finite errors within the evaluated frame range. If there are no valid point errors, VEPE is undefined.

Interpretation. FlowTV measures spatial variation of flow, while DivCurlE measures its divergence/curl energy. Neither compares predicted positions with the reference future. Spatially constant flow can have zero spatial derivatives even when its direction is incorrect, and real motion can contain expansion or rotation. These diagnostics therefore complement distributional and prediction-error metrics; lower values alone do not establish accurate or realistic motion.

B Additional Baseline Comparison

We extend the comparison with WHN (L) by including the regression-based trajectory predictor Tra-MoE (Yang et al., 2025). This experiment complements the motion-distribution and spatial-coherence results in the main paper with prediction-error metrics and a sample-based assessment of the predicted distribution.

Baselines and conditioning. We evaluate WHN (L) (Boduljak et al., 2026) in its single-image-conditioned setting. For Tra-MoE, we use the authors' official released trajectory-prediction checkpoint. Tra-MoE predicts trajectories for queried image points under image and instruction conditioning. TrajLoom additionally uses explicit trajectory history, its visibility, and history-video features, together with a text caption. This history-conditioned formulation allows the model to use observed dynamics when generating future motion, rather than infer those dynamics from a still image alone.

Evaluation horizons and sampling. The main comparison with WHN (L) uses its 24-frame generation setting. To accommodate Tra-MoE, this additional experiment evaluates all three methods over a common 16-frame prediction window on Kinetics, RoboTAP, Kubric, and MagicData. Tra-MoE is evaluated in its native 16-frame/32-point setting and provides one deterministic prediction ($K = 1$), while WHN (L) and TrajLoom each use $K = 8$ generated samples per condition. Table 3

reports the results. Beyond these shorter-window comparisons, TrajLoom generates complete 81-frame futures jointly, without chaining shorter predictions. The evaluated WHN (L) configuration does not natively provide this generation window. The 81-frame VAE results in the main paper concern reconstruction of supplied trajectories, not future generation by the WHN generator.

Temporal alignment and evaluation masks. Using one-based frame numbering, TrajLoom observes frames 1–81, and all methods are evaluated on future windows starting at frame 82. The 16-, 24-, and 81-frame settings therefore cover frames 82–97, 82–105, and 82–162, respectively. All methods use the same AllTracker-derived reference visibility mask for evaluation, rather than their model-predicted visibility.

Prediction accuracy. Following the aggregation conventions of WHN (Boduljak et al., 2026), MSE-MeanT first averages the predicted trajectories across samples and then evaluates the mean prediction, whereas MSE-Min reports the lowest MSE among the sampled predictions. The former assesses the mean forecast; the latter measures the accuracy of the best available hypothesis. For the single-prediction Tra-MoE baseline, these two aggregations coincide. Because MSE-Min evaluates only the best sample, we also report a distributional score that accounts for the other generated hypotheses.

Distributional quality and spatial coherence. We report the Energy Score (ES) (Shahroudi et al., 2024) for WHN (L) and TrajLoom using the same sample count. ES accounts for both agreement with the reference future and dispersion among predictions, complementing the MSE metrics when evaluating uncertain future motion. We also report FVMD and the FlowTV/DivCurlE diagnostics defined in Section 4.1.1. FVMD compares motion-feature distributions, whereas FlowTV and DivCurlE measure spatial variation of predicted flow. The latter are interpreted alongside prediction-error and distributional scores, rather than as accuracy measures on their own. ES is not reported for the single-prediction Tra-MoE baseline in this comparison, and FVMD is not reported for its sparse output.

Table 3: Additional baseline comparison on 16-frame future prediction. WHN (L) and our method use $K = 8$ samples per condition; Tra-MoE uses one deterministic prediction. Lower is better for every metric. FlowTV and DivCurlE are displayed after multiplication by 10^2 and 10^3, respectively. Unreported entries are identified explicitly.

Metric	Method	Kinetics	RoboTAP	Kubric	MagicData
MSE-MeanT ↓	WHN (L)	12740.01	2838.34	1389.47	9871.38
	Tra-MoE	13767.96	2889.05	1233.92	10032.76
	Ours	**563.48**	**350.38**	**38.25**	**301.76**
MSE-Min ↓	WHN (L)	12492.03	2747.29	1359.12	9661.41
	Tra-MoE	13767.96	2889.05	1233.92	10032.76
	Ours	**486.07**	**303.72**	**25.86**	**261.34**
ES ↓	WHN (L)	108.41	60.35	46.12	102.68
	Tra-MoE	Not reported (single prediction)			
	Ours	**20.69**	**17.49**	**6.62**	**14.13**
FVMD ↓	WHN (L)	13870	11728	7869	14627
	Tra-MoE	Not reported (sparse output)			
	Ours	**4119**	**3315**	**1410**	**3372**
FlowTV (10^2) ↓	WHN (L)	25.56	26.85	13.30	27.29
	Tra-MoE	23.00	27.33	17.00	24.76
	Ours	**7.76**	**6.32**	**6.51**	**7.42**
DivCurlE (10^3) ↓	WHN (L)	37.10	37.02	11.48	41.52
	Tra-MoE	37.71	43.04	16.87	43.51
	Ours	**4.37**	**2.18**	**2.26**	**3.20**

Results. Under this evaluation setting, TrajLoom achieves the lowest reported MSE-MeanT and MSE-Min on all four datasets. For example, on Kinetics, MSE-MeanT decreases from 12740.01 for

WHN (L) to 563.48 for our method; on Kubric, it decreases from 1233.92 for Tra-MoE to 38.25. TrajLoom also obtains lower ES than WHN (L) on every dataset, showing that its advantage is not limited to selecting the best sample. Lower FVMD provides complementary evidence of closer motion-feature distributions, while lower FlowTV and DivCurlE indicate reduced spatial variation of the predicted flow. Together, these results show improvements in prediction-error and distributional metrics alongside the spatial-coherence diagnostics.

C Computational Efficiency

C.1 Runtime and Peak Memory

Sampling an 81-frame future-trajectory window takes **1.07 s** under the reported benchmarking configuration. We use a single RTX PRO 6000 GPU, Euler integration with 10 steps, FP16 precision, and batch size 1, with **25.23 GB** peak GPU memory.

C.2 Sampler Choice and Generation Quality

We sample TrajLoom-Flow by integrating the rectified-flow ODE in trajectory latent space. Our default sampler uses Euler integration with 10 steps, matching the default WHN setting (Boduljak et al., 2026). To examine sensitivity to the sampler, we additionally evaluate both WHN (L) and our method with the Dormand–Prince solver (DOPRI5) (Dormand & Prince, 1980) under the DOPRI5-100 setting, examining higher-order ODE sampling (Tong et al., 2024). Table 4 reports this comparison on *24-frame* future prediction, separately from the 81-frame runtime measurement above.

Table 4: Effect of sampler choice on 24-frame future prediction. Unmarked rows use Euler-10; starred rows use DOPRI5-100. Lower is better for all metrics. FlowTV and DivCurlE are displayed with factors of 10^2 and 10^3, respectively. Bold numbers indicate the lowest value in each metric–dataset group.

Metric	Method	Kinetics	RoboTAP	Kubric	MagicData (E)
FVMD ↓	WHN (L)	8999	7587	4872	10383
	WHN (L)*	8755	5855	6284	9691
	Ours	3626	**2467**	**1338**	2968
	Ours*	**3182**	2546	1724	**2358**
FlowTV (10^2) ↓	WHN (L)	24.71	27.26	12.56	26.74
	WHN (L)*	22.06	14.46	18.41	14.46
	Ours	7.53	6.03	6.03	7.11
	Ours*	**4.50**	**3.40**	**3.09**	**3.72**
DivCurlE (10^3) ↓	WHN (L)	34.69	37.39	9.89	40.04
	WHN (L)*	28.52	14.41	17.58	14.41
	Ours	3.97	2.03	2.09	2.99
	Ours*	**2.86**	**1.17**	**1.10**	**1.59**

Strong quality with Euler-10. Our default Euler-10 results have lower FVMD, FlowTV, and DivCurlE than WHN (L) with either sampler on all four datasets. In particular, our Euler-10 FVMD on Kubric is 1338, compared with 6284 for WHN (L) under DOPRI5-100. The quality advantage therefore persists without switching our model to the alternative sampler.

Effect of the alternative sampler. For our model, DOPRI5-100 lowers FlowTV and DivCurlE on every dataset, but its effect on FVMD is mixed: FVMD improves on Kinetics and MagicData, while Euler-10 has lower FVMD on RoboTAP and Kubric. Thus, lower spatial-variation scores do not necessarily imply a closer motion-feature distribution. These results support Euler-10 as our default: it provides strong quality across the reported metrics together with the low sampling latency measured above.

Practical implication. TrajLoom-Flow predicts the future window jointly in trajectory latent space, producing an explicit motion representation before future RGB frames are synthesized. This separates obtaining a candidate motion plan from rendering it as a video: predicted trajectories can be inspected before a downstream video generator is invoked. The Wan-Move examples (Chu et al., 2025) in Appendix E.3 demonstrate how these trajectories can subsequently guide video synthesis. The runtime reported here characterizes the trajectory-generation benchmark, not the end-to-end Wan-Move pipeline.

D Robustness to Noisy Trajectory History

Our generator conditions explicitly on observed trajectory history. We therefore examine its sensitivity to perturbations of these input trajectories, comparing the original history with a noisy-history condition for 81-frame future prediction.

Perturbation protocol. For the noisy condition, we add independent $\mathcal{N}(\mathbf{0}, 4^2\mathbf{I}_2)$ noise in pixel coordinates to each visible history point at each frame. Each coordinate thus has a noise standard deviation of 4 pixels, with perturbations sampled independently across points and frames. The original condition uses the extracted history without this additional perturbation. Table 5 reports FVMD-Long, FlowTV, and DivCurlE using the definitions in Section 4.1.1.

Table 5: Robustness to noisy trajectory history on 81-frame future prediction. Original uses the extracted history without additional noise; Noisy adds independent $\mathcal{N}(\mathbf{0}, 4^2\mathbf{I}_2)$ pixel noise to each visible history point per frame. The displayed FVMD-Long, FlowTV, and DivCurlE values are multiplied by 10^{-3}, 10^2, and 10^3, respectively, as in the main comparison.

Metric	History	Kinetics	RoboTAP	Kubric	MagicData
FVMD-Long (10^{-3}) ↓	Original	35.06	21.75	7.24	28.27
	Noisy	35.26	24.15	10.86	29.39
FlowTV (10^2) ↓	Original	7.65	5.96	5.03	7.11
	Noisy	7.68	5.97	5.06	7.08
DivCurlE (10^3) ↓	Original	4.21	1.89	1.45	3.09
	Noisy	4.34	1.90	1.47	2.92

Results. The spatial-coherence diagnostics remain close to their original-history values. Based on the reported scores, the absolute relative changes are below 0.6% for FlowTV and 6% for DivCurlE on every dataset. FVMD-Long is more sensitive: it increases by approximately 0.6%, 11.0%, 50.0%, and 4.0% on Kinetics, RoboTAP, Kubric, and MagicData, respectively. In particular, Kubric increases from 7.24 to 10.86, despite little change in its flow diagnostics. These results support robustness of the spatial-coherence measures to the tested positional jitter, while showing dataset-dependent degradation in the generated motion distribution. The experiment assesses this Gaussian perturbation, rather than other tracking-error patterns such as persistent drift or missing tracks.

E Qualitative Evaluation and Downstream Applications

E.1 Additional Qualitative Comparisons

Figure 7 shows additional side-by-side comparisons with WHN (L). Each row corresponds to a different clip. Panels (a) and (b) show our history condition and predicted future, while panels (c) and (d) show WHN (L)'s image condition and predicted future. Our results more consistently preserve long-horizon coherence, with fewer drift and spatial-tearing artifacts in these examples.

E.2 User Study

Evaluation protocol. We complement the quantitative evaluation with a pairwise preference study of perceived motion realism. Participants compared video pairs from our method and WHN (L),

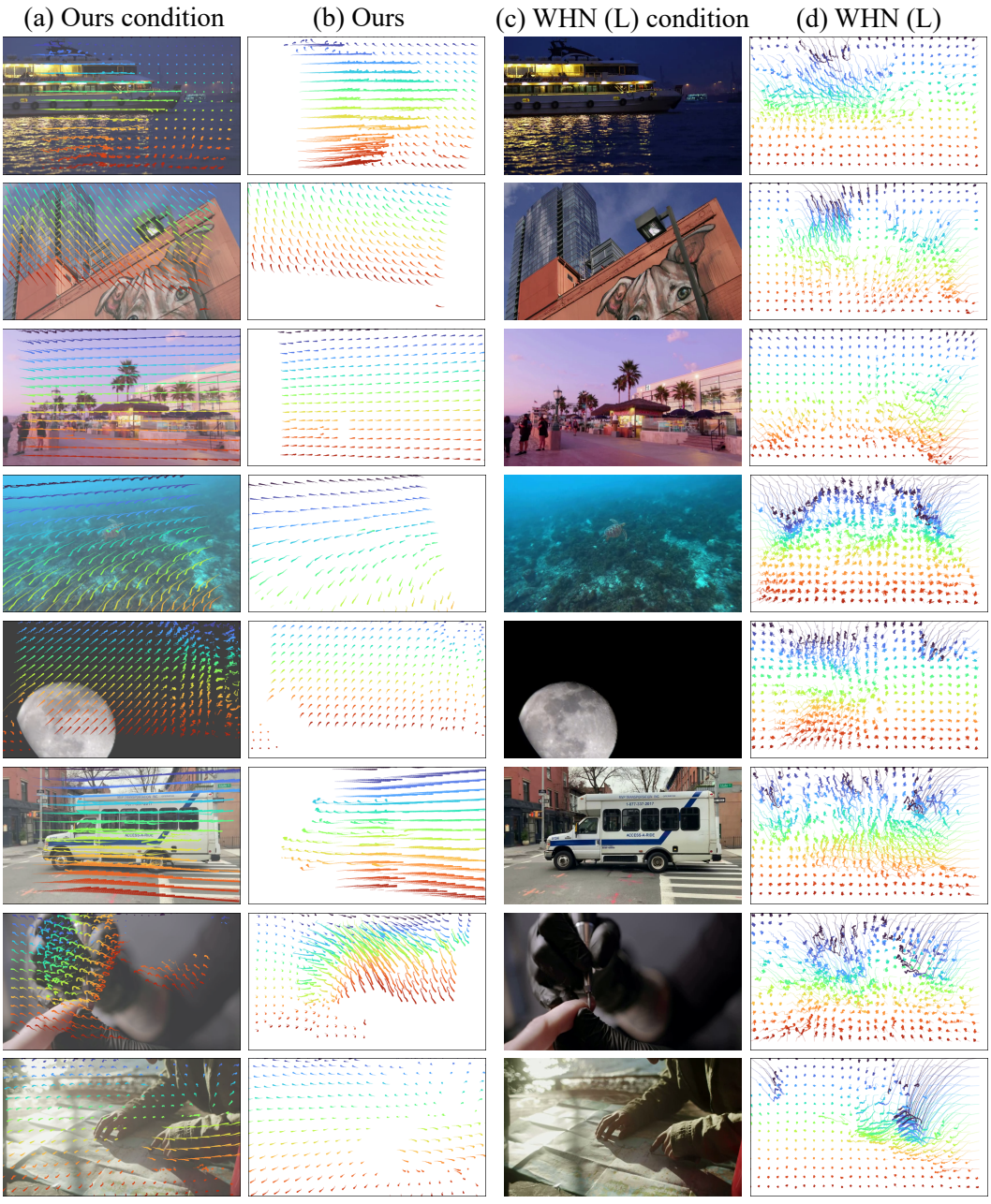


Figure 7: Additional trajectory comparisons with WHN (L). Each row shows a different clip. Panels (a)–(b) show our history condition and predicted future; panels (c)–(d) show WHN (L)'s image condition and predicted future.

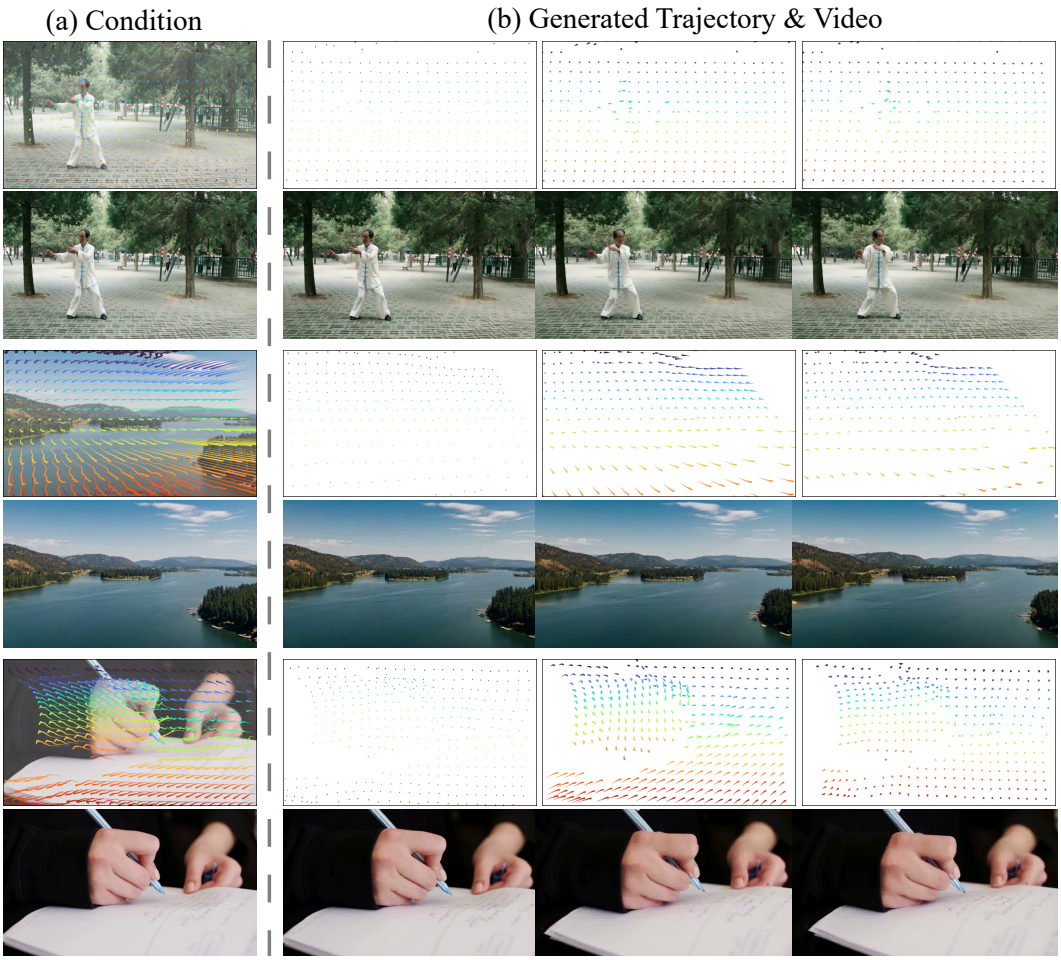


Figure 8: Additional Wan-Move results driven by our predicted trajectories. Left: conditioning inputs (history trajectory and the conditioning image). Right: our generated 81-frame future trajectories and the corresponding 81-frame videos synthesized by Wan-Move.

using the clips illustrated in Figure 7. For each pair, they judged which output exhibited more realistic motion.

Results. The study collected 248 responses from 31 participants. TrajLoom was selected in **90.7% of responses**. This response-level preference supports the perceived motion quality of our predictions on the evaluated clips and complements the quantitative results. It measures perceived realism rather than agreement with the reference future.

E.3 Downstream Video Generation

We further demonstrate downstream controllable video synthesis using Wan-Move (Chu et al., 2025). Given the observed history, we first sample 81-frame future trajectories with TrajLoom-Flow. We then provide Wan-Move with the predicted trajectories and the conditioning image to generate an 81-frame video that follows the generated motion. Figure 8 shows additional examples.

F TRAJECTORY REPRESENTATION AND RECONSTRUCTION ANALYSIS

F.1 LOCATION-BIAS ANALYSIS

This appendix provides details for the model-free analysis in Figure 3b. The goal is to measure how much of the coordinate variance is explained purely by the *grid location* (a static spatial baseline), compared to the remaining variance that comes from *temporal motion*.

We use dense point tracks extracted by AllTracker (Harley et al., 2025) on a stride-s grid (here $s = 32$) at resolution $H \times W = 480 \times 832$. Each track corresponds to a fixed grid cell $n \in \{1, \dots, N\}$, and provides per-frame 2D coordinates and visibility. We convert pixel coordinates to normalized coordinates in $[-1, 1]$ and form offsets by subtracting the (normalized) center anchor of the corresponding stride cell, i.e. $\mathbf{X}_{n,t} = \mathbf{D}_{n,t} - \mathbf{G}_n$. We evaluate on 128 randomly sampled clips.

For a single coordinate axis d (either D_x, D_y, X_x, or X_y), let $d_{n,t}$ be the value at grid location n and time t, with visibility $v_{n,t} \in \{0, 1\}$. For each location n, we compute a visibility-weighted time average $\mu_n = \frac{\sum_t v_{n,t}\, d_{n,t}}{\sum_t v_{n,t}}$ as the *location baseline*, and a visibility-weighted temporal variance around that baseline $\sigma_n^2 = \frac{\sum_t v_{n,t}\,(d_{n,t}-\mu_n)^2}{\sum_t v_{n,t}}$.

We then decompose the total coordinate variance into a between-location term and a within-location term. Let $\mathrm{Var}_n(\cdot)$ and $\mathbb{E}_n[\cdot]$ denote variance and mean over grid locations n. We define the fraction of variance explained by grid location as:

$$\mathrm{Var}(d) = \mathrm{Var}_n(\mu_n) + \mathbb{E}_n\left[\sigma_n^2\right], \quad \text{Explained}(\%) = 100 \cdot \frac{\mathrm{Var}_n(\mu_n)}{\mathrm{Var}_n(\mu_n) + \mathbb{E}_n[\sigma_n^2]}. \tag{18}$$

Here, $\mathrm{Var}(d)$ denotes the total variance under this decomposition: $\mathrm{Var}_n(\mu_n)$ measures how much coordinates differ across grid locations due to the static spatial baseline, while $\mathbb{E}_n[\sigma_n^2]$ measures the average temporal variance (motion) at a fixed location. A high explained percentage indicates that most coordinate variation is attributable to grid position rather than motion. As shown in Figure 3b, absolute coordinates have high explained variance, while offsets strongly reduce this location-driven component.

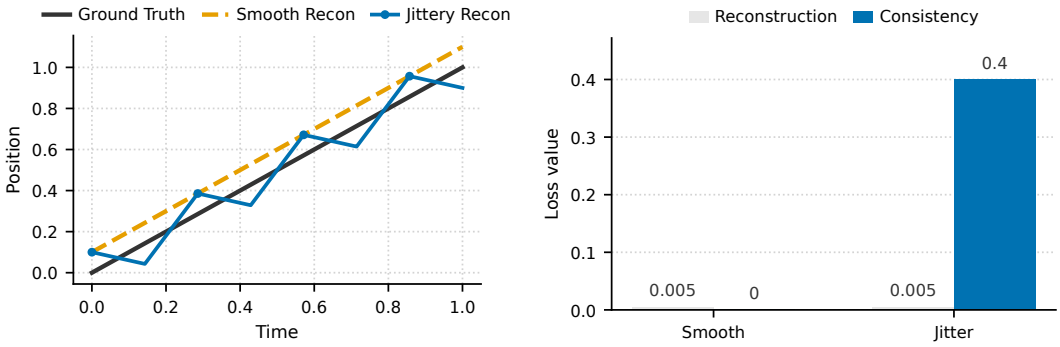
F.2 ROLE OF SPATIOTEMPORAL CONSISTENCY REGULARIZATION

In this section, we explain why pointwise reconstruction loss alone is insufficient and motivate the need for spatiotemporal consistency regularization. Pointwise reconstruction alone does not uniquely determine a physically plausible motion trajectory. To illustrate this, we consider a 1D trajectory with full visibility and ground truth $x(t) = t$ for discrete time t. We compare two reconstructions that have the same per-frame deviation magnitude: a *smooth* reconstruction with a constant bias, $\hat{x}_{\text{smooth}}(t) = t + b$, and a *jittery* reconstruction with an alternating bias, $\hat{x}_{\text{jitter}}(t) = t + b(-1)^t$. Both reconstructions have identical pointwise error $|\hat{x}(t) - x(t)| = |b|$ at every frame, and therefore can achieve the same masked Huber reconstruction loss L_{rec} (for any Huber threshold that treats these deviations similarly).

However, the two reconstructions imply very different *motion*. The ground-truth velocity is constant, $\Delta x(t) = x(t) - x(t-1) = 1$. The smooth reconstruction preserves this, $\Delta\hat{x}_{\text{smooth}}(t) = 1$, while the jittery reconstruction alternates its frame-to-frame displacement, $\Delta\hat{x}_{\text{jitter}}(t) = 1 + 2b(-1)^t$. As a result, the velocity-matching term penalizes the jittery solution even though L_{rec} cannot distinguish it. The same issue generalizes to dense 2D trajectory fields: reconstructions can match per-frame positions yet exhibit temporal jitter and locally inconsistent motion across neighboring points. Our spatiotemporal consistency regularizer directly targets these failure modes by matching temporal displacements (L_{temporal}) and neighbor relations (L_{spatial}), as shown in Figure 9.

F.3 TRAJLOOM-VAE ABLATIONS

We ablate two components of TrajLoom-VAE: (i) the Grid-Anchor Offset Encoding and (ii) the spatiotemporal consistency regularizer. Table 6 reports VEPE (pixels; lower is better) for reconstructing 24- and 81-frame trajectory segments. Removing offset encoding consistently increases reconstruction error across datasets and horizons, confirming that offsets reduce location-dependent bias and



(a) Ground truth, smooth, and jittery.

(b) Same L_{rec}, different L_{st}.

Figure 9: 1D toy example: pointwise reconstruction loss does not guarantee realistic motion. Both smooth and jittery reconstructions of $x(t) = t$ can match L_{rec}, but only the smooth one aligns with ground truth motion. Consistency regularization separates realistic from unrealistic solutions.

Table 6: TrajLoom-VAE ablations. Effect of Grid-Anchor Offset Encoding and spatiotemporal consistency regularization on VEPE (↓).

Frames	Method	Kinetics	RoboTAP	Kubric	MagicData (E)
24	w/o offset	4.24	3.98	3.26	3.91
	w/o regularizer	2.99	2.35	1.42	2.65
	Ours	**1.99**	**1.59**	**0.75**	**1.72**
81	w/o offset	4.30	4.01	3.24	3.99
	w/o regularizer	3.11	2.41	1.47	3.99
	Ours	**2.04**	**1.61**	**0.77**	**1.72**

simplify trajectory modeling. Removing the consistency regularizer also degrades VEPE, especially on longer windows, consistent with the regularizer suppressing temporal jitter and improving local coherence in the reconstructed motion.

F.4 VAE Training Dynamics

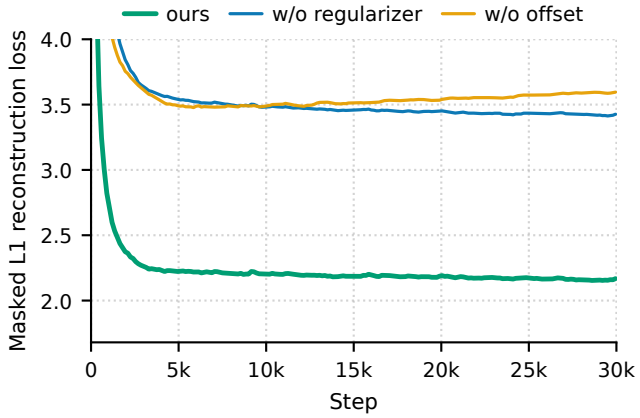


Figure 10: TrajLoom-VAE training curves. Masked L_1 reconstruction metric vs. training step. Using Grid-Anchor Offset Encoding and the spatiotemporal regularizer yields faster, more stable convergence and a substantially lower loss than removing offsets or the regularizer.

Figure 10 plots the TrajLoom-VAE masked L_1 reconstruction loss for our full model and two ablations. Removing the Grid-Anchor Offset Encoding leads to worse convergence and a gradually increasing loss, consistent with overfitting to location-dependent coordinate biases in absolute space. In contrast, removing the spatiotemporal consistency regularizer yields a higher-loss plateau, indicating underfitting: pointwise reconstruction alone does not sufficiently constrain motion structure. Together, these trends support our design: offset encoding makes the learning problem better conditioned by removing global-position bias, while the regularizer provides motion-level constraints that help the latent space capture temporally smooth and locally coherent trajectories.

G TrajLoom-Flow Details and Ablations

G.1 Boundary Hints

This appendix specifies the two *boundary hints* used by TrajLoom-Flow (Section 3.3.1): (i) a boundary-anchored source state $\mathbf{z}_0$ for rectified-flow sampling, and (ii) a token-aligned fusion term that injects boundary information from $\mathbf{z}^p$ into the model input.

Boundary-anchored source state $\mathbf{z}_0$. Let $\mathbf{z}(-1, n)$ denote the last history latent slice. For a future slice selected for boundary anchoring, we use

$$\mathbf{z}_0(k, n) = \mathbf{z}(-1, n) + \sigma_0 \, \boldsymbol{\eta}(k, n), \qquad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{19}$$

In our implementation, this rule is applied only at $k = 0$, with $\sigma_0 = 0.1$. All remaining future slices are initialized independently from standard Gaussian noise.

Token-aligned history fusion. In addition to conditioning through $\mathbf{c}$, we add a lightweight *additive* boundary cue to the model's input tokens. At each flow time t, we form token embeddings from $\mathbf{z}_t$ and add an aligned history term:

$$\mathrm{Tok}(\mathbf{z}_t)(k, n) \leftarrow \mathrm{Tok}(\mathbf{z}_t)(k, n) + \alpha \, g_k \Big(\mathbf{b}(n) + \omega_k \, \mathbf{d}(n) \Big), \tag{20}$$

where $\mathrm{Tok}(\cdot)$ denotes the query-input projection, α is a learnable scalar, and $g_k \in (0, 1)$ is a learnable per-step gate when enabled ($g_k = 1$ otherwise). Let $\mathbf{h}^p(k, n)$ denote the separately projected history token, including the history-visibility embedding when enabled. Indices -1 and -2 refer to the last and second-last history latent steps, respectively. The boundary feature $\mathbf{b}(n)$ and velocity hint $\mathbf{d}(n)$ use these steps at the *same* spatial token n:

$$\mathbf{b}(n) = \mathbf{h}^p(-1, n), \qquad \mathbf{d}(n) = \mathbf{h}^p(-1, n) - \mathbf{h}^p(-2, n). \tag{21}$$

The coefficient ω_k increases linearly from 0 at the first future latent step to 1 at the last.

G.2 TrajLoom-Flow Ablations

This appendix provides additional ablations on TrajLoom-Flow components described in Section 3.3.1, including boundary hints and on-policy fine-tuning. All results are computed on 81-frame future prediction using Euler integration with 10 steps, and report the same motion and flow-field diagnostics as in the main paper.

We ablate:

1. **History fusion:** removing the token-aligned history fusion module, so history is provided only through the conditioning stream $\mathbf{c}$.
2. **Boundary anchoring:** disabling first-slice boundary anchoring, i.e. not adding $\mathbf{z}(-1, n)$ to $\mathbf{z}_0(0, n)$.
3. **On-policy fine-tuning:** training with flow matching only (Section 3.3.2), *i.e.* setting $\lambda_{\mathrm{k-step}} = 0$ and omitting the on-policy K-step fine-tuning stage.

In addition, Table 7 reports an early flow-matching checkpoint (20k steps) and a full-budget model trained without on-policy fine-tuning (w/o on-policy) to separate the effect of training duration from the proposed on-policy stage. Overall, longer flow-matching training improves motion quality, but on-policy fine-tuning consistently provides further gains across datasets. Among the boundary-hint variants, disabling first-slice boundary anchoring (w/o anchoring) causes the largest degradation, while removing token-aligned history fusion (w/o fusion) yields a smaller but consistent drop.

Table 7: Ablation study with flow-field diagnostics (generator components). Metrics are computed on 81-frame future prediction. Lower is better (↓). All results use Euler integration with 10 steps. **w/o fusion** removes the token-aligned history fusion module; **w/o anchoring** disables boundary anchoring of the first future latent slice ; **20k steps** is an early flow-matching checkpoint of the generator; **w/o on-policy** matches the full flow-matching training budget of our method but omits the on-policy K-step fine-tuning stage.

Metric	Setting	Kinetics	RoboTAP	Kubric	MagicData (E)
FVMD-Long (10^{-3}) ↓	w/o fusion	39.60	28.86	16.38	33.87
	w/o anchoring	43.50	33.12	20.29	37.11
	20 k steps	40.13	28.81	17.02	33.51
	w/o on-policy	38.87	27.38	13.86	31.71
	Ours	**35.06**	**21.75**	**7.24**	**28.27**
FlowTV (10^2) ↓	w/o fusion	9.66	8.66	9.56	8.50
	w/o anchoring	10.07	9.11	9.80	8.90
	20 k steps	9.99	9.09	9.99	8.94
	w/o on-policy	8.98	8.22	8.90	8.16
	Ours	**7.65**	**5.96**	**5.03**	**7.11**
DivCurlE (10^3) ↓	w/o fusion	7.29	5.49	11.24	5.04
	w/o anchoring	7.83	5.85	11.71	5.53
	20 k steps	8.09	5.79	11.44	5.63
	w/o on-policy	6.11	4.58	8.12	4.41
	Ours	**4.21**	**1.89**	**1.45**	**3.09**

G.3 Visibility Prediction

As an auxiliary component, we predict token-level visibility for the generated future motion, since points may become occluded or leave the image. Given future latent tokens $\mathbf{z}^f \in \mathbb{R}^{T\times N\times C}$, we train a lightweight predictor f_ω that outputs per-token visibility logits $\boldsymbol{\ell} \in \mathbb{R}^{T\times N}$.

The predictor is applied independently to each spatial token. For each token, we first apply a linear projection to the C-dimensional features, then use a short stack of 1D temporal convolutions along the latent-time axis, and finally apply a 1×1 projection to produce one logit per latent time step. For supervision, we downsample the dense future visibility mask $\mathbf{M}^f$ to the TrajLoom-VAE token grid using max-pooling (equivalently, a logical OR) to obtain token-level targets. We optimize a binary cross-entropy loss with logits.

At inference time, we run f_ω on generated future latents $\hat{\mathbf{z}}^f$ and threshold $\sigma(\boldsymbol{\ell})$ to obtain predicted visibility $\hat{\mathbf{m}}^f$. This visibility head is lightweight and independent of the trajectory generator.

H LIMITATIONS AND FAILURE CASES

Although our model can generate spatially coherent 2D trajectories, it can miss plausible future motion involving hidden 3D dynamics. Figure 11 illustrates such a case: the observed object moves leftward and away, and a plausible continuation could involve out-of-plane motion driven by physical forces. The generated trajectories instead give a smooth image-plane continuation that does not capture this behavior. This example highlights the distinction between spatial coherence and physical plausibility. Incorporating explicit 3D information or physical priors is a potential direction for addressing this limitation.

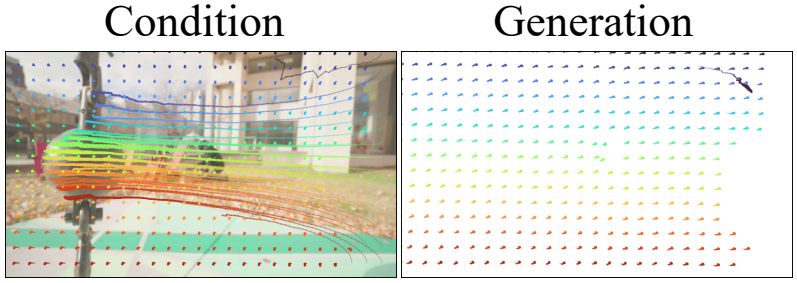


Figure 11: Illustrative failure case. Left: observed condition. Right: generated continuation. Smooth image-plane trajectories can miss the plausible out-of-plane dynamics discussed in the text.