Generative Movement of Object Trajectories in Videos

Kiran Chhatre[1,2], Hyeonho Jeong[2], Yulia Gryaditskaya[2], Christopher E. Peters[1], Chun-Hao Paul Huang[2], and Paul Guerrero[2]

[1] KTH Royal Institute of Technology
[2] Adobe Research

Abstract. Generative video editing has enabled several intuitive editing operations for short video clips that would previously have been difficult to achieve, especially for non-expert editors. Existing methods focus on prescribing an object's 3D or 2D motion trajectory in a video, or on altering the appearance of an object or a scene, while preserving both the video's plausibility and identity. Yet a method to move an object's 3D motion trajectory in a video, i.e. moving an object while preserving its relative 3D motion, is currently still missing. The main challenge lies in obtaining paired video data for this scenario. Previous methods typically rely on clever data generation approaches to construct plausible paired data from unpaired videos, but this approach fails if one of the videos in a pair can not easily be constructed from the other. Instead, we introduce TrajectoryAtlas, a new data generation pipeline for large-scale synthetic paired video data and a video generator TrajectoryMover fine-tuned with this data. We show that this successfully enables generative movement of object trajectories. Project page: chhatrekiran.github.io/trajectorymover

Keywords: Video editing · Trajectory movement · Synthetic data

1 Introduction

Moving the trajectory of an object in a video is a fundamental video editing task. For example, an editor may want to move the original trajectory of a basketball to enter the hoop or to bounce off the rim. Traditionally, changing an existing video requires re-shoots that may be expensive, time-consuming, or sometimes even impossible.

Recent advances in generative video editing have resulted in several methods that allow specifying an exact 2D [39] or 3D object trajectory [14]. However, while this detailed control is sometimes desirable, it has several drawbacks. First, manually specifying a full object trajectory in 2D or even 3D, such as a smooth parabola for a thrown object, is a time-consuming task that requires expertise. Second, it puts the responsibility for creating a trajectory that is plausible for the given scene on the shoulders of the user, rather than letting the prior of the generative model ensure its plausibility. This can be a difficult task, especially if the trajectory should capture interactions of the object with a scene: for example, what trajectory is realistic for a basketball bouncing off a rim?

Instead, we propose a much simpler fundamental task: can we translate the trajectory of an object in a video, while keeping its motion and scene interactions plausible? This affords a much simpler editing experience, where the user can simply drag a moving object in a video frame to a new position. We introduce TrajectoryMover, a generative model that implements this task. Given a source video, the user annotates the first frame with two bounding boxes: a source bounding box around the object that is to be moved, and a target bounding box defining the target location. Our approach then generates a video where the object trajectory is translated to start at the target bounding box, while also updating it where necessary to maintain plausible scene interactions. Figure 4 shows an example of an object no longer hitting the ground after moving the trajectory to pass through a hole in the ground.

The main challenge in training such a model is the lack of paired data for this task. To address this challenge, we introduce a synthetic data generation pipeline that automatically places new objects into existing scenes and can synthesize a variety of plausible trajectories for these objects. A physics simulation ensures plausible interaction between the objects and the scene. To ensure sufficient free space in the scene to perform more complex object motions, we introduce an optional online scene modification approach that adapts a scene for a given video pair by removing objects that would block object motion. Given this paired data, we formulate trajectory movement as video-to-video generation task where the bounding box control is encoded as an additional frame of the input video. We carefully fine-tune a video generator [38] while preserving its original prior by alternating between a pure generation task on real videos and our trajectory movement task on our paired data.

We show in extensive experiments that TrajectoryMover can move trajectories more accurately and plausibly than existing methods that require a more explicit specification of the moved trajectory.

In summary, we propose the following contributions:

- We identify a lack of simple trajectory editing tools and propose trajectory movement as simple and fundamental tool to edit trajectories. We show that this approach addresses video editing tasks that are challenging with current tools.
- To address the lack of paired data, we introduce a synthetic data generation pipeline that generates video pairs that share the same content, except for one object with moved trajectory.
- We propose to formulate trajectory movement as a video-to-video generation task and carefully fine-tune a video generator without losing its original prior.

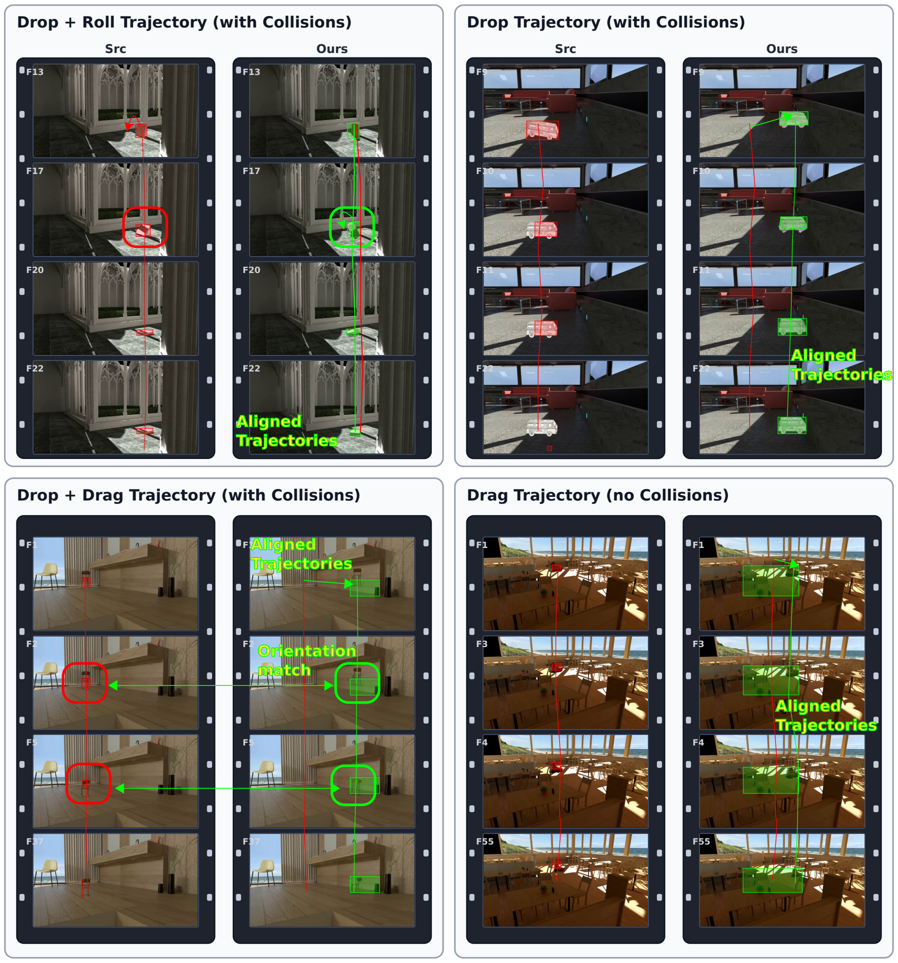

Drop + Roll Trajectory (with Collisions)
Src
Ours
F13
F17
F20
F22
Aligned Trajectories
Drop Trajectory (with Collisions)
Aligned Trajectories
Drop + Drag Trajectory (with Collisions)
Aligned Trajectories
Orientation match
Drag Trajectory (no Collisions)
F1
F3
F4
F55
Aligned Trajectories

Fig. 1. TrajectoryMover enables intuitive video editing by allowing users to translate an object's 3D motion path to a new starting location using simple bounding box controls across diverse and complex scenarios, including drop, roll, and drag motions. Our model successfully aligns the generated trajectory with the target initial location. Furthermore, the model dynamically adapts the motion to the new path to ensure physical plausibility, seamlessly handling novel scene interactions such as realistic collisions with the environment.

2 Related Work

2.1 Conditional Video Generation and Editing

Conditional video diffusion models extend base text- or image-conditioned architectures by incorporating auxiliary control signals. Inspired by the success of ControlNet [42] on controllable image generation, several approaches have introduced ControlNet-style hypernetworks to video synthesis [15,7,18,33,4,19,31]. These methods adapt temporal mechanisms to enable guidance via diverse visual signals, such as depth maps, edge maps, and camera parameters. Alternatively, other frameworks utilize source video input to facilitate video-to-video editing. These approaches generally address tasks such as targeted appearance editing [19,21,1,31], global stylization [40,25], and novel view synthesis [2,16,32,17,41]. To incorporate source guidance, many existing methods concatenate clean source video tokens with noisy target video tokens, either along the channel [1,41] or token dimension [2,21,41,24]. Following this paradigm, TrajectoryMover integrates source video input by concatenating source video tokens with noisy target video tokens.

2.2 Motion-controlled Video Generation and Editing

Recent generative video models have increasingly prioritized motion control, offering users precise manipulation of object and camera trajectories. Existing approaches generally fall into two paradigms: trajectory-based control and optical flow-based motion transfer. Trajectory-based methods [4,24,34,12,37] condition generation on sparse point tracks, enabling precise manipulation of object paths and complex interactions. Conversely, optical flow-based methods [9,23] leverage dense correspondence priors to achieve detailed motion transfer.

While these works have demonstrated impressive capabilities in video synthesis, they are primarily designed as conditional video generators rather than video editors. Many prominent image-to-video (I2V) models utilizing trajectory conditioning rely on a single reference frame [12,34,33,39], causing them to neglect the temporal context of the input video and lose the original scene information when motion is modified.

In this work, we address the complex task of shifting an object's 3D motion trajectory within a source video while strictly preserving its original relative 3D dynamics. The primary obstacle to achieving such motion-preserving trajectory movement is the scarcity of paired video data suitable for supervised training. We address this by introducing TrajectoryAtlas, a data generation pipeline for large-scale synthetic data of paired videos specifically designed to provide the ground-truth supervision necessary for trajectory manipulation. By fine-tuning a video generator on this diverse corpus, we achieve precise, generative control over object paths.

3 Method

Our goal in this work is to generatively move the trajectory of an object in a video. Conceptually, this requires removing the object from its original trajectory in the video and inserting it with a new trajectory, while 1) preserving the object's identity, 2) ensuring that the object follows the new trajectory, and 3) making sure the resulting video remains plausible, including interactions between the object and the containing scene. Note that 2) and 3) may be at odds – for example, the moved trajectory of a thrown ball may pass through a wall. In this case, the object motion should remain plausible, for example, we would expect the ball to bounce off the wall. As control signal, a user provides two bounding boxes (see Figure 1). The first bounding box selects an object in the first frame and the second describes its target position in the same frame.

To achieve this, we propose a new synthetic data pipeline capable of generating a large video dataset that is aligned to this task, consisting of instructive videos pairs that would be hard to find in any current real-world or synthetic datasets: video pairs that show the same video, except for one object with moved trajectory (Sec. 3.1). We call this pipeline *TrajectoryAtlas*. Data generated by TrajectoryAtlas is used to fine-tune an existing diffusion-based video generator [38] *TrajectoryMover* that has a strong prior distribution over plausible videos (Sec. 3.2). To make sure the generator does not forget its prior, we use interleaved training that switches between unconditional generation of real videos and our task with synthetic videos. As conditioning strategy, we follow [2] in treating the source video, the conditioning signal, and the target video as a single sequence, keeping tokens for the source video and conditioning signal fixed during generation.

3.1 Data Generation Pipeline

Assume we are given a source video A with N_F frames where a foreground object follows a 3D trajectory $X = (x_i)_{i=1}^{N_F}$ with per-frame center positions $x_i \in \mathbb{R}^3$. We want to create a video B where the same object instead follows a 3D trajectory $Y = (y_i)_{i=1}^{N_F}$ that is offset from X by δ: $y_i = x_i + \delta$, unless this would result in an implausible motion, like passing through walls. Our data generation pipeline creates pairs (A, B) that can be used to fine-tune a video generator. Additionally, we create binary segmentation videos (M_A, M_B) for each video corresponding to the moved object; these can be used to create the bounding boxes of the object in videos A and B that we need for our control signal. We call this dataset Trajectory Atlas, and its data generation pipeline is shown in Fig. 2.

Scenes and objects. Our pipeline takes as input a set of synthetic 3D scenes $\{S_i\}_{i=1}^{N_S}$ with a set of valid camera poses $\{C_i\}_{i=1}^{N_S}$, $C_i = c_1, c_2, \ldots$ for each scene. Additionally, we assume a set of 3D objects $\{O_i\}_{i=1}^{N_O}$ to be inserted into the scenes is provided. To create a video pair (A, B), we start by randomly choosing a scene S_i, a camera pose c_j in the scene, and an object O_k.

Object trajectories. In addition to scenes and objects, our pipeline defines a set of object trajectory types that the object should attempt to follow in both videos A and B. Assuming an initial placement x_0 of the object in the first frame:

- `Drop`: x_0 places the object in the air and the object drops down.
- `Throw`: x_0 places the object in the air with an initial velocity that points away from the camera, resulting in an arced trajectory.

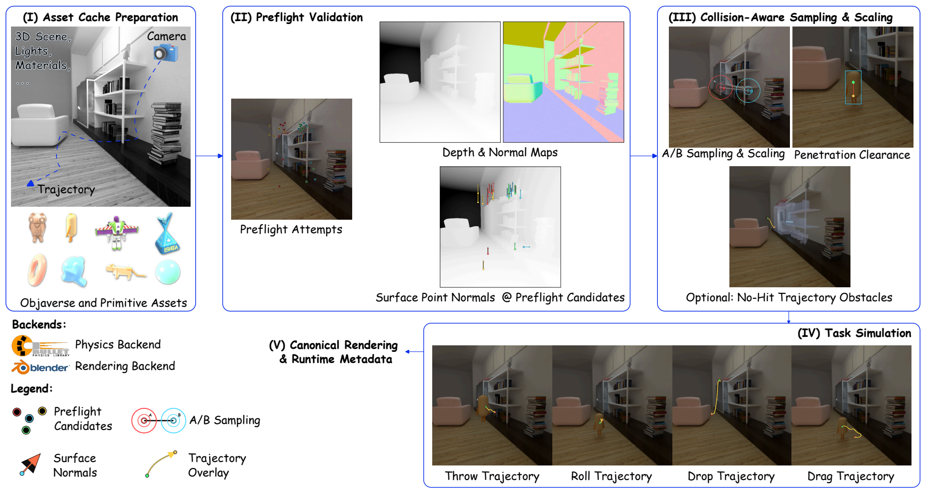


Fig. 2. TrajectoryAtlas data generation pipeline. The pipeline has five stages, Asset Cache Preparation, Preflight Validation, Collision Aware Sampling and Scaling, Task Simulation, and Canonical Rendering with Runtime Metadata. Inputs including camera, 3D scene, lights and materials, and Objaverse or primitive assets are converted to reusable collision caches, then skip render preflight selects valid frames. Paired A/B placements with shared scale are filtered by visibility, support normal, and penetration clearance, and optional no hit processing removes only non structural obstacles in the trajectory corridor. Throw, drop, roll, and drag trajectories are simulated with Bullet and rendered with Blender into canonical RGB and binary segmentation videos. Please zoom in for details.

- `Roll`: x_0 places the object on a supporting surface with an initial velocity that points away from the camera, resulting in a rolling motion.
- `Drag`: x_0 places the object on a supporting surface with and an elastic force that drags the object along a pre-defined path (see supplement for details).
- `Static`: x_0 places the object on a supporting surface and the object does not move.

All forces and paths in these trajectories are defined relative to the initial position x_0. We use a robust rigid body physics simulation [10] to create the trajectories, giving us motions that plausibly interact with the scene geometry. Note that we do not aim to teach the video generator about physics or plausible object motions with these trajectories, as this knowledge is already available in the video prior. The trajectories just need to be diverse enough to make the task that the video generator should perform clear.

First-frame object placement. The initial positions x_0 and y_0 of the object in the source- and target videos need to be chosen such that (i) the object is clearly visible in the first frame, (ii) does not implausibly intersect any scene geometry, and (iii) allows for the chosen object trajectory type. We distinguish between two types of initial object placements: in the air and on the ground, depending on the chosen object trajectory type. For air placements, we randomly choose a pixel of the first frame, shoot a ray from the camera through the pixel, and place the object at a random point along the visible part of the ray (before it hits the scene). For ground placements, we randomly choose a pixel that correspond to a supporting surface (a pixel with an upward-pointing normal), shoot a ray through the pixel, and place the object so it rests on the point hit by the ray. In both cases, we reject and re-sample placements that result in intersections between the object and the scene.

Given the initial positions, we can define $\delta = y_0 - x_0$. How we correlate the initial object positions in the source and target videos determines the behavior of the video generator when fine-tuned on our data. For ground placements, we opt to sample the initial positions in source and target videos independently using the approach described above, as we want the user to be able to freely choose an image location in the target video. However, sampling air placements independently reduces the predictability of the fine-tuned generator, as different depths along the same pixel ray give similar bounding boxes, giving the generative model some leeway for choosing an arbitrary depth of the target object. To reduce this ambiguity, we opt to choose pixels independently, but to correlate the depth of air placements in source and target videos. Specifically, we sample the depth in the target video using a Gaussian centered at the placement in the source video, resulting in predictable same-depth trajectory offsets δ.

The scale of objects is chosen based on their screen size in the first frame. We scale objects so their screen bounding box has a maximum side length randomly chosen between 7% and 20% of the video height.

Online scene modification. Running the pipeline as described above results in video pairs with plausible object trajectories. However, due to scene clutter, in many videos only a small part of the trajectory remains similar in a video pair. As soon as the object starts to interact with scene geometry, the trajectories in a video pair are likely to diverge. To get a more instructive set of videos that more clearly demonstrate our requirement of trajectory preservation, we modify the scene in a random subset of video pairs so that larger parts of the trajectories in a video pair remain similar. Specifically, we remove any clutter in the scene that would intersect the projected trajectory in any of the two videos. Additional details are provided in the supplement.

3.2 Video Generator

Using TrajectoryAtlas (Section 3.1), we train a video-to-video model that maps a source video $V_{\text{src}} \in \mathbb{R}^{F \times C \times H \times W}$ to a target video $V_{\text{trg}} \in \mathbb{R}^{F \times C \times H \times W}$, conditioned on an object displacement control signal $I_{\text{bb}} \in \mathbb{R}^{C \times H \times W}$. Here F, C, H, and W denote the number of frames, color channels, and frame resolution. The control signal I_{bb} encodes which object to move and where to place it in the target frame.

We use Wan2.1-T2V-1.3B [38] as the DiT backbone and its VAE to map video frames V to spatio-temporal latents $z \in \mathbb{R}^{F' \times C' \times H' \times W'}$. We form three latent streams, z_{trj}, z_{src}, and z_{bb}, and concatenate them along the temporal axis before denoising. To match frame-level RoPE indexing [35], we place target frames first, from 0 to $F' - 1$, then append source frames and the control latent.

4 Implementation Details

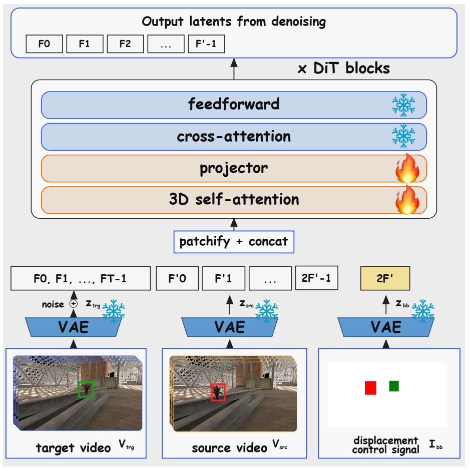


Fig. 3. TrajectoryMover architecture. We concatenate three latent streams z_{trj}, z_{src}, and z_{bb} before denoising. In the control image, red marks the source box and green marks the target box.

Data generation. TrajectoryAtlas uses Blender (Cycles) for rendering and PyBullet for physics. We use curated Evermotion [13] indoor scenes and a foreground object pool of 119 assets, with 98 Objaverse objects [11] and 21 primitives (canonical shapes plus Bullet-stable proxy variants). For each video pair, we sample initial object placements with shared scale and enforce visibility, support, and intersection checks, then run task-specific physics simulation (drop, throw, roll, drag), optionally apply online scene modification, and render the final videos. We run generation as parallel GPU jobs with one render worker per GPU and worker-level output sharding. We precompute and reuse scene and object collision meshes, and each run stores fixed seeds and full runtime configurations for reproducibility.

Training. Using bounding boxes extracted from ground-truth masks, we construct the input control signal I_{bb}. Specifically, red and green boxes specify the object's location in the initial frames of the source and target sequences. Following the native spatial and temporal resolution of the Wan2.1-T2V-1.3B backbone, each frame is resized to 832×480, and we limit sequences to the first $F{=}81$ frames, leading to $F'{=}21$ latent frames. As shown in Fig. 3, we adopt a parameter-efficient tuning strategy by training only the self-attention and projector layers while freezing the remainder of the network. To preserve the original generative prior, we employ a joint training scheme, alternating between the TrajectoryMover V2V task on our training set and the standard T2V task on a large video corpus at a 7:3 ratio. Training is conducted over 3,200 steps on an 8-H100-GPU machine with a total batch size of 16.

5 Results

We evaluate TrajectoryMover by comparing results of our trajectory movement task to several state-of-the-art video editing baselines, comparing identity preservation, trajectory adherence, and plausibility of the results.

Dataset. We generate a total of $\sim$ 21k video pairs using TrajectoryAtlas that we split into 20k training pairs and 1k test pairs. From the test pairs, we randomly choose 40 videos for evaluation. All videos have a resolution of 1280×720 and 81 frames at 16 fps. More detailed dataset information is provided in the supplement.

Baselines. While no existing baseline supports our trajectory movement task out-of-the-box, we repurpose existing video generators to our task while keeping the core methods unchanged. ATI [39] moves objects to follow given 2D trajectories. DaS [15] moves objects by editing 3D tracks of the source video. VACE [19] supports object movement through a bounding box based trajectory reference video. I2VEdit [31] propagates the first frame edits to other frames, allowing us to move the object in the first frame. SFM [27] is a 3D aware video editing method that reconstructs an editable 3D mesh and uses it to guide motion editing. To provide inputs for these methods, we estimate a 3D trajectory from the input video by tracking the estimated depth [6] of the foreground object. We then move this trajectory to start at the target bounding box and convert this moved 3D trajectory to the native format of each baseline. See the supplement for additional method-specific details. Note that unlike our method, the trajectory moved by this simple approach does not take into account scene interactions.

Metrics. We evaluate each method with three primary metrics, computed per frame, averaged per video, and then averaged across the test set: SSIM_{bg} (background preservation), DINO_{fg} (foreground identity preservation), and IoU_{traj} (trajectory adherence). For each generated video, we extract foreground masks with SAM3 [5], while ground-truth masks come from dataset segmentations. For SSIM_{bg}, we compare generated frames to GT target frames only on background pixels. Specifically, we define foreground as the union of (i) GT target mask and (ii) SAM3 predicted mask after a small dilation, then define background as its complement; masked SSIM is computed per frame and averaged over time. For DINO_{fg}, we take the first valid source-object crop (from GT source mask) as reference, encode it with `dinov2-base` [30,20], and compute cosine similarity against generated-object crops from SAM3 masks on visible frames; the metric is the mean similarity over valid frames. For IoU_{traj}, we convert SAM3 predicted masks and GT target masks into per-frame bounding boxes, compute IoU at each frame, and report the temporal mean. Together, these three metrics quantify whether a method preserves scene content, preserves object identity, and follows the intended motion path.

Table 1. Comparison to Baselines. We compare background preservation (SSIM_{bg}), foreground identity preservation (DINO_{fg}), trajectory adherence (IoU_{traj}), and user study plausibility strength (u_m) of our generated video edits against baselines on 40 test videos.

	$\text{SSIM}_{\text{bg}} \uparrow$	$\text{DINO}_{\text{fg}} \uparrow$	$\text{IoU}_{\text{traj}} \uparrow$	$u_m \uparrow$
ATI [39]	<u>0.71</u>	<u>0.39</u>	0.18	-0.27
DaS [15]	0.45	0.22	0.16	-0.36
VACE [19]	0.68	0.17	0.16	-0.48
I2VEdit [31]	0.17	0.15	0.02	-0.24
SFM [27]	0.56	0.29	<u>0.23</u>	<u>0.10</u>
TrajectoryMover (ours)	**0.92**	**0.45**	**0.27**	**1.25**

5.1 Quantitative Evaluation

Table 1 shows that TrajectoryMover achieves the best performance across all three quantitative metrics and in human plausibility. Among the baselines, ATI is strongest on background preservation and foreground identity, while SFM is strongest on trajectory adherence and user study plausibility. Compared to ATI, TrajectoryMover improves SSIM_{bg} from 0.71 to 0.92 (+0.21) and DINO_{fg} from 0.39 to 0.45 (+0.06). Compared to SFM, TrajectoryMover improves IoU_{traj} from 0.23 to 0.27 (+0.04) while also substantially improving SSIM_{bg} from 0.56 to 0.92 and DINO_{fg} from 0.29 to 0.45. The differences are even larger for DaS, VACE, and I2VEdit, especially in trajectory adherence and identity consistency. Overall, TrajectoryMover provides the strongest combined control of motion, object identity, scene consistency, and plausibility.

5.2 Qualitative Evaluation

Visualization protocol. Figure 4 and Fig. 5 follow a consistent visual convention to facilitate comparison across methods and ablations. In the source video (GT Src), red bounding boxes denote the source object's initial location, while green boxes indicate its target relocation derived from ground truth masks. Additionally, pink boxes highlight specific local regions of prominent failure modes of each method while cyan boxes highlight successful results.

Comparison to baselines. Figure 4 compares TrajectoryMover against ATI, DaS, VACE, I2VEdit, and SFM across a diverse set of complex motion types, including drop, throw, and roll sequences, both with and without scene modifications. Across all scenarios, TrajectoryMover tracks the intended trajectory more faithfully while maintaining foreground identity and background structure. The translated object remains temporally coherent, exhibiting minimal texture collapse or shape drift, even as it adapts to novel scene depths and changing support surfaces.

Among the baselines, SFM is relatively stronger at coarse trajectory following, but it often suffers from failure modes introduced by its reconstruction and edit

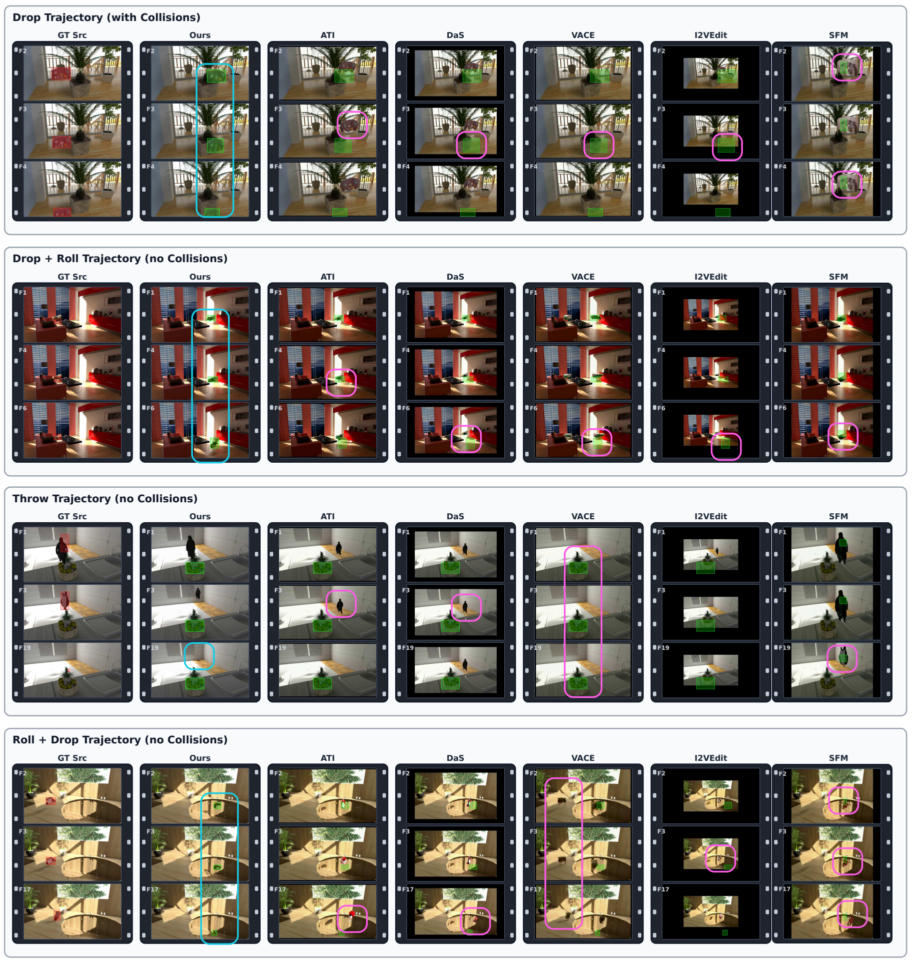

Drop Trajectory (with Collisions)
Drop + Roll Trajectory (no Collisions)
Throw Trajectory (no Collisions)
Roll + Drop Trajectory (no Collisions)
GT Src
Ours
ATI
DaS
VACE
I2VEdit
SFM

Fig. 4. Qualitative comparison with baselines. We compare TrajectoryMover with SFM, ATI, DaS, VACE, and I2VEdit on four representative motion scenarios. Red boxes indicate the source object location in the input video, green boxes indicate the target location at frame 0, pink boxes highlight regions of failure, and cyan boxes highlight regions of success. TrajectoryMover follows the intended motion most consistently while preserving object appearance and scene identity. Please zoom in for details.

pipeline, including reconstructed object fragmentation, unintended multi-instance object generation, and temporal object deformation during propagation. ATI and DaS more closely match our trajectory control setting, but their externally constrained control signals do not robustly resolve scene dependent motion changes after relocation, which leads to trajectory drift and physically implausible motion in challenging scenes. VACE can preserve appearance in simpler cases, but under our control setting it frequently shows composition conflicts such as duplicated object regions or partial re-renderings of the source object. I2VEdit performs adequately when the appearance of the first frame strongly determines subsequent motion, but it struggles with long term propagation and often produces fading, disappearance, or severe trajectory drift after a few frames.

In contrast, a key advantage of TrajectoryMover is its ability to predict inherently plausible, scene aware motion directly from the source video and the simple relocation signal, without requiring manually designed trajectory specific adjustments at test time. This advantage is particularly evident in complex scenarios where an object must translate laterally while also naturally settling at a different depth or support surface. Please zoom in on the figures for fine grained details. Additional qualitative examples, TrajectoryAtlas samples, and model result videos are provided in the supplementary material and supplementary video.

5.3 User Study

Study Design. We conduct a blind pairwise perceptual study on motion plausibility. For each test case, participants see two anonymized videos A and B from the same source target setup and choose the video with more plausible object motion. The Aand B order is randomized per item and method identities are recovered with a hidden answer key. For analysis, we estimate method strength with a regularized Bradley–Terry model using iterative Luce Spectral Ranking as implemented in `choix` [3,28,29]. The model is $P(i \succ j) = \frac{e^{u_i}}{e^{u_i}+e^{u_j}}$, where u_i is the utility of method i.

Results. The user study includes $n = 10$ participants with 25 pairwise judgments each, for a total of 250 votes. Estimated utilities are TrajectoryMover = 1.25, SFM = 0.10, I2VEdit = −0.24, ATI = −0.27, DaS = −0.36, and VACE = −0.48, giving the ranking TrajectoryMover > SFM > I2VEdit > ATI > DaS > VACE.

InternVL analysis. As a complementary automatic proxy, we also score plausibility with InternVL [8] over 10 runs (seeds 0 to 9) and report mean ± std. TrajectoryMover ranks first with 0.633 ± 0.132, followed by DaS 0.533 ± 0.233, VACE 0.515 ± 0.067, SFM 0.4954 ± 0.0995, ATI 0.400 ± 0.211, and I2VEdit 0.383 ± 0.113, giving the ranking TrajectoryMover > DaS > VACE > SFM > ATI > I2VEdit. InternVL agrees with the human study in ranking TrajectoryMover first, but differs substantially on the ordering of the baselines. We therefore treat the VLM score as supporting evidence and use human Bradley–Terry strength u_m as the primary plausibility metric.

5.4 Ablation

We ablate the effect of several key design choices in our data generation pipeline on a model trained with the resulting data. We train a model with data that has only primitives rather than a diverse set of objects from Objaverse (only

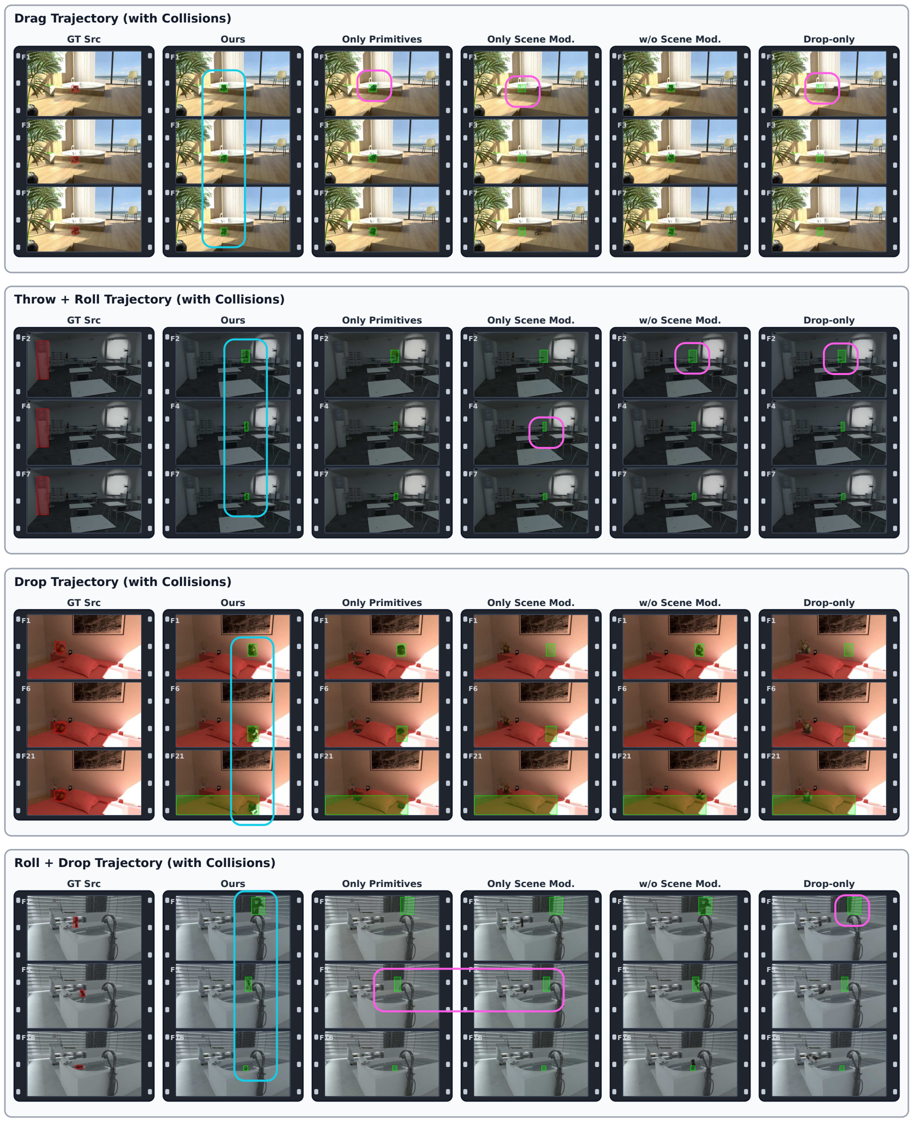

Drag Trajectory (with Collisions)
GT Src
Ours
Only Primitives
Only Scene Mod.
w/o Scene Mod.
Drop-only
Throw + Roll Trajectory (with Collisions)
Drop Trajectory (with Collisions)
Roll + Drop Trajectory (with Collisions)
F1
F2
F3
F4
F6
F7
F21

Fig. 5. Qualitative ablation analysis. We compare the full model with ablations using only primitives, only scene modification, without scene modification, and drop-only motion training. Red boxes indicate source object location, green boxes indicate target frame-0 location, and pink boxes mark representative regions of failure while cyan boxes highlight region of success results. The full model gives the best balance of trajectory fidelity, object identity preservation, and scene-aware motion plausibility. Please zoom in for details.

primitives). We ablate our choice to do online scene modifications in half of

Table 2. Ablation. We ablate several key design choices in our pipeline: only using primitives as foreground objects, performing online scene modifications for all videos, or for none of the videos, and using only a single object trajectory type.

	$\text{SSIM}_{\text{bg}} \uparrow$	$\text{DINO}_{\text{fg}} \uparrow$	$\text{IoU}_{\texttt{traj}} \uparrow$
only primitives	**0.93**	0.15	0.20
only scene mod.	<u>0.92</u>	<u>0.44</u>	0.16
w/o scene mod.	<u>0.92</u>	<u>0.44</u>	0.17
`Drop`-only	<u>0.92</u>	**0.45**	<u>0.21</u>
TrajectoryMover (full)	<u>0.92</u>	**0.45**	**0.27**

the video pairs by training on data where all video pairs have been modified (only scene mod.) or none have been modified (w/o scene mod.). Finally, we ablate the need for diverse trajectories using only the `Drop` trajectory (`Drop`-only). Table 2 shows that TrajectoryMover (full) achieves the best trajectory adherence with $\text{IoU}_{\texttt{traj}} = 0.27$, showing that all of the tested choices contribute to trajectory adherence. For object identity, only primitives is substantially worse, with $\text{DINO}_{\text{fg}} = 0.15$, while the other ablations remain close to full at 0.44 to 0.45, showing that a diverse set of objects is important to preserve object identity. Background preservation is not strongly affected by these design choices. Overall, the full model provides the best balance of trajectory adherence, object identity preservation, and background consistency.

Ablation qualitative analysis. Figure 5 shows that each data generation component contributes distinct behavior. The full model gives the strongest combined performance in trajectory fidelity, identity preservation, and scene interaction realism. Only primitives degrades identity most clearly, with blob-like shapes, weaker texture, and lower temporal consistency in difficult frames. Only scene mod. helps obstacle avoidance in no-hit settings but underexposes the model to richer interactions, often reducing placement accuracy in complex scenes. Without scene mod. improves some hit interactions but weakens learning of long-range unobstructed motion, leading to larger trajectory misalignment under changed initialization. `Drop`-only is too narrow and transfers poorly to roll and drag motions. Overall, the qualitative results support the full training recipe with mixed object sources, mixed hit and no-hit data, and diverse motion tasks. Please zoom in on the figure 5 for details.

6 Conclusion

In this work, we presented TrajectoryMover, the first framework to move an object's 3D motion trajectory in a video. To overcome the inherent lack of paired training data, we introduced TrajectoryAtlas, a synthetic data generation pipeline that leverages physics simulations and online scene modification to create plausible, diverse trajectory-shifted video pairs. By formulating this task as a video-to-video generation problem and employing a balanced fine-tuning strategy, our model successfully preserves the generative prior of the original backbone while learning complex spatial transformations.

Limitation. Although TrajectoryMover significantly outperforms existing baselines in trajectory adherence, the absolute peak performance ($\mathrm{IoU}_{\mathrm{traj}}$ of 0.27) reveals that perfectly mapping an object to a strict generative path remains a fundamentally challenging task. The model sometimes trades absolute trajectory precision for the sake of preserving object identity and background consistency. Finally, our proof-of-concept generator does currently not yet generalize to arbitrary real-world videos. We plan to refine the generator for stronger generalization with additional training.

References

1. AI, D.: Open-weight text-guided video editing (2025), https://platform.decart.ai/
2. Bai, J., Xia, M., Fu, X., Wang, X., Mu, L., Cao, J., Liu, Z., Hu, H., Bai, X., Wan, P., et al.: Recammaster: Camera-controlled generative rendering from a single video. ICCV (2025)
3. Bradley, R.A., Terry, M.E.: Rank analysis of incomplete block designs i. the method of paired comparisons. Biometrika **39**(3/4), 324–345 (1952)
4. Burgert, R., Herrmann, C., Cole, F., Ryoo, M.S., Wadhwa, N., Voynov, A., Ruiz, N.: Motionv2v: Editing motion in a video. arXiv preprint arXiv:2511.20640 (2025)
5. Carion, N., Gustafson, L., Hu, Y.T., Debnath, S., Hu, R., Suris, D., Ryali, C., Alwala, K.V., Khedr, H., Huang, A., Lei, J., Ma, T., Guo, B., Kalla, A., Marks, M., Greer, J., Wang, M., Sun, P., Rädle, R., Afouras, T., Mavroudi, E., Xu, K., Wu, T.H., Zhou, Y., Momeni, L., Hazra, R., Ding, S., Vaze, S., Porcher, F., Li, F., Li, S., Kamath, A., Cheng, H.K., Dollár, P., Ravi, N., Saenko, K., Zhang, P., Feichtenhofer, C.: Sam 3: Segment anything with concepts (2026)
6. Chen, S., Guo, H., Zhu, S., Zhang, F., Huang, Z., Feng, J., Kang, B.: Video depth anything: Consistent depth estimation for super-long videos. CVPR (2025)
7. Chen, W., Ji, Y., Wu, J., Wu, H., Xie, P., Li, J., Xia, X., Xiao, X., Lin, L.: Control-a-video: Controllable text-to-video diffusion models with motion prior and reward feedback learning. arXiv preprint arXiv:2305.13840 (2023)
8. Chen, Z., Wu, J., Wang, W., Su, W., Chen, G., Xing, S., Zhong, M., Zhang, Q., Zhu, X., Lu, L., et al.: Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In: CVPR. pp. 24185–24198 (2024)
9. Cong, Y., Xu, M., Simon, C., Chen, S., Ren, J., Xie, Y., Perez-Rua, J.M., Rosenhahn, B., Xiang, T., He, S.: Flatten: optical flow-guided attention for consistent text-to-video editing. ICLR (2023)
10. Coumans, E., Bai, Y.: Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org` (2016–2019)
11. Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A.: Objaverse: A universe of annotated 3d objects. CVPR (2023)
12. Deng, Y., Wang, R., Zhang, Y., Tai, Y.W., Tang, C.K.: Dragvideo: Interactive drag-style video editing. In: ECCV. pp. 183–199. Springer (2024)
13. Evermotion: Evermotion. `https://evermotion.org/`, accessed: 2026-03-05

14. Gu, Z., Yan, R., Lu, J., Li, P., Dou, Z., Si, C., Dong, Z., Liu, Q., Lin, C., Liu, Z., Wang, W., Liu, Y.: Diffusion as shader: 3d-aware video diffusion for versatile video generation control. ACM SIGGRAPH 2025 Conference Papers (2025)
15. Gu, Z., Yan, R., Lu, J., Li, P., Dou, Z., Si, C., Dong, Z., Liu, Q., Lin, C., Liu, Z., et al.: Diffusion as shader: 3d-aware video diffusion for versatile video generation control. In: ACM SIGGRAPH Asia 2025 Conference Papers. pp. 1–12 (2025)
16. Huang, Z., Jeong, H., Chen, X., Gryaditskaya, Y., Wang, T.Y., Lasenby, J., Huang, C.H.: Spacetimepilot: Generative rendering of dynamic scenes across space and time. arXiv preprint arXiv:2512.25075 (2025)
17. Jeong, H., Lee, S., Ye, J.C.: Reangle-a-video: 4d video generation as video-to-video translation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 11164–11175 (2025)
18. Jeong, H., Ye, J.C.: Ground-a-video: Zero-shot grounded video editing using text-to-image diffusion models. ICLR (2023)
19. Jiang, Z., Han, Z., Mao, C., Zhang, J., Pan, Y., Liu, Y.: Vace: All-in-one video creation and editing. ICCV (2025)
20. Jose, C., Moutakanni, T., Kang, D., Baldassarre, F., Darcet, T., Xu, H., Li, D., Szafraniec, M., Ramamonjisoa, M., Oquab, M., Siméoni, O., Vo, H.V., Labatut, P., Bojanowski, P.: Dinov2 meets text: A unified framework for image- and pixel-level vision-language alignment (2024)
21. Ju, X., Wang, T., Zhou, Y., Zhang, H., Liu, Q., Zhao, N., Zhang, Z., Li, Y., Cai, Y., Liu, S., et al.: Editverse: Unifying image and video editing and generation with in-context learning. arXiv preprint arXiv:2509.20360 (2025)
22. Koo, J., Guerrero, P., Huang, C.H.P., Ceylan, D., Sung, M.: Videohandles: Editing 3d object compositions in videos using video generative priors. In: CVPR (2025)
23. Koroglu, M., Caselles-Dupré, H., Jeanneret, G., Cord, M.: Onlyflow: Optical flow based motion conditioning for video diffusion models. In: CVPR. pp. 6226–6236 (2025)
24. Lee, Y.C., Zhang, Z., Huang, J., Wang, J.H., Lee, J.Y., Huang, J.B., Shechtman, E., Li, Z.: Generative video motion editing with 3d point tracks. arXiv preprint arXiv:2512.02015 (2025)
25. Li, M., Chen, J., Zhao, S., Feng, W., Tu, P., He, Q.: Dreamstyle: A unified framework for video stylization. arXiv preprint arXiv:2601.02785 (2026)
26. Liu, S., Wang, T., Wang, J.H., Liu, Q., Zhang, Z., Lee, J.Y., Li, Y., Yu, B., Lin, Z., Kim, S.Y., Jia, J.: Generative video propagation. CVPR (2025)
27. Liu, Y., Wang, T., Liu, F., Wang, Z., Lau, R.W.: Shape-for-motion: Precise and consistent video editing with 3d proxy. ACM SIGGRAPH Asia 2025 Conference Papers (2025)
28. Luce, R.D.: Individual Choice Behavior: A Theoretical Analysis. John Wiley & Sons, New York (1959)
29. Maystre, L., Grossglauser, M.: Fast and accurate inference of plackett–luce models. In: Advances in Neural Information Processing Systems 28 (2015)
30. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Howes, R., Huang, P.Y., Xu, H., Sharma, V., Li, S.W., Galuba, W., Rabbat, M., Assran, M., Ballas, N., Synnaeve, G., Misra, I., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: Dinov2: Learning robust visual features without supervision (2023)
31. Ouyang, W., Dong, Y., Yang, L., Si, J., Pan, X.: I2vedit: First-frame-guided video editing via image-to-video diffusion models. ACM SIGGRAPH Asia 2024 Conference Papers (2024)

32. Park, B., Kim, B.H., Chung, H., Ye, J.C.: Redirector: Creating any-length video retakes with rotary camera encoding. arXiv preprint arXiv:2511.19827 (2025)
33. Shi, X., Huang, Z., Wang, F.Y., Bian, W., Li, D., Zhang, Y., Zhang, M., Cheung, K.C., See, S., Qin, H., et al.: Motion-i2v: Consistent and controllable image-to-video generation with explicit motion modeling. In: ACM SIGGRAPH 2024 Conference Papers. pp. 1–11 (2024)
34. Shin, J., Li, Z., Zhang, R., Zhu, J.Y., Park, J., Shechtman, E., Huang, X.: Motionstream: Real-time video generation with interactive motion controls. arXiv preprint arXiv:2511.01266 (2025)
35. Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., Liu, Y.: Roformer: Enhanced transformer with rotary position embedding. Neurocomputing **568**, 127063 (2024)
36. Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., Lempitsky, V.: Resolution-robust large mask inpainting with fourier convolutions. WACV (2021)
37. Teng, Y., Xie, E., Wu, Y., Han, H., Li, Z., Liu, X.: Drag-a-video: Non-rigid video editing with point-based interaction. arXiv preprint arXiv:2312.02936 (2023)
38. Wan, T., Wang, A., Ai, B., Wen, B., Mao, C., Xie, C.W., Chen, D., Yu, F., Zhao, H., Yang, J., Zeng, J., Wang, J., Zhang, J., Zhou, J., Wang, J., Chen, J., Zhu, K., Zhao, K., Yan, K., Huang, L., Feng, M., Zhang, N., Li, P., Wu, P., Chu, R., Feng, R., Zhang, S., Sun, S., Fang, T., Wang, T., Gui, T., Weng, T., Shen, T., Lin, W., Wang, W., Wang, W., Zhou, W., Wang, W., Shen, W., Yu, W., Shi, X., Huang, X., Xu, X., Kou, Y., Lv, Y., Li, Y., Liu, Y., Wang, Y., Zhang, Y., Huang, Y., Li, Y., Wu, Y., Liu, Y., Pan, Y., Zheng, Y., Hong, Y., Shi, Y., Feng, Y., Jiang, Z., Han, Z., Wu, Z.F., Liu, Z.: Wan: Open and advanced large-scale video generative models. arXiv preprint arXiv:2503.20314 (2025)
39. Wang, A., Huang, H., Fang, J.Z., Yang, Y., Ma, C.: Ati: Any trajectory instruction for controllable video generation. arXiv preprint arXiv:2505.22944 (2025)
40. Ye, Z., Huang, H., Wang, X., Wan, P., Zhang, D., Luo, W.: Stylemaster: Stylize your video with artistic generation and translation. In: CVPR. pp. 2630–2640 (2025)
41. Yu, M., Hu, W., Xing, J., Shan, Y.: Trajectorycrafter: Redirecting camera trajectory for monocular videos via diffusion models. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 100–111 (2025)
42. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: ICCV. pp. 3836–3847 (2023)

Supplementary Material

A Overview

This supplementary material includes two parts: (i) detailed baseline repurposing procedures (Sec. B), including the shared 3D trajectory extraction pipeline, method-specific control conversion for ATI, DaS, VACE, I2VEdit, and SFM, and notes on excluded baselines; and (ii) TrajectoryAtlas details (Sec. C), including trajectory simulation, online scene modification for no-hit cases, implementation settings, and dataset statistics.

In addition, on our project webpage we provide qualitative videos showing ground truth source-target pairs, comparative results of TrajectoryMover against all baseline methods, and comparisons between TrajectoryMover and its ablation variants.

B Baseline Repurposing Details

Setup. No baseline natively supports our source to target object relocation task, so we repurpose each baseline through external controls while keeping the original model architecture and training procedure unchanged. Each video pair we apply the baselines to provides source video A, ground truth target video B, and source/target segmentation videos where the foreground object is black and the background is white. For prompt conditioned methods, we use the same neutral prompt across methods. We normalize all baseline outputs to a common evaluation format using letterbox resizing, preserving aspect ratio to 1280×720, followed by temporal resampling to 81 frames at 16 fps. Temporal normalization is performed by uniformly sampling 81 target positions on the source video timeline and assigning each target position its nearest source frame, which reuses frames when the original video is shorter.

Common 3D trajectory extraction for motion conditioned baselines. ATI, DaS, and VACE require some form of trajectory-like control as input condition, so for these baselines we reconstruct a 3D object trajectory from the source video. As shown in Fig. B.1, we first estimate source depth from A using Video-Depth-Anything [6]. As this per-frame depth includes correspondences across the video, we can simply select one or multiple sample points in the first-frame mask of the source object, get their depth, and follow them across the video to obtain 3D object trajectories. We then move these 3D trajectories with the ground truth edit offset δ, giving us 3D object trajectories for the edited video. These moved 3D trajectories are converted into each baseline's native control format. Note that these moved trajectories cannot take into account interactions with the scene, which is a natural limitation of existing methods that require exact trajectories as input condition.

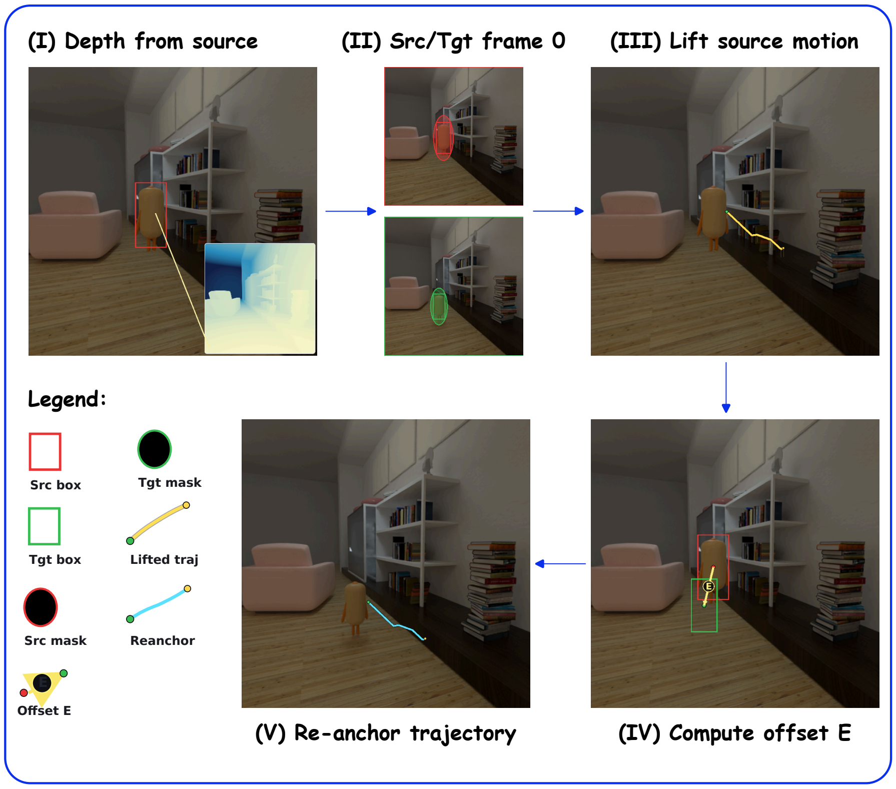


Fig. B.1. Common baseline repurposing pipeline. We convert each source-target case into method specific controls. We estimate source depth, extract source and target frame 0 masks, lift source object motion to a 3D trajectory proxy, compute the frame 0 displacement E, and re-anchor the trajectory to the target start. Red indicates source localization, green indicates target localization, and trajectory overlays visualize source and re-anchored motion elements used for downstream baseline control conversion.

ATI. ATI [39] is driven through its native trajectory interface. We project the moved 3D trajectories back to the 2D image plane of the target camera to obtain 2D trajectories that we can input directly into ATI. We optionally add a small number of static background anchors to make sure the camera remains static. ATI inference is then run normally with the target frame 0 image and generated tracks. The original output resolution is 832×464, at 16 fps, with 81 frames over 5 s.

DaS. We obtain 3D tracks for the source video using one of the tracking methods used by DaS [15], apply the edit offset δ to the tracks of the foreground object, and feed the edited tracks, as well as the source video as input conditions to DaS. This is fully compatible with the existing DaS pipeline. The original output resolution is 720×480, at 9.68 fps, with 49 frames over 5 s.

Algorithm 1 TrajectoryAtlas data generation

```
Require: Scene set $\mathcal{S}$ with cameras, object set $\mathcal{O}$, task set $\mathcal{T}$, seeds $\mathcal{R}$
Ensure: Dataset $\mathcal{D}$ of paired videos and masks
  $\mathcal{D} \leftarrow \emptyset$
  for $r \in \mathcal{R}$ do
      Sample scene $s \in \mathcal{S}$, camera frame $f$, object $o \in \mathcal{O}$, task $\tau \in \mathcal{T}$
      Build or reuse collision caches $M_s$ (scene), $M_o$ (object)
      $f^\star \leftarrow \textsc{PreflightSkipRender}(s, f, o, \tau)$
      $(x_0, y_0, m) \leftarrow \textsc{PairedPlacementScale}(s, f^\star, o, \tau)$
      if not $\textsc{ValidStart}(x_0, y_0, m)$ then
          continue
      end if
      $M \leftarrow M_s$
      if $\textsc{NoHitMode}(\tau)$ then
          $\hat{\Gamma} \leftarrow \textsc{NominalTrajectory}(\tau, x_0, m)$
          $M \leftarrow \textsc{RemoveNonStructuralObstacles}(M_s, \hat{\Gamma})$
      end if
      $A \leftarrow \textsc{SimulateAndRender}(s, f^\star, o, \tau, x_0, m, M)$
      $B \leftarrow \textsc{SimulateAndRender}(s, f^\star, o, \tau, y_0, m, M)$
      $(M_A, M_B) \leftarrow \textsc{RenderBinaryMasks}(A, B)$
      if $\textsc{CanonicalOutputCheck}(A, B, M_A, M_B)$ then
          Save runtime config and logs
          $\mathcal{D} \leftarrow \mathcal{D} \cup \{(A, B, M_A, M_B)\}$
      end if
  end for
  return $\mathcal{D}$
```

VACE. VACE [19] is repurposed using its trajectory video control interface. We create bounding boxes of the edited foreground object using the per-frame depth point cloud of the source object moved with the edit offset δ and use these bounding boxes to render a native layout trajectory control video. We additionally provide (i) a source object reference crop from source segmentation and (ii) background preserving controls marking editable regions. These controls are passed to the original VACE inference. The original output resolution is 832×480, at 16 fps, with 81 frames over 5 s.

I2VEdit. I2VEdit [31] is repurposed via first-frame editing. We extract the source object from frame 0 using source segmentation, remove the original object region with LaMa [36] inpainting using a dilated removal mask, and paste the object at the target frame 0 location using geometric compositing. The edited first frame is then used with the standard I2VEdit pipeline to propagate edits over time. The original output resolution is 512×512, at 8 fps, with 40 frames over 5 s. After normalization for fair comparison, the result appears visually smaller within the padded frame because we preserve aspect ratio rather than stretching the lower-resolution, shorter raw output.

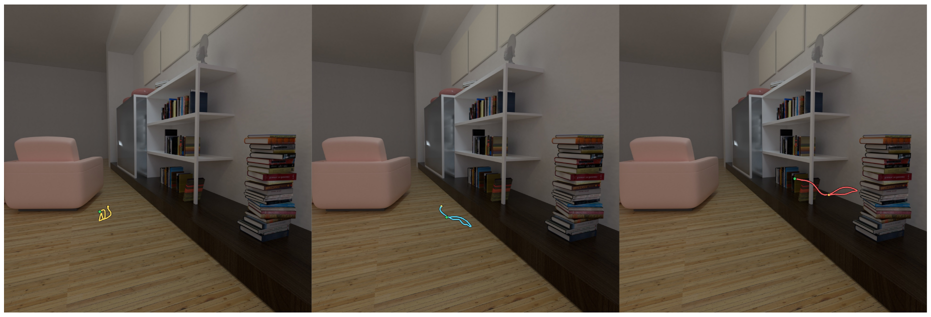


Fig. B.2. Drag trajectory variants. We visualize the three planar drag references used in our simulator: spiral (left), circular (middle), and S-shaped (right), rendered from the scene camera as 3D trajectories. Each curve shows directed start-to-goal progression and highlights the motion templates used for drag supervision.

SFM. Shape-for-Motion (SFM) [27] is a 3D-aware video editing framework executed through its original multi-stage reconstruction, edit, and render pipeline. Unlike ATI, DaS, and VACE, SFM does not use our depth-based trajectory transfer control. Instead, for relocation we apply an external geometric edit by translating the reconstructed canonical mesh using the source and target frame 0 mask displacement. The main SFM model remains unchanged. The original output resolution is 768×512, at 7 fps, with 21 frames over 3.0 s.

Other excluded baselines. We do not include GenProp [26] because public code is not available. We also do not include VideoHandles [22], since it is designed for static object composition editing in static scenes and is not directly suitable for moving object trajectory relocation.

Limitation of simple trajectory transfer. For ATI, DaS, and VACE, we use copied source motion re-anchored to the target start position. By design, this transfer does not explicitly reason about new scene interactions encountered after relocation, which can reduce plausibility under collisions.

C TrajectoryAtlas Details

TrajectoryAtlas generation pipeline. Algorithm 1 summarizes our end-to-end data generation process for paired source-target videos. For each seed, we sample a scene, camera frame, foreground object, and task, then run a lightweight preflight step to obtain a valid frame and paired start and target placements with scale control. In `NoHit` mode, we first estimate a nominal trajectory and remove only non-structural obstacles from the scene collision model, otherwise we keep the original collision setup. We then simulate and render both source and target runs, generate binary masks, and retain a sample only if canonical output checks pass.

For every accepted sample, we save videos, masks, and runtime configuration for reproducibility.

C.1 Object trajectory details

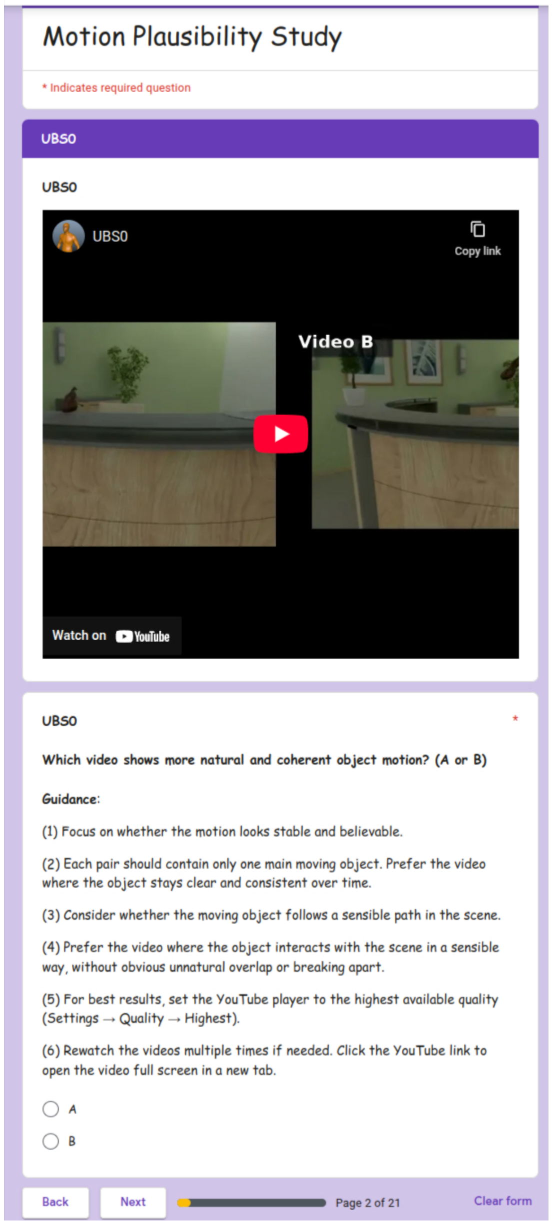
We simulate all object motions with rigid body dynamics in Bullet [10] and produce our trajectories either via initial conditions of the simulation, or by introducing additional forces during the simulation. Each trajectory starts from a valid first-frame placement x_0 that is visible from the camera and intersection-free.

- `Drop`: We set initial velocity to zero and rely on the physics simulation's gravity to create the motion.
- `Throw`: We set the initial velocity to be parallel to the camera direction projected to the ground plane, with a magnitude equal to 6.0 by default.
- `Roll`: The initial position x_0 places the object on a supporting surface and we set the initial velocity to be parallel to the camera direction, projected to the supporting surface with a default speed 2.0.
- `Drag`: The initial position x_0 places the object on a supporting surface and we set the initial velocity to zero. As we want to simulate the object moving or being dragged along a path, we first randomly choose the path that the object should follow from one of three pre-defined paths: a circular path, an S-shaped path, or a spiral path, all of them parallel to the ground. Figure B.2 shows a few examples. We define a target position that moves along the path with constant normalized progress and apply a damped spring force that pulls the object toward the target point. For drag, control is planar with no vertical correction.

Using a physics simulation ensures that any interactions of the object with the scene are plausible.

C.2 Online scene modification details

The goal of this step is to remove clutter in the scene that would obstruct object motion. We only want to remove non-structural objects, i.e., objects that could realistically be moved in an actual scene, such as tables and chairs, but not structural objects like walls or floors. To distinguish structural from non-structural objects, we use heuristics based on object name and geometry. For example, walls and floors are detected as large flat geometry, or through object names containing words such as 'wall', 'floor', or 'ceiling'. We then compute a nominal trajectory corridor from the task dynamics and remove non-structural items that intersect this corridor. The filtering is applied as a single pass union filter over the predicted trajectory points for the sampled placements, after which we export the filtered collision mesh and rerun the task with unchanged motion parameters.


Motion Plausibility Study

* Indicates required question

UBS0

UBS0

UBS0 *

Which video shows more natural and coherent object motion? (A or B)

Guidance:

(1) Focus on whether the motion looks stable and believable.

(2) Each pair should contain only one main moving object. Prefer the video where the object stays clear and consistent over time.

(3) Consider whether the moving object follows a sensible path in the scene.

(4) Prefer the video where the object interacts with the scene in a sensible way, without obvious unnatural overlap or breaking apart.

(5) For best results, set the YouTube player to the highest available quality (Settings → Quality → Highest).

(6) Rewatch the videos multiple times if needed. Click the YouTube link to open the video full screen in a new tab.

○ A

○ B

Fig. C.1. User study interface. Participants compare two anonymized videos (A/B) from the same source-target setup and select the one with more natural and coherent object motion. The guidance text standardizes the evaluation criteria (motion stability, object consistency, trajectory reasonableness, and scene interaction plausibility). The same interface format is used for all user study experiments in this project.

C.3 Implementation details

We implement data generation with Blender Cycles for rendering and Bullet for physics simulation. Runs are executed in parallel on a GPU cluster with one render worker per GPU and reusable collision caches for both scene and object geometry.

C.4 Dataset statistics

TrajectoryAtlas is built from curated Evermotion indoor scenes and a foreground pool of 119 assets, with 98 Objaverse objects and 21 primitives with Bullet proxy variants. After filtering, the dataset contains 21,533 paired samples, which corresponds to 43,066 RGB videos and 43,066 binary mask videos. We use 20,459 pairs for training and 1,074 pairs for testing. Trajectory composition in the filtered set is 9,537 drag pairs, 3,113 throw pairs, 3,162 roll pairs, 2,867 drop pairs, 1,564 placement static pairs, and 1,290 placement falling pairs. For drag, we use three path templates with counts, 3,151 O, 3,184 S, and 3,202 spiral. For hit annotation, we have 9,325 hit pairs and 9,354 no-hit pairs.

D Acknowledgments

We thank Valentin Deschaintre and Iliyan Georgiev for insightful discussions and valuable feedback. We are also grateful to Yannick Hold-Geoffroy and Vladimir Kim for their help with the assets used in dataset generation, and to Zhening Huang for support with the model pipeline. The core ideas for this project were developed while Kiran Chhatre was an intern at Adobe Research.