Table of Contents

1 Implementation Details

1.1 Video Generation

We generate video collections for each input image and text prompt using multiple image-to-video (I2V) models, including FramePack [Zhang et al. 2025a], Wan2.2 [Wan et al. 2025], and ATI [Wang et al. 2025a], by varying the random seed of the initial noise. For FramePack, Wan2.2, and ATI, we use the official implementations with their default generation parameters. Time-to-Move [Singer et al. 2025] is used only for the Video Motion Montage application described in the paper. For this setting, we found that the default parameters of the official implementation did not sufficiently preserve the input motion guidance, so we increased `tweak-index` from 3 to 6 and `tstrong-index` from 6 to 10. For the inbetweening application, we adapted the inverted anti-drifting implementation provided in FramePack to generate intermediate frames between two input keyframes.

Generating large video collections incurs a nontrivial preprocessing cost, depending on the I2V model, generation settings, and hardware. In our environment, generating an 81-frame video at 832×480 resolution with 40 sampling steps using 4 NVIDIA A100 40GB GPUs required approximately 2.6 minutes with Wan2.2 and 1.7 minutes with ATI, resulting in roughly 4.5 and 3 hours, respectively, for generating 100 videos. Under our settings, generating a 61-frame video at 832×480 resolution with 25 sampling steps using FramePack on a single NVIDIA RTX 4090 GPU required approximately 2.5 minutes, resulting in roughly 4 hours for generating 100 videos.

1.2 Object Detection

We extract object and part region masks from the shared initial frame I_{init} using the automatic object segmentation functionality of Segment Anything (SAM) [Kirillov et al. 2023]. These masks are treated as objects for subsequent motion analysis and interaction. Specifically, we use `SAMAutomaticMaskGenerator` with `crop_n_layers = 2`, which applies SAM to progressively cropped image regions to produce masks at multiple spatial scales.

We first generate a set of masks using SAM and sort them in descending order of area. To construct object-part groupings, we analyze inclusion relationships between masks. For each pair of masks, we compute the intersection area normalized by each mask area. If one mask is largely contained in another, with an inclusion ratio above 0.9 in our implementation, we treat the larger mask as a parent and the smaller one as its child. When masks partially overlap without satisfying this containment criterion, we treat them as separate candidates. Masks not contained in any other mask are treated as parent candidates. We then recursively build the hierarchy until no further child masks are found. This hierarchical decomposition represents both coarse object-level masks and fine part-level masks, enabling flexible interaction at multiple scales. In terms of computational cost, the object detection process takes approximately 20–30 seconds per scene on an NVIDIA RTX 4080 GPU.

1.3 Motion Analysis

For each detected object, we select a representative point based on the medial axis computed from the binary mask, yielding a stable and compact motion representation from a central region location. We track this point across all frames using the official implementation of CoTracker3 [Karaev et al. 2025] with default settings, initializing the point in the first frame. While tracking multiple points and averaging their trajectories could improve robustness, we adopt a single-point representation to reduce computational cost. This also avoids discrepancies between averaged trajectories and the motion of any actual tracked point, which could otherwise lead to a mismatch between the user's trajectory selection intent and the resulting video.

We do not implement explicit handling for point-tracking failures. Tracking may degrade in challenging scenarios such as low-texture regions, motion blur, occlusions, fast motion, or complex scenes with multiple interacting objects. In such cases, the extracted trajectories may become noisy or unstable. Although these failure cases were not prominent in our evaluated scenarios, they remain a limitation of our approach. Future work could incorporate multi-point tracking or robustness-aware filtering to improve robustness.

The runtime of the motion analysis increases with both video length and the number of tracked points. On an NVIDIA RTX 4080 GPU, processing a 60-frame video with 34 tracking points takes 1.13 seconds, while an 81-frame video with 262 tracking points takes 6.45 seconds. Beyond the precomputed representative trajectories, users can select and track arbitrary points of interest in the generated videos: tracking a single point in a 60-frame video takes 0.29 seconds (about 29 seconds for 100 videos).

Trajectory simplification. We simplify each trajectory using a variant of the Ramer–Douglas–Peucker (RDP) algorithm [Douglas and Peucker 1973]. Instead of using the distance threshold ϵ in the original formulation, we enforce a fixed number of vertices l (set to 15 in our implementation), which stabilizes computational cost. Starting from the endpoints, we iteratively add the point with the largest approximation error until l vertices are selected.

Trajectory distance. We compute trajectory distance using Dynamic Time Warping (DTW) based on the dynamic programming formulation of Berndt and Clifford [Berndt and Clifford 1994], using Euclidean distance between trajectory points. We perform alignment over the full grid without window constraints, enabling the method to capture temporal variations without imposing heuristic limits. Accordingly, our DTW implementation does not require additional heuristic parameters.

Video distance. We define video distance at the scene-level by taking the maximum DTW distance across all detected objects, rather

than focusing on variation in a single object. This encourages the selection of videos with globally distinct motion patterns, aligning with MVB's exploratory workflow of first understanding the overall motion distribution and then narrowing the search space. For example, videos with similar primary object motion but different surrounding object behaviors would be treated as distinct, enabling broader exploration of scene dynamics.

The computational complexity of DTW between two trajectories of lengths m and n is $O(mn)$. In our implementation, trajectories are simplified to length l, resulting in a cost of $O(l^2)$ per comparison.

Diversity-based video selection. To reduce visual clutter and present representative trajectories, we adopt a diversity-based selection strategy based on Maximal Marginal Relevance (MMR) [Carbonell and Goldstein 1998]. The original MMR formulation balances query relevance and diversity. In our setting, however, all videos are generated from the same input image and prompt, making the relevance term effectively constant across candidates. Therefore, we use a diversity-only variant of MMR by setting $\lambda = 0$ in the original formulation. Under this diversity-only formulation, we iteratively select videos that maximize the minimum DTW distance to already selected ones. This encourages diverse motion patterns while avoiding redundancy.

Compared to clustering-based approaches, this method offers two advantages. First, it does not require specifying the number of clusters, which is difficult to determine in practice because it depends on the structure of the generated collection. Second, it produces an ordered list of trajectories, allowing users to progressively reveal increasingly diverse motions by interactively adjusting the number of displayed trajectories. This property is particularly suited for interactive exploration.

The selection process is performed as an offline precomputation step. Given N videos, greedy selection requires $O(N^2)$ video distance evaluations. Each distance computation takes $O(|O_{\text{init}}|\, l^2)$, where $|O_{\text{init}}|$ denotes the number of objects and l the number of points per trajectory. Thus, the overall precomputation complexity is $O(N^2 |O_{\text{init}}|\, l^2)$.

1.4 Exploration

Overview. When users double-click near an object or part, we identify the corresponding precomputed object mask from the clicked position. We do this by looking up the clicked pixel in the object maps, where each pixel stores the integer label of the object that contains it. If the clicked pixel belongs to an object mask, we retrieve the associated trajectories across all generated videos. Each trajectory is represented as a 2D polyline overlaid on I_{init} and is associated with the index of the generated video from which it was extracted. Thus, each polyline serves as an interactive handle: clicking it selects the corresponding video.

For speed visualization, we convert each object trajectory into a speed curve by measuring the frame-to-frame displacement of the tracked point. We smooth the resulting sequence with a short sliding window and normalize it across the candidate videos so that curves can be compared on a common scale. The curve is then displayed as normalized time versus relative speed.

Visualize representative trajectories. Representative trajectories are ordered offline using the diversity-based selection described above. During interaction, users can adjust the number of displayed trajectories using the mouse wheel. In our implementation, the number of displayed trajectories changes in increments of 10, with a minimum of nine trajectories displayed. When no query-based filtering is active, the displayed trajectories follow the precomputed representative ordering. However, when filtering is active, the displayed trajectories follow the current query-based ranking.

Filter by drawing strokes. For stroke-based filtering, users draw a freehand stroke on the shared initial frame. We associate each stroke with the selected object whose representative point is closest to the stroke's starting point. We then compare the stroke only with the trajectories of the associated object using Dynamic Time Warping (DTW), using Euclidean distance between 2D points. Candidate videos are ranked by their DTW distance to the user-drawn stroke. When multiple object-specific strokes are specified, we sum the DTW distances across the corresponding objects. We then retain the top-K ranked videos, where $K = 27$ in our implementation.

During interaction, a user-drawn stroke typically contains $m = O(10^2)$ points, and each candidate trajectory contains $l = O(10^2)$ points. Thus, stroke-based filtering over N videos has a complexity of $O(Nlm)$ for a single queried object. In practice, for $N \approx 100$, this computation typically takes less than 0.1 seconds in our implementation, enabling interactive response times.

Speed-curve filtering follows the same idea as trajectory-stroke filtering. When the user draws a stroke on the speed panel, we compare it with each candidate speed curve using DTW and rank videos by the resulting distance. When both trajectory and speed filters are active, we keep videos that are highly ranked by both.

Filter by positive or negative regions. Users can specify rectangular positive or negative regions by dragging while holding the Shift key. A trajectory satisfies the region constraints if it passes through at least one positive region when positive regions are specified, and does not pass through any negative region. We support both global region constraints and object-specific region constraints. A candidate video is retained only when its trajectories satisfy both the global constraints and the corresponding object-specific constraints.

When both region constraints and stroke-based filters are specified, we first apply the region constraints to remove videos whose trajectories do not satisfy the specified spatial conditions. We then compute DTW distances only for the remaining candidate videos and rank them according to their stroke-matching distances. In other words, region constraints act as a hard filter, while stroke-based filtering ranks the remaining candidates by motion similarity.

Adjust the scale of focus. Because SAM produces masks at multiple spatial scales, a single click location may correspond to multiple nested region masks, such as a coarse object mask and finer part masks. We store these candidate regions as an ordered hierarchy based on mask containment relationships. After selecting an object, users can move between these candidates using the arrow keys, allowing them to shift the focus from coarse object-level motion to fine-grained part-level motion. When the active region changes, we

update the displayed mask and replace the trajectory set with the trajectories associated with the newly selected region.

Detect newly appearing objects. Newly appearing object candidates are obtained by segmenting the last frame with SAM and tracking each candidate mask backward to the first frame using CoTracker. A candidate is kept if its backward trajectory does not correspond to an existing first-frame object.

2 User Study Details

2.1 Procedure Details

For the user study, we generated 100 videos from the images shown in Figure 1 (basketball, drift-car, tree-stump, and miniature-police) using a video generation model [Zhang et al. 2025a]. The text prompts used to generate videos from each image are provided in Table 1. Additionally, to prevent camera motion and visual appearance changes in the generated videos, we used the following negative prompt consistently for all generations: *"Lighting changes, flickering, background changes, camera angle changes, camera movement, blurriness, low resolution."* We categorized the basketball and drift-car cases as single-object video collections, and the tree-stump and miniature-police cases as multiple-object video collections. Finally, the task details for each case (Table 2) and the survey questions (Table 3) are provided.

Fig. 1. **Images used in the user study.** We label each case as follows: (a) basketball, (b) drift-car, (c) tree-stump, and (d) miniature-police.

Table 1. **Text prompts used to generate videos from each image.**

Case	Text Prompt
basketball	A basketball in mid shot.
drift-car	A car dynamically drifting on a racetrack.
tree-stump	A bird, a dog, and a child interacting together.
miniature-police	Police officers catching a thief.

2.2 Additional Results

We conducted a task-specific usability evaluation of each interface using custom questions (Table 3) related to generated video exploration (Figure 2). The results show significant differences between MVB and the baseline for the understanding task and the search task (understanding: $t(12) = -3.630, p = .004$; search: $t(12) = -3.118, p = .010$). In contrast, no significant difference was observed for the comparison task (comparison: $t(12) = -2.138, p = .056$). In addition, we observed a significant difference in overall usefulness for the tasks (overall usefulness: $t(12) = -4.290, p = .001$).

Besides, we measured the usability of each interface using the SUS scale [Brooke et al. 1996], obtaining scores of 67.50 ± 13.90 for the baseline and 70.21 ± 15.86 for our interface. The difference in SUS scores was not statistically significant ($t(12) = -0.391, p = .704$).

Finally, we analyzed participants' open-ended feedback to qualitatively assess the effectiveness of each interface. For the baseline, participants pointed out the unreliability of text-based search. For example, P5 noted that "irrelevant videos often appeared at the top of the results." In addition, multiple participants commented on the difficulty of describing video content using text alone when videos were visually similar. P9 stated that "there were many similar videos, and it was difficult to come up with queries that could distinguish them," and P11 remarked that "when you are comparing a bunch of similar videos or seeking a specific video, using only text to distinguish small details is hard and time-consuming."

In contrast, participants provided positive feedback on the effectiveness of the trajectory-query-based interface in MVB. P5 commented that "for tasks where I needed to find a specific video, specifying the motion of an object or its parts made it relatively easy to locate." Similarly, P11 noted that "overall the interface is very useful and easy to understand and operate." At the same time, participants also identified areas for improvement in our interface. Several participants commented on object selection in MVB. P1 mentioned that "when multiple objects were selected for a car, it was hard to tell which part was being selected," and P6 stated that "it would have been better to specify more fine-grained objects, such as tires," indicating a need for more detailed object-level control. In addition, P7 pointed out the lack of prior knowledge about object boundaries, noting that "it would be helpful to know where feature points are before clicking, such as whether the upper and lower body are separated." Finally, P2 pointed out the limitations of planar motion representation, commenting that "using only 2D motion made it difficult to select the correct object when depth relationships, such as front and back, were involved." suggesting the potential of incorporating depth information for object selection.

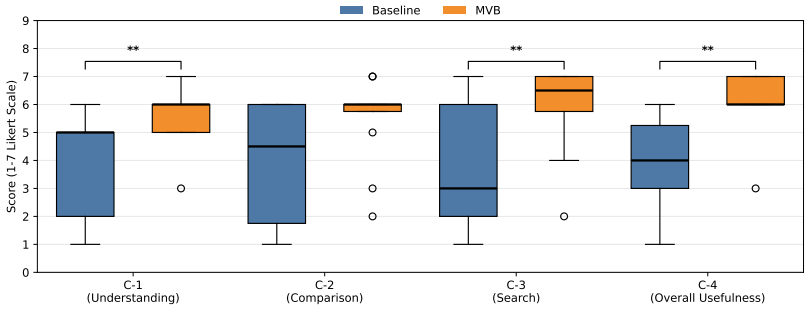

Fig. 2. Answers for custom questions Table 3. The p-values indicate statistical significance: $p < 0.05$ (*), $p < 0.01$ (**), and $p < 0.001$ (***).

Table 2. **Exploration tasks used in the user study.** For the understanding and comparison tasks, we present the exact questions posed to the participants. For the search tasks, we describe the content of the videos that participants were asked to find.

Case	Task Type	Task Example
basketball	Understanding	Is there any video where the basketball enters the hoop?
	Comparison	Which is more frequent: (A) videos where the basketball hits the hoop or (B) those where it passes by the side of the hoop?
	Search (typical)	Please find the target video (Video-24, where the basketball hits the hoop).
	Search (rare)	Please find the target video (Video-38, where the basketball bounces off the hoop and goes to the left).
drift-car	Understanding	Is there any video where the racing car goes off the right edge of the screen?
	Comparison	Which is more frequent: (A) videos where the racing car drives on the curb on the left side of the screen or (B) those where it does not?
	Search (typical)	Please find the target video (Video-12, where the racing car drifts while taking the inside line).
	Search (rare)	Please find the target video (Video-21, where the racing car drives off toward the right-side of the screen).
tree-stump	Understanding	Is there any video where the bird lands on the dog after flying?
	Comparison	Which approaches the bird more frequently, (A) the child or (B) the dog?
	Search (typical)	Please find the target video (Video-54, where the dog and the child approach the bird).
	Search (rare)	Please find the target video (Video-93, where the bird flies toward the dog).
miniature-police	Understanding	Is there any video where the thief escapes toward the left?
	Comparison	Which is more frequent: (A) videos where the thief and the right-side police officer pass by each other or (B) those where they do not?
	Search (typical)	Please find the target video (Video-38, where the thief escapes to the right, passing the police officer).
	Search (rare)	Please find the target video (Video-13, where the thief escapes toward the back, chased by two police officers).

Table 3. **Questionnaire items used in the user study.**

ID	Question	Response scale
A. NASA-TLX Questionnaire		
A1	How mentally demanding was the task?	1. Very Low – 7. Very High
A2	How physically demanding was the task?	1. Very Low – 7. Very High
A3	How hurried or rushed was the pace of the task?	1. Very Low – 7. Very High
A4	How successful were you in accomplishing what you were asked to do?	1. Perfect – 7. Failure
A5	How hard did you have to work to accomplish your level of performance?	1. Very Low – 7. Very High
A6	How insecure, discouraged, irritated, stressed, and annoyed were you?	1. Very Low – 7. Very High
B. System Usability Scale (SUS)		
B1	I think that I would like to use this system frequently.	1. Strongly Disagree – 5. Strongly Agree
B2	I found the system unnecessarily complex.	1. Strongly Disagree – 5. Strongly Agree
B3	I thought the system was easy to use.	1. Strongly Disagree – 5. Strongly Agree
B4	I think that I would need the support of a technical person to be able to use this system.	1. Strongly Disagree – 5. Strongly Agree
B5	I found the various functions in this system were well integrated.	1. Strongly Disagree – 5. Strongly Agree
B6	I thought there was too much inconsistency in this system.	1. Strongly Disagree – 5. Strongly Agree
B7	I would imagine that most people would learn to use this system very quickly.	1. Strongly Disagree – 5. Strongly Agree
B8	I found the system very cumbersome to use.	1. Strongly Disagree – 5. Strongly Agree
B9	I felt very confident using the system.	1. Strongly Disagree – 5. Strongly Agree
B10	I needed to learn a lot of things before I could get going with this system.	1. Strongly Disagree – 5. Strongly Agree
C. System Effectiveness (Custom Questions)		
C1	Using this system, I was able to effectively understand the motions in the video collection.	1. Strongly Disagree – 7. Strongly Agree
C2	Using this system, I was able to effectively compare motions across videos in the collection.	1. Strongly Disagree – 7. Strongly Agree
C3	Using this system, I was able to effectively search for specific motions in the video collection.	1. Strongly Disagree – 7. Strongly Agree
C4	Overall, I found this system helpful for accomplishing the tasks.	1. Strongly Disagree – 7. Strongly Agree
D. Open-ended Comments		
D1	Please feel free to write any additional comments or observations.	—