CAMRM: TOWARDS RELIABLE CAMERA-MOTION CONTROL IN VIDEO GENERATION WITH MOTION-GROUNDED RATING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time scaling provides a practical way to improve camera-motion control in video generation: sample a clip, rate its alignment with the request, and revise the prompt if needed. This process requires a reliable rating model (RM) to determine when to stop. Geometry-based methods can struggle when object motion dominates the view, while fine-tuned visual language models (VLMs) often fail to adapt to synthetic videos. Our benchmark of $1{,}064$ human-labelled real and synthetic videos shows that these methods struggle to judge camera-motion alignment. Strong performance on real video does not ensure reliable judgment on generated clips: a model tuned only on real footage reaches 82.3% balanced accuracy on real video but drops to 62.0% on Omni clips. We introduce CamRM, which combines video frames with motion cues to judge camera-motion alignment. We develop CamRM on both the proprietary Gemini model and the open-source Qwen model, with consistent improvements of over 10% across both. We also develop a training-free agentic model (CamRM-A), showing that our approach requires no fine-tuning. Our CamRM achieves 80.7% accuracy and improves balanced accuracy over the strongest baseline by 6.6%. In the motion-guided refinement loop, using CamRM as RM improves human-judged alignment by 13.2% for Veo 3.1 and 18.4% for Omni Flash at $2.2\times$ the generation cost, reaching 83.7% alignment in the Omni Flash T2V case. At 8 generations, the loop reaches 88.3% pooled candidate-set oracle alignment versus 78.4% for i.i.d. resampling, and matches its 8-generation oracle rate with three generations, without generator training.

1 INTRODUCTION

Camera motion determines how a generated video presents its subject, yet a text instruction often fails to produce the intended movement, as shown in Figure 1(a). VBench-2.0 (Zheng et al., 2025) scores the camera-motion dimension of the current state-of-the-art generators at 65.15%, while FilmBench (Wang et al., 2026) scores over 80% for the best generator on a simplified taxonomy. Both benchmarks report scores from automatic evaluators rather than direct human judgments. However, our study shows that a single video from Veo 3.1 or Gemini Omni Flash matches the requested camera motion only 41% to 68% of the time. This disagreement motivates auditing the RMs and improving the generators' camera-motion alignment.

Retraining with better data requires substantial resources and is unavailable to users of commercial video APIs. Meanwhile, a test-time feedback loop provides a practical alternative: an RM checks the video, a rewriter refines the prompt, and the generator produces a new candidate (Long et al., 2025; Liu et al., 2025a; Song et al., 2025). Sampling more candidates increases the chance of fulfilling the request, but the RM decides which video to accept and when to stop. An inferior RM may falsely approve incorrect synthetic clips while sending aligned videos to re-generation. Geometry-based RMs depend on reliable camera estimation (Wang et al., 2025a; Schönberger & Frahm, 2016; Wang et al., 2025b; Li et al., 2025). Even dynamic-scene methods have limitations: MegaSAM reports tracking failures when a moving object dominates the video and difficulties when object and camera motion are collinear (Li et al., 2025). These cases are relevant to generated clips with large foreground subjects or coupled subject and camera motion. VLM-based RMs, in turn, can fail to distinguish detailed camera motion from object motion. In our experiments, the untuned open

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

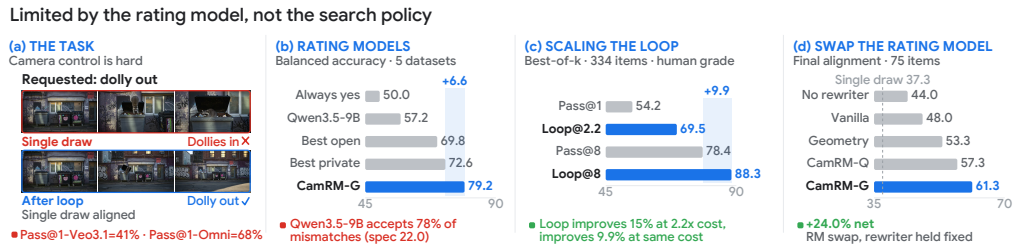


Figure 1: **RM is a critical limitation in test-time camera control.** **(a)** A single draw matches the requested motion only 41% of the time on Veo and 68% on Omni. **(b)** Existing RMs underperform on our benchmark with synthetic videos, while CamRM provides 6.6% improvement. **(c)** Using CamRM improves the camera alignment by 15.3% with $2.2\times$ cost. Even under the same budget, our method performs better. **(d)** Swapping only the RM, CamRM moves net alignment from $+10.7\%$ to $+24.0\%$; dropping the rewriter and keeping CamRM-G nets $+6.7\%$ (Table 3).

RM accepts about 78% of mismatched clips, terminating the loop before correcting the requested motion. Evaluating RMs on generated video is also necessary because performance on real footage does not reliably transfer. Existing benchmarks (Lin et al., 2025; Wang et al., 2025c; Liu et al., 2025b) evaluate camera or cinematographic understanding primarily on real video. Generated clips can introduce texture drift, inconsistent motion, and unwanted shot changes. An RM tuned with real video only can drop over 22% in alignment accuracy for synthetic video. We therefore evaluate RMs on real and generated clips and improve their decisions and reasoning to support test-time scaling.

We propose CamRM, which provides a VLM with two complementary motion cues: a spatial summary of dense optical flow and an estimate of global image motion. The cues make local displacement, parallax, and scale changes explicit, allowing the model to interpret them alongside the frames and the requested motion. We fine-tuned both a proprietary Gemini 3.5 Flash model (CamRM-G) and an open Qwen-3.5-9B model (CamRM-Q) to demonstrate the effectiveness of our design. Our training-free agentic framework, CamRM-A, tests the same cue design without tuning on synthetic data. CamRM-G reaches 79.2% balanced accuracy, outperforming state-of-the-art RM by over 6.6%, as shown in Figure 1(b). Meanwhile, CamRM-Q reaches 75.7% and CamRM-A reaches 78.0% without training. The motion-guided refinement loop improves delivered alignment by 15.3% with $2.2\times$ generation (Figure 1(c)). At eight generations, pooled candidate-set oracle alignment reaches 88.3% versus 78.4% for independent resampling. The loop also matches resampling's eight-generation oracle rate with three generations, showing more effective test-time scaling (Figure 4). Compared with other RMs, CamRM improves alignment under the same settings (Figure 1(d)). Our contributions are listed below:

1. We build a $1{,}064$-clip camera-motion alignment benchmark, including 571 generated clips, and show that strong real-video performance does not guarantee transfer to generated video.
2. We develop CamRM, a motion-grounded RM for synthetic video with 3 variants, all surpassing state-of-the-art camera motion understanding models.
3. We demonstrate improved delivered alignment across both generators and higher candidate-set oracle alignment at matched generation budgets.

2 Related Work

Camera-motion understanding. CameraBench (Lin et al., 2025) evaluates established camera-motion categories in the VLM setting. CineTechBench (Wang et al., 2025c) and ShotBench (Liu et al., 2025b) cover broader cinematographic concepts. CamReasoner (Wu et al., 2026) combines structured reasoning supervision with RLVR. These methods develop camera-understanding capabilities but mainly focus on real videos, ignoring the domain gap between real and synthetic data. While ACaM (Tan et al., 2026) includes both real and Veo-generated clips and fine-tunes Qwen3-VL (Bai et al., 2025), it uses supervised fine-tuning without reasoning, leading to lower performance.

Geometry-based camera understanding. COLMAP (Schönberger & Frahm, 2016) estimates camera geometry through feature matching and reconstruction. Learned methods including VGGT, CUT3R, MegaSAM, MonST3R and π^3 (Wang et al., 2025a;b; Li et al., 2025; Zhang et al.,

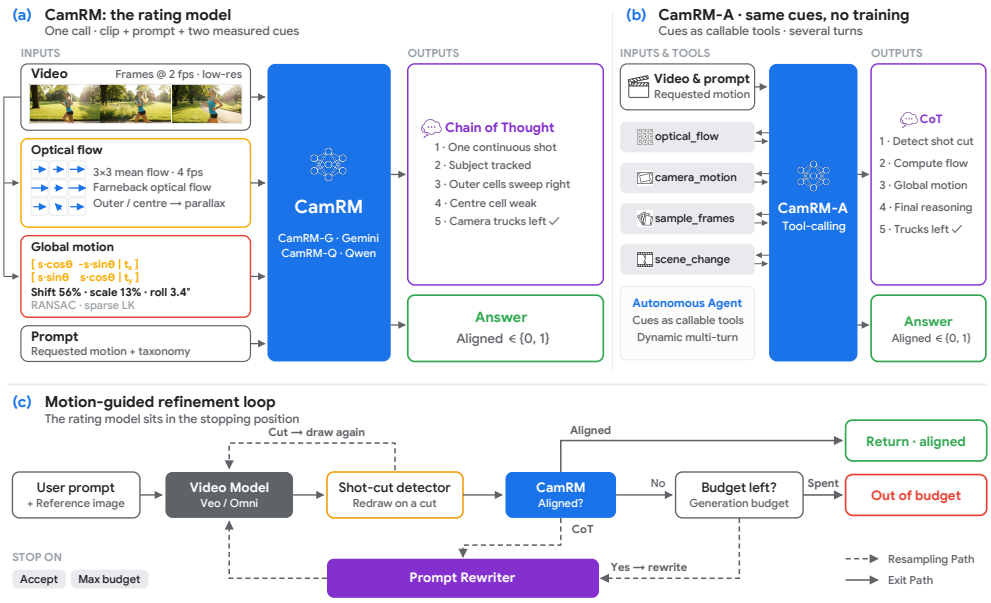


Figure 2: CamRM and the motion-guided refinement loop. (a) The RM reads video frames, the prompt and text summaries of optical flow and global motion. (b) The training-free CamRM-A accesses the same measurements through tools. (c) The RM decides whether to return a clip or ask the rewriter to revise the prompt using its reasoning.

2025; Wang et al., 2025d) extend reconstruction to more varied scenes. Geometry-guided VideoLLMs (Feng et al., 2026) predict motion primitives from VGGT camera tokens for video descriptions. VGGT is designed for static scenes and struggles with substantial non-rigid motion (Wang et al., 2025a). We instead use optical flow and global image motion; our benchmark compares against VGGT as an RM, not the full VideoLLM pipeline. Reconstruction can be difficult with limited parallax, changing focal length or unreliable correspondences. MegaSAM handles dynamic scenes by reducing the influence of moving regions, but its limitations analysis reports loss of tracking when a moving object dominates the video and difficulty when object and camera motion are collinear (Li et al., 2025, Fig. 8). We compare geometric RMs at both sparse and dense frame rates, and use inexpensive local and global image-motion cues to support semantic judgment.

Video evaluation and feedback. VBench-2.0 (Zheng et al., 2025) evaluates camera motion using point tracking and motion-specific rules, while FilmBench (Wang et al., 2026) includes a fine-tuned camera-movement VLM operator in its cinematic evaluation. VideoScore (He et al., 2024), VisionReward (Xu et al., 2026) and VQAScore (Lin et al., 2024) assess broader video quality or prompt alignment. VF-Eval (Song et al., 2025) studies limitations of VLM feedback and explores iterative re-prompting. Our benchmark focuses on a specified camera motion, and our controlled experiment measures the effect of the RM on the video returned to the user.

Test-time scaling for generation. Video-T1 (Liu et al., 2025a) searches sampling trajectories, and VISTA (Long et al., 2025) uses a training-free critic to revise prompts and regenerate videos. Prompt-optimisation methods (Gao et al., 2025a; Cheng et al., 2025) also improve generation through revised instructions. WMReward (Yuan et al., 2026) uses a latent world model to guide denoising-trajectory search for physical plausibility. These methods assume the corresponding RM is reliable. Our loop works through video-generation APIs and targets camera-motion alignment. Building on the role of verification in language-model test-time scaling (Snell et al., 2024; Lightman et al., 2024), we hold the rewriter and search policy fixed to measure the effect of replacing the RM.

3 METHOD

CamRM judges whether a generated video follows the requested camera motion. As shown in Figure 2, it combines video frames with motion cues computed from the same clip. We first define the task and dataset, describe the cues, model training, and motion-guided refinement loop.

3.1 TASK AND DATA CONSTRUCTION

Task definition. Given a user prompt p describing video content and a desired camera motion, a generator g produces a video $v = g(p)$, optionally conditioned on a reference image. A human label $y \in \{0, 1\}$ indicates whether the camera motion matches the request. Our RM $\mathcal{F}_\theta$ predicts $\hat{y} = \mathcal{F}_\theta(p, v) \in \{0, 1\}$. We consider a single continuous shot: a cut can change the viewpoint without any camera movement and can combine shots with different motions. We therefore filter scene changes when constructing the benchmark and check for them before rating a clip in the loop.

Data construction. Given a pre-defined list of 20 camera motions in Appendix A.1, we use Gemini 3.1 Pro to generate 30 prompts containing a description of the scene and a separate instruction for each camera motion for the text-to-video (T2V) case. For the image-to-video (I2V) case, we download 30 real images from a third-party data provider and ask the Gemini model to write a video generation prompt based on the image content and the requested camera motion. We then generate $2{,}400$ clips using both Veo 3.1 and Omni Flash across 600 image-to-video and 600 text-to-video settings. Five people independently rate each clip on a 0–2 alignment scale; a clip is labelled aligned when the mean score exceeds 1. We manually check annotation quality and exclude clips with scene changes or ambiguous annotations. As a result, the remaining synthetic video set contains $2{,}011$ videos. We further include videos from the evaluation set of the CameraBench dataset. We pick the binary classification subset and extract a balanced subset of $1{,}881$ samples. We split the data $70/30$ for training and testing, stratified by camera-motion category, with no shared clips or I2V source images between partitions. The training and test sets contain $2{,}828$ and $1{,}064$ videos, respectively. More details are in Section 4 and Appendix A.1.

3.2 MOTION CUES

Camera motions can produce similar image displacements. Both a pan and a lateral truck move scene content sideways, but translation can introduce depth-dependent parallax. Tilt and pedestal have a similar ambiguity in the vertical direction. Zoom and dolly can both enlarge the subject, but differ in how scene depth affects relative motion. We compute local and global motion cues to expose these differences to the RM, while retaining the frames to interpret scene content.

Optical-flow summary. We compute dense Farneback flow (Farnebäck, 2003) between consecutive frames sampled at 4 fps and average adjacent flow fields to report summaries at 2 fps. We average the displacement vectors within each cell of a 3×3 grid, then report the magnitude of each mean vector and its direction quantized to eight compass headings. The grid preserves differences between regions while reducing local noise. We also report the mean magnitude of the eight outer cells alongside the center-cell magnitude, and horizontal and vertical divergence cues. The regional magnitudes describe relative motion between central and surrounding content; the divergence cues describe expansion or contraction. These are useful signs of parallax and scale change when interpreted with the scene. Because optical flow captures both moving objects and the background, the RM can compare regions to distinguish object motion from camera motion. Appendix A.3 defines the statistics, units, and thresholds.

Global-motion estimate. We track sparse corners with Lucas–Kanade (Lucas & Kanade, 1981) and fit a partial-affine transform with RANSAC (Fischler & Bolles, 1981). The affine matrix in Figure 2 contains horizontal and vertical translation, in-plane rotation, and isotropic scale. We sum the pairwise translation and rotation estimates and multiply their scales to summarise motion over the clip. RANSAC selects consistent tracks within each frame pair, reducing the influence of independently moving foreground objects when sufficient background tracks remain. This complements the local grid: the global estimate describes the dominant shift, roll, and scale change, while the grid shows regional differences. Neither cue alone uniquely determines the 3D camera path, so we provide both along with the video frames. Appendix A.3.2 gives the accumulation rule and its interpretation.

3.3 REPRESENTING MOTION CUES AS TEXT

We express the cues as compact text containing regional directions, magnitudes and global-motion parameters. This representation removes pixel-level variation while retaining the measurements needed to compare motion across regions and over time. A VLM can read a magnitude or scale change directly, without decoding arrow directions or colours from another compressed visual in-

put. Text also allows us to place the outer- and center-region summaries alongside the grid measurements. The model can then relate these values to the requested motion and visible scene structure. Appendix A.3.3 describes the text format and timestamp convention. This design keeps the video as the primary visual input and avoids adding a second sequence of flow images. Providing optical flow as images would double the visual tokens, leading to a much longer input sequence. The VLM would also need to learn to interpret complex arrow maps.

3.4 Rating models and training

The RM receives sampled frames, the user prompt, camera-motion definitions, and the motion cues, and produces reasoning followed by a binary prediction. We use binary human labels to train the tuned models, allowing the reward to be checked directly. Our goal is not only to demonstrate that our design achieves state-of-the-art performance, but also to verify whether our design can provide universal improvement regardless of the model or training data. Thus, we propose 3 variants of CamRM using (1) CamRM-G: a proprietary Gemini model; (2) CamRM-Q: an open-weight Qwen model; (3) CamRM-A: a training-free agentic framework. Appendices A.2 and A.4 give the training settings, checkpoint-selection protocol, and GRPO/SAPO objectives.

CamRM-G. We tune the state-of-the-art VLM, Gemini 3.5 Flash, through the public single-step reinforcement-learning API. Each training example contains the frame sequence, the prompt, and the text motion cues. The API uses a GRPO-style update with a reward for the correct binary judgment and a length penalty to control response length.

CamRM-Q. We fine-tune Qwen3.5-9B (Qwen Team, 2026) with supervised learning followed by RLVR using SAPO (Gao et al., 2025b). The model receives the full motion cues together with frames and the prompt. Its reward combines binary correctness, output format, and a length term. We use SFT to first teach the model the basic understanding of camera motion. We then use SAPO for reinforcement learning, which smoothly reduces the weight of updates as the policy ratio moves away from the sampling policy, avoiding hard clipping of importance ratios. SAPO allows gradual control of policy updates while learning from verifiable labels.

CamRM-A. To test motion cues without task-specific tuning, we give an untuned Gemini 3.5 Flash access to four tools: `optical-flow computation`, `global-motion estimation`, `frame sampling`, and `scene-change detection`, where the first two provide the motion cues as described above. The `frame sampling` allows the model to sample sub-clips with a higher FPS for dense video information. The `scene-change detection` tool helps avoid multi-shot videos. The model chooses which evidence to request before returning its verdict. This design tests whether explicit motion evidence also helps without parameter updates. Tool definitions and ablations are in Appendix A.12, and example reasoning traces are in Appendix A.15.

3.5 Motion-guided refinement loop

We improve camera-motion alignment without updating the generator's weights using the loop in Figure 2(c). We generate a clip, check for scene changes, and ask the RM whether the motion matches the request. Accepted clips are returned directly. After rejection, a vanilla Gemini 3.5 Flash rewriter receives the video, user prompt, and RM's reasoning. It preserves the content and intended motion while describing how the camera, foreground, and background should move. The RM's diagnosis guides these changes, and the rewriter requests a continuous shot without cuts. All controlled configurations allow three refinement rounds after the initial clip. The loop stops on acceptance or budget exhaustion, returning the last candidate if none is accepted. Redraws count towards the same budget.

4 Experiments

We evaluate real-to-generated RM transfer (RQ1), geometric alternatives (RQ2), motion cues and training (RQ3), loop performance (RQ4), and transfer to motion classification (RQ5).

Data. The evaluation benchmark contains $1{,}064$ clips: 184 Veo 3.1 image-to-video clips, 175 Veo text-to-video clips, 120 Omni image-to-video clips, 92 Omni text-to-video clips, and 493 real CameraBench videos that pass the scene-change filter. The generated clips use the human labels de-

Table 1: Rating-model benchmark on the frozen 1,064-clip set. Rates are in %; Veo and Omni merge image-to-video and text-to-video. We compare balanced accuracy (bAcc) alongside accuracy and F1. Bold and underline mark the best and second-best results, excluding the always-aligned baseline; gray cells mark the best open model. Appendix A.1 describes aggregation. * scored on the *mappable subset*: its fixed 10-class head has no tilt, pedestal, or dolly-zoom label.

	Veo (i2v+t2v)			**Omni** (i2v+t2v)			**CameraBench**			**Average**		
Model	Acc	F1	**bAcc**	Acc	F1	**bAcc**	Acc	F1	**bAcc**	Acc	F1	**bAcc**
Trivial												
optical-flow rule	60.45	55.58	64.80	54.72	57.14	61.18	69.98	62.81	70.35	63.72	59.24	66.65
always-"aligned"	58.50	73.15	50.00	67.92	80.82	50.00	50.91	67.47	50.00	56.86	72.05	50.00
Geometry												
VGGT (FPS=2)	57.38	47.39	62.23	58.96	61.41	65.23	72.82	65.10	73.25	64.85	58.39	67.93
VGGT (FPS=24)	59.05	49.88	63.95	58.02	60.98	63.11	73.83	66.49	74.26	65.70	59.79	68.56
COLMAP (FPS=2)	39.00	20.90	44.10	32.55	24.20	41.98	41.18	15.70	41.74	38.72	19.15	42.58
COLMAP (FPS=24)	39.83	21.12	45.91	26.42	16.12	36.01	42.19	7.77	42.89	38.25	13.94	42.54
CUT3R (FPS=2)	58.77	49.29	63.49	52.36	53.92	59.01	70.18	61.21	70.63	62.78	55.74	65.91
CUT3R (FPS=24)	53.20	39.84	58.39	47.64	46.31	55.30	67.14	54.49	67.67	58.55	47.92	62.07
MegaSAM (FPS=2)	55.15	43.32	59.83	55.66	56.76	63.09	66.53	54.04	67.05	60.53	50.96	63.82
MegaSAM (FPS=24)	55.99	45.06	60.82	58.02	59.79	64.91	70.79	61.70	71.25	63.25	55.70	66.47
Open-weight VLMs												
Qwen3.5-9B	59.35	72.54	53.59	69.43	81.51	52.20	62.58	70.44	62.01	62.85	73.35	57.21
Qwen3.5-27B	63.39	73.39	59.05	66.51	78.23	54.07	65.11	73.46	64.56	64.81	74.39	60.61
Qwen3.6-27B	68.38	75.73	63.57	72.69	80.47	65.41	76.24	77.80	76.19	72.88	77.63	69.79
gemma-4-31B-it	67.13	75.62	61.99	69.81	80.71	56.43	71.81	72.14	71.81	69.83	75.02	65.43
FilmOps camera-movement op.*	63.88	57.03	66.35	54.03	57.42	58.85	70.81	62.58	71.14	65.13	59.68	67.08
CameraBench-Qwen2.5-VL-7B	57.38	46.00	61.66	51.89	49.56	59.21	60.85	41.34	61.48	57.89	44.55	61.09
CamReasoner-7B	49.03	32.67	54.93	45.75	40.12	54.78	55.08	31.58	55.65	51.18	33.65	55.23
ACaM Qwen3-VL-4B	64.90	64.57	66.62	55.19	63.53	53.49	69.98	61.66	70.40	65.32	63.01	65.75
ACaM Qwen3-VL-8B	69.36	71.86	69.81	56.13	66.13	52.14	74.85	70.48	75.14	69.27	70.08	68.76
Ours - Based on Qwen3.5-9B												
CamRM-Q	78.83	81.77	76.83	71.70	79.72	65.28	79.31	78.48	79.41	77.63	79.84	75.72
Proprietary VLMs (zero-shot)												
Gemini 3.1 Pro	69.36	70.93	70.71	72.17	77.94	**70.14**	60.24	58.47	60.34	65.70	66.55	65.79
Gemini 3.5 Flash	69.64	74.74	67.95	71.23	79.73	63.57	71.60	75.00	71.38	70.87	75.85	68.67
Gemini 3.6 Flash	76.32	80.91	73.70	72.64	81.23	63.84	75.66	76.92	75.58	75.28	79.13	72.61
Claude Sonnet-5	67.13	65.43	65.64	54.72	59.29	54.68	60.08	44.19	60.64	61.39	54.36	61.14
Claude Opus-4.5	62.67	72.81	58.39	65.57	77.24	54.05	69.44	71.46	69.33	66.38	73.06	62.59
Claude Opus-4.8	63.51	68.74	63.41	62.74	71.51	59.15	61.75	49.73	62.22	62.54	60.48	62.01
Ours - Based on Gemini 3.5 Flash												
CameraBench-only RFT	76.04	79.02	74.98	70.28	79.35	61.97	82.15	81.36	82.26	77.73	80.17	75.76
CamRM-G	78.83	81.33	78.17	**76.42**	**83.43**	69.53	**83.98**	**82.79**	**84.13**	**80.73**	**82.43**	**79.21**
CamRM-A (agentic)	**81.34**	**84.22**	**80.58**	74.06	82.33	64.39	81.74	80.43	81.90	80.07	82.09	77.97

scribed in Section 3.1; CameraBench provides binary labels. The loop experiments use four fixed subsets ($n = 75/95/74/92$), while the scaling study contains 334 items. Appendices A.1 and A.6 give the composition and evaluation protocols.

Metrics. We report accuracy, F1, and balanced accuracy (bAcc), using bAcc to compare RMs under class imbalance. We also report the sample-level micro-averaged performance. For the loop measure, we ask human graders to judge if the final returned video is aligned or not, and then report the averaged performance across all graders. The baseline labels define the subsets used to measure fixes and regressions. For test-time scaling, oracle pass@k is the fraction of requests with at least one human-aligned clip among the first k candidates; we report it separately from final alignment.

Baselines and implementation. We compare constant and optical-flow-rule baselines, untuned open and proprietary VLMs, published camera-motion models, geometric RMs, and FilmBench's camera-movement operator. VLMs see frames sampled at 2 fps. Geometry baselines use 2 and 24 fps to test frame density. Appendices A.2 and A.11 give configurations and frame-rate results.

Human grading. We evaluate loop outputs against the requested camera motion using a rating interface. On the Veo 3.1 image-to-video subset, we compare loops with different RMs (Table 3); Figure 3a reports alignment across the four generated subsets. We also compare our loop with

Table 2: Gemini 3.5 Flash ablation on the frozen 1,064-clip set (size-weighted rates, %). V2 tests flow text versus arrow maps; the cue-only V4 row removes all video inputs. SFT uses binary targets unless marked as distilled-CoT supervision.

	Components						5-dataset weighted (%)			
Variant	Video	Prompt+CoT	Flow	Geom	SFT	RL	Acc	Spec	Rec	bAcc
Vanilla	✓						70.87	54.90	82.44	68.67
V1	✓	✓					72.56	79.63	64.65	72.14
V2	✓	✓	✓				72.65	81.00	64.59	72.80
V2 + *visual optical flow*	✓	✓	✓				73.03	76.35	68.19	72.27
V3	✓	✓	✓	✓			75.57	76.01	72.65	74.33
V4		✓	✓	✓			65.41	84.42	48.39	66.40
Training scheme										
answer-only SFT	✓	✓	✓		✓		78.38	71.00	81.43	76.21
distilled-CoT SFT	✓	✓	✓		✓		74.25	73.39	71.77	72.58
Vanilla + RL	✓					✓	73.68	55.94	83.02	69.48
CamRM-G	✓	✓	✓	✓		✓	**80.73**	**78.16**	**80.26**	**79.21**

resampling under different sampling budgets (Figure 4). At least three outside graders with no prior knowledge provide the ratings. Before grading, we train them to recognize each camera motion.

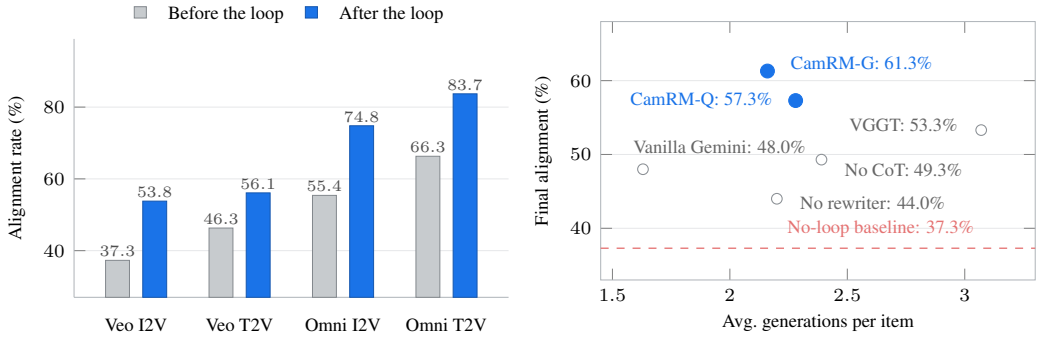
4.1 Camera-motion alignment on generated video (RQ1)

Table 1 reports the main benchmark results. CamRM-G achieves the best average accuracy (80.7%), F1 (82.4%), and bAcc (79.2%), improving bAcc over the strongest baseline, Gemini 3.6 Flash, by 6.6%. CamRM-Q reaches 75.7% bAcc and improves on the best untuned open model by 5.9%. The training-free CamRM-A reaches 78.0% bAcc without tuning on any Veo or Omni clips. This supports the motion-cue design independently of adaptation to either generator. Among the published camera-motion specialists, ACaM Qwen3-VL-8B, also tuned on generated videos, performs best at 68.8% bAcc. A simple optical-flow rule reaches 66.7%, making it a competitive baseline for this task. Additionally, CamRM-G combines 78.2% specificity with 80.3% recall, showing balanced acceptance and rejection. While Gemini 3.6 Flash appears to be the strongest baseline, its specificity is below 60%, indicating frequent acceptance of mismatched clips. The full specificity–recall plot is in Figure 6. Performance may vary by video generator: Gemini 3.1 Pro slightly exceeds CamRM-G on Omni bAcc (70.1% versus 69.5%), although CamRM-G is substantially better on average.

Our CameraBench-only RM uses the same base model, cues, and tuning recipe as CamRM-G, but is trained only on real footage. It achieves 82.3% bAcc on real video, close to CamRM-G's 84.1%, and 75.0% on Veo. On Omni, however, its bAcc falls to 62.0%. Similar drops occur for VGGT (74.3% on real video versus 63.1% on Omni) and Gemini 3.6 Flash (75.6% versus 63.8%). Thus, good real-video performance does not guarantee reliable transfer to a generated-video domain. Training on both sources helps CamRM-G adapt to synthetic video; these results do not establish transfer to unseen generators. Appendix A.13 reports the results by camera-motion category.

4.2 Geometry-based rating models (RQ2)

Geometric RMs estimate camera motion explicitly, but remain less accurate than CamRM-G in our benchmark. VGGT reaches 68.6% bAcc, 10.6% below CamRM-G. Increasing frame density helps MegaSAM, whose dynamic-scene reconstruction can use background evidence despite foreground movement, but it remains sensitive to the available tracks and scene structure. Its bAcc improves from 63.8% to 66.5% at 24 fps, while its runtime rises from 75 to 155 seconds per clip. CUT3R gains no accuracy from the denser input and becomes roughly ten times slower. COLMAP fails to reconstruct about 25% of clips at 2 fps; counting these failures as incorrect yields 42.6% bAcc. These results support motion cues for VLMs over reconstruction alone. Appendix A.7 reports cost and coverage.



(a) (b)

Figure 3: (a) Human-judged alignment before and after the loop on four generated subsets ($n = 75/95/74/92$). (b) Generation cost and final alignment for the six systems in Table 3. The no-rewriter control retains CamRM-G and resamples the original prompt.

Table 3: **Rating-model swap in the motion-guided refinement loop, Veo 3.1 I2V.** We compare six loop configurations on the same items. The baseline uses one generation without an RM or rewriter. CoT indicates whether the RM's reasoning is supplied to the rewriter; VGGT supplies a geometry report without CoT. The no-rewriter configuration retains CamRM-G and resamples the original prompt. Rates are in %. All loop configurations allow three refinement rounds after the initial clip; average generations include all redraws.

Rating model	Rewriter	CoT to Rewriter	Motion alignment acc.	Fix%	Regr%	Avg #gens
—			37.3	—	—	—
VGGT	✓		53.3	31.9	10.7	3.07
Gemini 3.5 Flash	✓	✓	48.0	27.7	17.9	1.63
CamRM-G			44.0	23.4	21.4	2.20
CamRM-G	✓		49.3	36.2	28.6	2.39
CamRM-Q	✓	✓	57.3	38.3	10.7	2.28
CamRM-G	✓	✓	**61.3**	**42.6**	**7.1**	2.16

4.3 Motion cues and training (RQ3)

Table 2 ablates the effects of prompting, cues, and training. The revised prompt and reasoning improve bAcc from 68.7% to 72.1%, and adding flow and global motion raises it to 74.3%. RL without cues gives 69.5%, while RL with the full cues reaches 79.2%. Both the measured motion and task tuning contribute. Frames remain necessary: removing the video from V3 lowers bAcc by 7.9% to 66.4%, with 84.4% specificity but only 48.4% recall. The cue-only model often rejects aligned videos, showing that it needs both video and motion cues. The encoding comparison supports our use of text summaries. Replacing the Gemini V2 flow text with arrow maps lowers bAcc from 72.8% to 72.3% while almost doubling the input sequence length. These comparisons support the complete text cue set. The agentic ablation also shows that frames and motion cues work together (Appendix A.12); Table 7 reports the CamRM-Q ablation.

4.4 Improving the delivered video (RQ4)

Figure 3a shows an average alignment gain of 15.8% across four settings on two generators when using our motion-guided refinement loop, with gains of at least about 10% in each setting. Examples appear in Appendix A.14. Table 3 compares six systems on 75 Veo 3.1 I2V items. The four full-feedback systems change only the RM, with the same rewriter, stopping rule, and cap of three refinement rounds after the initial clip. Replacing vanilla Gemini 3.5 Flash with CamRM-G raises final alignment from 48.0% to 61.3%, a 13.3% gain ($p = 0.021$, paired McNemar test). CamRM-Q reaches 57.3% and is not statistically distinguishable from CamRM-G ($p = 0.55$). Removing the rewriter reduces alignment to 44.0% ($p = 0.007$); withholding the RM's CoT gives 49.3%. The RL-trained RM's reasoning helps the rewriter identify motion errors and guide the next generation. CamRM-G has the highest fix rate (42.6%) and lowest regression rate (7.1%), using 2.16 generations

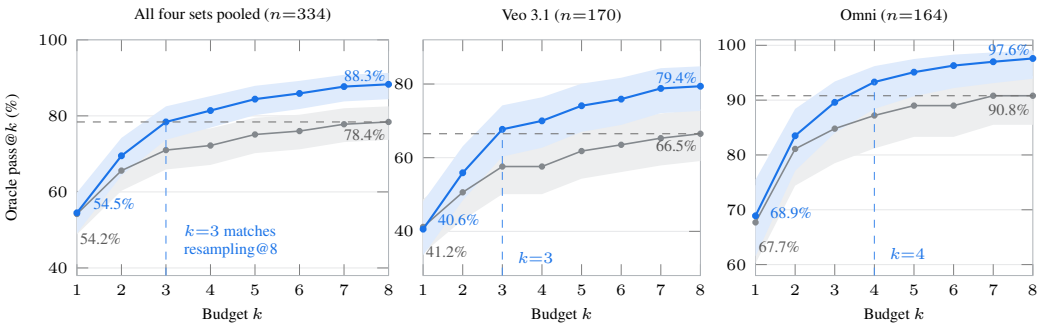


Figure 4: Candidate-set oracle alignment under matched generation budgets. Gray: independent resampling; blue: motion-guided refinement with CamRM-G. Shading shows Wilson 95% confidence intervals. Dashed lines mark the resampling oracle at $k = 8$ and the budget at which the loop reaches it These curves are selected by humans from the candidate set.

Table 4: Transfer to camera-motion classification. Five-dataset average bAcc@1 (%); all tuned models here use binary alignment labels only. Full results and multi-label scoring are in Appendix A.9.

Model family	Untuned	Binary-tuned	Gain (%)
Gemini 3.5 Flash / CamRM-G	38.54	**46.54**	+8.00
Qwen3.5-9B / binary-SFT	17.72	**30.71**	+12.99

per item versus 1.63 for the vanilla RM and 3.07 for VGGT (Figure 3b). Actual use depends on RM stopping decisions. False acceptance ends the vanilla loop early on incorrect clips, while VGGT uses more generations without achieving the best alignment.

4.5 Test-time scaling (RQ4)

At equal generation budgets, the loop improves candidate-set oracle alignment over resampling at every tested budget when pooled, and from $k = 2$ per generator (Figure 4). At $k = 8$, pooled oracle alignment is 88.3% versus 78.4%, a 9.9% gain ($p = 1 \times 10^{-5}$, paired McNemar test). Gains are 12.9% on Veo and 6.7% on Omni. Matching resampling's eight-draw oracle takes three draws when pooled and on Veo, and four on Omni. These savings assume a perfect selector. The deployed loop delivers 69.5% alignment, up 15.3% from the 54.2% single-draw baseline. The 88.3% candidate-set oracle leaves room to improve stopping and selection.

4.6 Transfer from binary alignment to motion classification (RQ5)

Binary alignment training transfers to 20-category motion recognition without classification supervision. CamRM-G improves bAcc@1 from 38.5% to 46.5%, and binary-SFT Qwen improves from 17.7% to 30.7% (Table 4). Demonstrating that CamRM not only learns binary judgment, but also generalizes to motion understanding. CamRM-G also leads the full comparison in Appendix A.9.

5 Discussion and conclusion

Our three CamRM designs show that motion cues improve camera-motion judgment and delivered videos. With the rewriter and search policy fixed, CamRM-G raises alignment from 48.0% to 61.3% over the untuned RM. The loop improves all four settings and matches resampling's pooled eight-generation oracle rate with three generations.

Limitations: We evaluate one controllable attribute and two commercial generators. Our 2D cues do not fully resolve depth-dependent motion. Our evaluation is limited to Veo 3.1 and the Omni Flash model, so transfer to other generators remains to be future work. Evaluating more generators and improving motion interpretation are next steps. As generators improve, the available gains shrink, but reliable rating remains useful for checking the delivered video.

REPRODUCIBILITY STATEMENT

Upon publication, we will release the dataset and its human annotations, the open-source CamRM-Q model weights, and the code for training and evaluation, including motion-cue extraction and the motion-guided refinement loop. Appendices A.1, A.2, A.5 and A.6 describe data construction, model configurations and evaluation protocols. The generators and CamRM-G use commercial APIs; reproduction requires access to the same model versions.

AI USE STATEMENT

We used language and vision-language models to help create the user prompts and generated videos used in this study. We downloaded the I2V reference images from a third-party data provider. We also used AI coding tools to assist with code development, experiment execution, and language editing of the manuscript. Human annotators assessed camera-motion alignment, and the authors checked the experimental results and took responsibility for the final paper.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *CoRR*, abs/2511.21631, 2025. doi: 10.48550/arXiv.2511.21631. URL https://arxiv.org/abs/2511.21631.

Jiale Cheng, Ruiliang Lyu, Xiaotao Gu, Xiao Liu, Jiazheng Xu, Yida Lu, Jiayan Teng, Zhuoyi Yang, Yuxiao Dong, Jie Tang, Hongning Wang, and Minlie Huang. VPO: aligning text-to-video generation models with prompt optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15636–15645. IEEE, 2025. doi: 10.1109/ICCV51701.2025.01451. URL https://openaccess.thecvf.com/content/ICCV2025/html/Cheng_VPO_Aligning_Text-to-Video_Generation_Models_with_Prompt_Optimization_ICCV_2025_paper.html.

Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, volume 2749 of *Lecture Notes in Computer Science*, pp. 363–370. Springer, 2003. doi: 10.1007/3-540-45103-X_50. URL https://doi.org/10.1007/3-540-45103-X_50.

Haoan Feng, Sri Harsha Musunuri, and Guan-Ming Su. Geometry-guided camera motion understanding in VideoLLMs. *CoRR*, abs/2603.13119, 2026. doi: 10.48550/arXiv.2603.13119. URL https://arxiv.org/abs/2603.13119.

Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. doi: 10.1145/358669.358692. URL https://doi.org/10.1145/358669.358692.

Bingjie Gao, Qianli Ma, Xiaoxue Wu, Shuai Yang, Guanzhou Lan, Haonan Zhao, Jiaxuan Chen, Qingyang Liu, Yu Qiao, Xinyuan Chen, Yaohui Wang, and Li Niu. RAPO++: cross-stage prompt optimization for text-to-video generation via data alignment and test-time scaling. *CoRR*, abs/2510.20206, 2025a. doi: 10.48550/arXiv.2510.20206. URL https://arxiv.org/abs/2510.20206.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft adaptive policy optimization. *CoRR*, abs/2511.20347, 2025b. doi: 10.48550/arXiv.2511.20347. URL https://arxiv.org/abs/2511.20347.

Xuan He, Dongfu Jiang, Ge Zhang, Max Ku, Achint Soni, Sherman Siu, Haonan Chen, Abhranil Chandra, Ziyan Jiang, Aaran Arulraj, Kai Wang, Quy Duc Do, Yuansheng Ni, Bohan Lyu, Yaswanth Narsupalli, Rongqi Fan, Zhiheng Lyu, Bill Yuchen Lin, and Wenhu Chen. VideoScore: Building automatic metrics to simulate fine-grained human feedback for video generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2105–2123. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.127. URL https://aclanthology.org/2024.emnlp-main.127/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626. ACM, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Zhengqi Li, Richard Tucker, Forrester Cole, Qianqian Wang, Linyi Jin, Vickie Ye, Angjoo Kanazawa, Aleksander Holynski, and Noah Snavely. MegaSaM: Accurate, fast and robust structure and motion from casual dynamic videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10486–10496. Computer Vision Foundation / IEEE, 2025. doi: 10.1109/CVPR52734.2025.00981. URL https://openaccess.thecvf.com/content/CVPR2025/html/Li_MegaSaM_Accurate_Fast_and_Robust_Structure_and_Motion_from_Casual_CVPR_2025_paper.html.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *Computer Vision – ECCV 2024*, volume 15067 of *Lecture Notes in Computer Science*, pp. 366–384. Springer, 2024. doi: 10.1007/978-3-031-72673-6_20. URL https://doi.org/10.1007/978-3-031-72673-6_20.

Zhiqiu Lin, Siyuan Cen, Daniel Jiang, Jay Karhade, Hewei Wang, Chancharik Mitra, Yu Tong Tiffany Ling, Yuhan Huang, Rushikesh Zawar, Xue Bai, Yilun Du, Chuang Gan, and Deva Ramanan. Towards understanding camera motions in any video. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-4477. URL https://papers.nips.cc/paper_files/paper/2025/hash/c2d7b739f61ffbea729dad2cf9ec8c60-Abstract-Datasets_and_Benchmarks_Track.html.

Fangfu Liu, Hanyang Wang, Yimo Cai, Kaiyan Zhang, Xiaohang Zhan, and Yueqi Duan. Video-T1: Test-time scaling for video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18671–18681. IEEE, 2025a. doi: 10.1109/ICCV51701.2025.01735. URL https://openaccess.thecvf.com/content/ICCV2025/html/Liu_Video-T1_Test-time_Scaling_for_Video_Generation_ICCV_2025_paper.html.

Hongbo Liu, Jingwen He, Yi Jin, Dian Zheng, Yuhao Dong, Fan Zhang, Ziqi Huang, Yinan He, Weichao Chen, Yu Qiao, Wanli Ouyang, Shengjie Zhao, and Ziwei Liu. ShotBench: Expert-level cinematic understanding in vision-language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 129987–130019. Curran Associates, Inc., 2025b. doi: 10.52202/085713-4329. URL https://papers.nips.cc/paper_files/paper/2025/hash/bc94f2e93a8b05573e8e787d35665ca6-Abstract-Conference.html.

Do Xuan Long, Xingchen Wan, Hootan Nakhost, Chen-Yu Lee, Tomas Pfister, and Sercan Ö. Arık. VISTA: A test-time self-improving video generation agent. *CoRR*, abs/2510.15831, 2025. doi: 10.48550/arXiv.2510.15831. URL https://arxiv.org/abs/2510.15831.

Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pp. 674–679. William Kaufmann, 1981.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4104–4113. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.445. URL https://openaccess.thecvf.com/content_cvpr_2016/html/Schonberger_Structure-From-Motion_Revisited_CVPR_2016_paper.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314, 2024. doi: 10.48550/arXiv.2408.03314. URL https://arxiv.org/abs/2408.03314.

Tingyu Song, Tongyan Hu, Guo Gan, and Yilun Zhao. VF-Eval: Evaluating multimodal LLMs for generating feedback on AIGC videos. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21126–21146. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.1027. URL https://aclanthology.org/2025.acl-long.1027/.

Yuwen Tan, Joey Huang, Jin Huang, Haoxiang Li, and Boqing Gong. Natural language camera movement understanding. *CoRR*, abs/2607.03043, 2026. doi: 10.48550/arXiv.2607.03043. URL https://arxiv.org/abs/2607.03043.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotný. VGGT: visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306. Computer Vision Foundation / IEEE, 2025a. doi: 10.1109/CVPR52734.2025.00499. URL https://openaccess.thecvf.com/content/CVPR2025/html/Wang_VGGT_Visual_Geometry_Grounded_Transformer_CVPR_2025_paper.html.

Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. Continuous 3D perception model with persistent state. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10510–10522. Computer Vision Foundation / IEEE, 2025b. doi: 10.1109/CVPR52734.2025.00983. URL https://openaccess.thecvf.com/content/CVPR2025/html/Wang_Continuous_3D_Perception_Model_with_Persistent_State_CVPR_2025_paper.html.

Shengyi Wang, Niantong Li, Guangzheng Hu, Hong Qi, Fei Ding, Weixu Qiao, Jinlin Wang, Xiaotong Lv, Peng Han, Zimeng Li, Fanshu Ding, Yushu Wang, Han Wu, Jingjing Chen, Chongxiao Wang, Yanhao Wu, Chenglong Huang, Xiaoqian Zhu, Jie Tian, Hua Li, Jingjing Fan, Mingshuang Tang, Zhong Li, Hengxia Qiang, Weibin Chen, Jinyang Zhen, Bing Zhao, Lin Qu, Jing Li, and Hu Wei. FilmBench: A film-grade benchmark for cinematic video generation. *CoRR*, abs/2607.24241, 2026. doi: 10.48550/arXiv.2607.24241. URL https://arxiv.org/abs/2607.24241.

Xinran Wang, Songyu Xu, Xiangxuan Shan, Yuxuan Zhang, Muxi Diao, Xueyan Duan, Yanhua Huang, Kongming Liang, and Zhanyu Ma. CineTechBench: A benchmark for cinematographic technique understanding and generation. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025c. doi: 10.52202/085713-1810. URL https://papers.nips.cc/paper_files/paper/2025/hash/4e17f622f4259d0d64331310d2674902-Abstract-Datasets_and_Benchmarks_Track.html.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. *CoRR*, abs/2507.13347, 2025d. doi: 10.48550/arXiv.2507.13347. URL https://arxiv.org/abs/2507.13347.

Hang Wu, Yujun Cai, Zehao Li, Haonan Ge, Bowen Sun, Junsong Yuan, and Yiwei Wang. CamReasoner: Reinforcing camera movement understanding via structured spatial reasoning. *CoRR*, abs/2602.00181, 2026. doi: 10.48550/arXiv.2602.00181. URL https://arxiv.org/abs/2602.00181.

Jiazheng Xu, Yu Huang, Jiale Cheng, Yuanming Yang, Jiajun Xu, Yuan Wang, Wenbo Duan, Shen Yang, Qunlin Jin, Shurun Li, Jiayan Teng, Zhuoyi Yang, Wendi Zheng, Xiao Liu, Dan Zhang, Ming Ding, Xiaohan Zhang, Shiyu Huang, Xiaotao Gu, Minlie Huang, Jie Tang, and Yuxiao Dong. VisionReward: Fine-grained multi-dimensional human preference learning for image and video generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(13):11269–11277, 2026. doi: 10.1609/AAAI.V40I13.38107. URL https://ojs.aaai.org/index.php/AAAI/article/view/38107.

Jianhao Yuan, Xiaofeng Zhang, Felix Friedrich, Nicolas Beltran-Velez, Melissa Hall, Reyhane Askari Hemmat, Xiaochuang Han, Nicolas Ballas, Michal Drozdzal, and Adriana Romero-Soriano. Inference-time physics alignment of video generative models with latent world models. *CoRR*, abs/2601.10553, 2026. doi: 10.48550/arXiv.2601.10553. URL https://arxiv.org/abs/2601.10553.

Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. MonST3R: A simple approach for estimating geometry in the presence of motion. In *The Thirteenth International Conference on Learning Representations*. OpenReview.net, 2025. URL https://openreview.net/forum?id=lJpqxFgWCM.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. SWIFT: A scalable lightweight infrastructure for fine-tuning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(28):29733–29735, 2025. doi: 10.1609/AAAI.V39I28.35383. URL https://ojs.aaai.org/index.php/AAAI/article/view/35383.

Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Lulu Gu, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, Yu Qiao, and Ziwei Liu. VBench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *CoRR*, abs/2503.21755, 2025. doi: 10.48550/arXiv.2503.21755. URL https://arxiv.org/abs/2503.21755.

A APPENDIX

A.1 BENCHMARK CONSTRUCTION AND DATASET COMPOSITION

We construct the benchmark to evaluate camera-motion alignment on both real and generated video. Including two generators and both image-to-video and text-to-video tasks allows us to examine whether an RM's performance transfers across video sources. Table 5 gives the composition of the $1{,}064$-clip evaluation set.

We construct prompts for 20 camera-motion categories following Section 3.1. For text-to-video, we use Gemini 3.1 Pro to generate 30 prompts per category, each containing a scene description and a separate camera-motion instruction. For image-to-video, we download 30 real reference images from a third-party data provider under a license that permits redistribution and use the Gemini model to write a video generation prompt for each image and requested camera motion. This gives 600 image-to-video settings and 600 text-to-video settings. We tag confusable pairs, including pan and truck, tilt and pedestal, and zoom and dolly, for the category analysis.

We generate clips through the public endpoints of Veo 3.1 and Omni at each provider's default quality setting. Veo clips are $8\,\mathrm{s}$ at 720p, and Omni clips are $10\,\mathrm{s}$. Across both generators, we generate $2{,}400$ clips and retain $2{,}011$ after filtering scene changes and ambiguous annotations. We split prompts $70/30$ into training and evaluation partitions, stratified by camera-motion category, and keep all clips from the same prompt in one partition. For I2V, all clips sharing a source image remain in the same partition, so training and evaluation use disjoint source images. We combine the generated training partition with the CameraBench training clips for RM adaptation. The 571 held-out generated clips and 493 real CameraBench clips form the evaluation set.

In the retained evaluation set, human labels mark $210/359$ Veo clips (58.50%) and $144/212$ Omni clips (67.92%) as aligned, leaving 149 and 68 misaligned clips, respectively. The aligned proportions are 47.28% and 70.29% for Veo I2V and T2V, and 64.17% and 72.83% for Omni I2V and T2V, respectively.

The classification task uses 20 categories. For the per-category analysis in Appendix A.13, we retain the 23 source labels used across the datasets: `pan_left`, `pan_right`, `tilt_up`, `tilt_down`, `truck_left`, `truck_right`, `pedestal_up`, `pedestal_down`, `move_up`, `dolly_in`, `dolly_out`, `zoom_in`, `zoom_out`, `roll`, `roll_clockwise`, `roll_counterclockwise`, `orbit_left`, `orbit_right`, `orbit_over`, `orbit_under`, `360_spin`, `dolly_zoom`, and `static`. These source labels are distinct from the classification label space.

Table 5: Dataset composition. We retain $2{,}011$ generated clips after filtering and split them $70/30$ by prompt, stratified by motion category. The held-out generated clips and real CameraBench clips form the same $1{,}064$-clip evaluation set for every rating model.

Split	Source	Train	Eval	Human labels
Veo 3.1, image-to-video	generated	439	184	5 raters, 0–2, aligned iff mean > 1
Veo 3.1, text-to-video	generated	432	175	*as above*
Omni, image-to-video	generated	311	120	*as above*
Omni, text-to-video	generated	258	92	*as above*
CameraBench - Binary	real	1388	493	explicit binary
Total	both	**2,828**	**1,064**	—

We evaluate every RM on the same frozen set. In Table 1, the generator columns combine image-to-video and text-to-video results using dataset size as the weight. We also use dataset-size weights for the aggregate rates. This convention differs from pooling confusion counts: pooled specificity would weight each dataset by its number of negative examples. We report parse failures separately so that readers can distinguish classification errors from failures to return a usable verdict.

Table 6: Model configurations. Flow includes the spatial grid and parallax summary; GM denotes the global-motion cue. The binary-SFT classification variant is a separate checkpoint from the final SAPO rating model.

Variant	Backbone	Adaptation	Inputs and use
CamRM-G	Gemini 3.5 Flash	Provider API, GRPO-style single-step RL with length penalty	Frames, prompt, flow and GM; binary rating
CamRM-Q (SAPO)	Qwen3.5-9B	Binary SFT, then SAPO	Frames, prompt, flow and GM; binary rating
Cue-SFT variants	Qwen3.5-9B	Binary SFT, matched recipe	Frames, prompt and the cue set in Table 7
Binary-SFT classification variant	Qwen3.5-9B	Binary SFT; no classification tuning	20-way motion query; classification transfer
CamRM-A	Gemini 3.5 Flash	No parameter updates	Frames, prompt and callable motion tools; binary rating

A.2 Training and inference details

We evaluate whether motion cues improve RMs under different training settings and whether binary alignment training transfers to motion classification. Table 6 separates the final RMs from the supervised variants used in the cue ablation and classification experiments. These distinctions matter because the cue ablations use matched SFT recipes, while the final CamRM-Q also receives reinforcement learning.

Motion cues. Appendix A.3 defines the optical-flow grid, regional summaries, global-motion estimates, and their text representation. The arrow-map variant in Section 4.3 renders the same flow field as images, using roughly 32 images per record compared with 16 frames for the text variants. Before training, we compare the open-model cue implementation with the reference implementation. All 1600 text renders match exactly, and the underlying arrays agree to 1.5×10^{-4}. This check allows us to compare model families using the same cue representation.

Mismatch augmentation. For each positive training example, we create one additional negative example by replacing its camera-motion instruction with a different, mismatched instruction. We keep the video fixed and assign the new pair label 0. This augmentation teaches the RM to compare the requested motion with the motion in the video.

Supervised fine-tuning of the open RM. We train Qwen3.5-9B (Qwen Team, 2026) with ms-swift (Zhao et al., 2025) on one node with 8×H100 GPUs. We use LoRA with rank 4 and $\alpha = 16$, a learning rate of 10^{-4}, and 5 epochs. The per-device batch size is 1, with gradient accumulation over 4 steps under DDP. We set the maximum sequence length to 16,384 and apply supervision only to the answer. All four supervised cue ablations use this recipe, so their comparison isolates the cue input.

Reinforcement learning of the open RM. We train the final CamRM-Q with SAPO (Gao et al., 2025b) after binary SFT. The model receives frames, the prompt, optical flow, and global motion. We keep the same LoRA configuration and train for up to 400 steps, with an initial learning rate of 10^{-5} and a KL coefficient of 0.02. The maximum sequence length is 20,480, including at most 8,192 completion tokens. We sample 8 rollouts per prompt at temperature 1.0 and serve them with vLLM (Kwon et al., 2023). The reward combines binary correctness against the human label, a format term requiring reasoning and verdict tags, and a length term. It uses no learned reward model. SAPO uses soft-gate temperatures $\tau_+ = 1.0$ and $\tau_- = 1.05$. Appendix A.4 gives the rewards and loss functions.

Training the proprietary RM. We adapt Gemini 3.5 Flash through the provider's public tuning API, using a GRPO-style single-step reinforcement-learning objective with a length penalty managed by the provider's training infrastructure. The inputs are frames, the prompt, optical flow, and global motion. We sample 8 rollouts per training example and use a binary correctness reward. We report the checkpoint at step 395 for CamRM-G and step 280 for the RL control without cues. The

two SFT rows in Table 2 use the V2 cue set. One uses answer-only targets, and the other uses chains of thought distilled from a stronger model, allowing us to compare the two supervision schemes. **Validation and checkpoint selection.** We reserve 15% of the training partition for validation and select checkpoints by validation performance. We evaluate every 20 training steps and stop after five consecutive validation evaluations without improvement. We report the checkpoint with the best validation performance. The $1{,}064$-example test set is separate from this validation subset and is not used for checkpoint selection.

Inference. Single-call VLM raters receive frames at 2 fps and decode greedily, producing reasoning followed by a binary verdict. CamRM-A gathers evidence through several tool calls, as described in Appendix A.12. We also evaluate geometry baselines at 24 fps to test whether denser frames improve their motion estimates. We count verdicts that fail to parse as unscored and report them through the parse rate. The model comparisons therefore retain the distinction between prediction quality and output coverage.

A.3 Motion-cue computation and text representation

We compute both cues from the original video. Image coordinates increase to the right and downward. Flow vectors describe image-content displacement; for a pan or lateral translation, content generally moves opposite to the camera.

A.3.1 Optical-flow grid and regional summaries

Sampling and preprocessing. We sample frames at $f_{\mathrm{calc}} = 4$ fps. For native frame rate f_{nat}, sample k uses frame index $\mathrm{round}(kf_{\mathrm{nat}}/f_{\mathrm{calc}})$. We resize each frame with bilinear interpolation so that its longer side is 512 pixels, rounding each resulting dimension up to an even integer, and convert it to grayscale. All flow magnitudes below use pixels in this resized image. Farneback flow (Farnebäck, 2003) uses pyramid scale 0.5, 2 pyramid levels, window size 15, 2 iterations, polynomial neighborhood size 5, polynomial standard deviation 1.2, and flags 0.

Let $\mathbf{f}_k(x) \in \mathbb{R}^2$ be the forward flow at pixel x from sampled frame k to $k+1$. We average adjacent flow fields to form a summary at 2 fps:

$$\widetilde{\mathbf{f}}_t(x) = \frac{1}{|S_t|} \sum_{k \in S_t} \mathbf{f}_k(x), \qquad S_t = \{2t, 2t+1\} \cap \{0, \dots, K-2\}, \tag{1}$$

where K is the number of sampled frames and $t = 0, \dots, N-1$, with $N = \lceil (K-1)/2 \rceil$. The final group can contain one field. Averaging preserves the displacement unit of one 0.25 s flow step; the reported values are not displacements accumulated over 0.5 s.

Cell magnitude and direction. For a resized image of height H and width W, the grid uses row boundaries $(0, \lfloor H/3 \rfloor, 2\lfloor H/3 \rfloor, H)$ and analogous column boundaries. Thus, the last row and column absorb remainder pixels. For cell C_{rc}, $r, c \in \{0, 1, 2\}$, define

$$\boldsymbol{\mu}_{t,rc} = \frac{1}{|C_{rc}|} \sum_{x \in C_{rc}} \widetilde{\mathbf{f}}_t(x), \qquad m_{t,rc} = \|\boldsymbol{\mu}_{t,rc}\|_2. \tag{2}$$

The magnitude is the norm of the mean vector. Opposing motions within a cell can therefore cancel. If $m_{t,rc} < 1$ px, the direction is `NONE`. Otherwise, with $\theta_{t,rc} = \mathrm{atan2}(\mu^y_{t,rc}, \mu^x_{t,rc})$, its direction index is

$$q_{t,rc} = \left\lfloor \frac{\theta_{t,rc} + \pi/8}{\pi/4} \right\rfloor \bmod 8. \tag{3}$$

Indices 0 through 7 map to `RIGHT`, `DOWN-RIGHT`, `DOWN`, `DOWN-LEFT`, `LEFT`, `UP-LEFT`, `UP`, and `UP-RIGHT`. Each sector is 45° wide and includes its lower boundary.

Outer and center motion. Let $\mathcal{O} = \{0, 1, 2\}^2 \setminus \{(1, 1)\}$ be the eight outer cells. We compute

$$B_t = \frac{1}{8} \sum_{(r,c) \in \mathcal{O}} m_{t,rc}, \qquad C_t = m_{t,11}, \qquad \mathbf{b}_t = \frac{1}{8} \sum_{(r,c) \in \mathcal{O}} \boldsymbol{\mu}_{t,rc}. \tag{4}$$

The window covers the full clip, or a requested time range for the agentic tools. We give all eight outer cells equal weight. The text reports the window means $\bar{B} = N^{-1} \sum_t B_t$ and $\bar{C} = N^{-1} \sum_t C_t$

separately; it does not compute or serialize their ratio. These values remain in resized-image pixels, without division by frame width or height. Outer and center locations provide a coarse spatial comparison, not a measured depth ordering.

The horizontal and vertical divergence cues are differences between opposing grid columns and rows:

$$d_t^x = \frac{1}{3}\sum_{r=0}^{2}(\mu_{t,r2}^x - \mu_{t,r0}^x),$$
$$d_t^y = \frac{1}{3}\sum_{c=0}^{2}(\mu_{t,2c}^y - \mu_{t,0c}^y). \tag{5}$$

Positive values indicate expansion along the corresponding axis, and negative values indicate contraction. These are displacement differences without division by spatial separation. We average them over the window to obtain $\bar{d}^x$ and $\bar{d}^y$, and report $\bar{\mathbf{b}} = N^{-1}\sum_t \mathbf{b}_t$ as the mean outer translation. Each interval has equal weight.

Interpretation text. The summary includes a short rule-based interpretation. It labels outer motion as static when $\bar{B} < 1\,\text{px}$. Otherwise, define $a = |\bar{d}^x| + |\bar{d}^y|$ and $b = \|\bar{\mathbf{b}}\|_2$. When $a > 1.5b$ and $a > 1\,\text{px}$, it describes radial motion: outward when $\bar{d}^x + \bar{d}^y > 0$, and inward otherwise. The remaining case is described as approximately uniform translation. Outward and inward motion suggest zoom/dolly in and out, respectively; translation suggests pan/tilt or truck/pedestal. The RM uses the frames and regional magnitudes to interpret these alternatives.

A.3.2 Global-motion estimation

Tracking and fitting. We sample the original video with frame-index stride $\max(1, \text{round}(f_{\text{nat}}/4))$. Frames whose longer side exceeds 512 pixels are downscaled with bilinear interpolation, with integer dimensions obtained by rounding down. We convert frames to grayscale and detect corners afresh in the first frame of each pair. Shi–Tomasi detection uses at most 400 corners, quality level 0.01, minimum distance 8 pixels, and block size 7. Pyramidal Lucas–Kanade tracking uses a 21×21 window, maximum pyramid level 3, and termination after 30 iterations or an update below 0.01.

We retain tracks with a successful status flag and fit a partial-affine model with RANSAC. The reprojection threshold is 3 resized-image pixels, with at most 2,000 iterations, confidence 0.99, and 10 refinement iterations. Pairs with fewer than six detected corners or successful tracks, or a failed affine fit, are skipped. No further inlier-count threshold is applied. If all pairs fail, the cue reports insufficient trackable features.

Accumulation. For each valid pair $k \in \mathcal{V}$, the forward transform is

$$\mathbf{x}' = s_k R(\theta_k)\mathbf{x} + \mathbf{u}_k, \qquad A_k = \begin{bmatrix} a_k & -b_k & u_k^x \\ b_k & a_k & u_k^y \end{bmatrix}, \tag{6}$$

where $s_k = \sqrt{a_k^2 + b_k^2}$ and $\theta_k = \text{atan2}(b_k, a_k)$. A zero scale estimate is replaced by 1. The clip-level summaries are

$$\mathbf{U} = \sum_{k\in\mathcal{V}} \mathbf{u}_k, \quad S = \prod_{k\in\mathcal{V}} s_k, \quad \Theta = \frac{180}{\pi}\sum_{k\in\mathcal{V}} \theta_k,$$
$$h = \frac{U_x}{W}, \qquad v = \frac{U_y}{H}, \qquad z = S - 1. \tag{7}$$

Here W, H are the dimensions used for tracking and Θ is in degrees. Translation is accumulated componentwise; we do not compose the affine matrices. Also, $\mathbf{u}_k$ is relative to the top-left image origin. A scale or roll about the image center $\mathbf{c}$ contributes $(I - s_k R(\theta_k))\mathbf{c}$ to this translation term. The summaries therefore describe dominant image motion rather than a recovered 3D camera trajectory.

Thresholds and ranking. Horizontal and vertical motion are significant when $|h| > 0.04$ and $|v| > 0.04$, respectively. Positive h means content shifts right and suggests camera motion left; positive v suggests camera motion up. Scale change is significant when $|z| > 0.06$: positive values suggest

zoom/dolly in and negative values suggest zoom/dolly out. Roll is reported when $|\Theta| > 2.5°$, using its absolute angle. With downward image coordinates, a positive fitted angle denotes clockwise content rotation.

We rank components by $D_{\text{trans}} = |h| + |v|$, $D_{\text{scale}} = |z|$, and $D_{\text{roll}} = |\Theta|/45$. Scores above 10^{-3} are listed in descending order. This ranking uses a smaller inclusion threshold than the per-component significance tests, so a weak component can appear in the ranked list while its description says it is not significant.

A.3.3 TEXT FORMAT AND TIMESTAMPS

Each flow interval lists the nine cells in row-major order, from `Top-Left` to `Bottom-Right`. Magnitudes use one decimal place and directions use the labels in Equation (3). The window summary gives $\bar{B}$, $\bar{C}$, $\bar{d}^x$, $\bar{d}^y$, and both components of $\bar{\mathbf{b}}$ to two decimal places, followed by the interpretation text. Global-motion text gives the time window and number of valid pairs, horizontal and vertical shifts as percentages, scale change as a percentage, roll in degrees, and the component ranking. Percentages and roll use one decimal place. The following excerpt illustrates the field format; intermediate cells and intervals are omitted.

```
--- Time interval:  0.0s to 1.0s ---
Top-Left:  Magnitude 5.4 px, Dominant Direction RIGHT
Center:  Magnitude 1.8 px, Dominant Direction RIGHT
[other cells and intervals omitted]
[Parallax summary over window]
background(outer) mean magnitude = 4.87 px;
center(foreground) mean magnitude = 3.77 px;
horizontal divergence = -1.76; vertical divergence = -1.04;
mean background translation = (+3.94, +0.10).
```

The global cue advises the RM to judge the dominant component, allowing small secondary motions, and to consult regional flow and frames for pan/truck, tilt/pedestal, and zoom/dolly ambiguities. These instructions accompany the numerical measurements.

For CamRM-G, we slow playback by a factor of two so that the provider's 1 fps video ingestion gives two frames per original second. Flow is still computed from the original clip, but its interval labels follow the slowed video: an original 8 s clip has flow labels spanning 16 s at a displayed rate of 1 fps. Global-motion timestamps remain in original-video seconds. Exported text cues retain this convention, whereas live extraction for the open model and agentic tools uses original-video timestamps at 2 fps. This changes the displayed time scale, not the flow magnitudes.

In the single-call models, the user input contains the video, rating instructions and requested motion, flow text, and global-motion text in that order. The agentic model receives the cues through tool responses when it requests them.

A.4 REINFORCEMENT-LEARNING REWARDS AND OBJECTIVES

We use verifiable binary labels to train both rating models. Let x contain the video, requested motion, and any motion cues, and let $y \in \{0, 1\}$ be its human label. For each training example, we sample $G = 8$ responses $o_1, \ldots, o_G$ from the rollout policy π_{old}. Each response contains reasoning followed by a binary verdict. We first define the response rewards, then the GRPO and SAPO losses used to turn those rewards into policy updates.

A.4.1 CORRECTNESS, FORMAT, AND LENGTH REWARDS

Let $\widehat{y}(o)$ be the parsed verdict. The correctness reward is

$$R_{\text{acc}}(o, y) = \mathbb{I}[\widehat{y}(o) = y], \tag{8}$$

with reward 0 when no valid verdict can be parsed. For CamRM-G, this is the task reward; the provider's training infrastructure also applies a length reward to control response length.

For CamRM-Q, the requested format is `<think>reasoning</think>` followed by `<answer>0</answer>` or `<answer>1</answer>`. The parser also accepts `yes`/`no` as binary verdicts. The format reward R_{fmt} is 1 for a response that follows the required structure and

0 otherwise; a response that fails this format check also receives correctness reward 0. Let n be the number of Qwen-tokenizer tokens inside the first closed reasoning block, excluding the tags and special tokens. If there is no closed reasoning block, $R_{\text{len}} = -1$. Otherwise, the length reward is

$$R_{\text{len}}(o) = \begin{cases} 1, & n \le 1024, \\ 1 - \dfrac{n - 1024}{2048 - 1024}, & 1024 < n \le 2048, \\ -1, & n > 2048. \end{cases} \tag{9}$$

This term penalizes longer reasoning while leaving short responses equally rewarded. The total response reward is

$$R_Q(o, y) = R_{\text{acc}}(o, y) + 0.1R_{\text{fmt}}(o) + 0.5R_{\text{len}}(o). \tag{10}$$

The reward uses the human label and the response format and length; it does not use a learned reward model.

A.4.2 Group-relative policy updates

For rewards $R_1, \ldots, R_G$, the advantage of response i and its token-level policy ratio are

$$\widehat{A}_i = \frac{R_i - \bar{R}}{s_R + \varepsilon_{\text{num}}}, \qquad \bar{R} = \frac{1}{G}\sum_{j=1}^{G} R_j, \qquad r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid x, o_{i,<t})}{\pi_{\text{old}}(o_{i,t} \mid x, o_{i,<t})}, \tag{11}$$

where s_R is the within-group reward standard deviation and $\varepsilon_{\text{num}} > 0$ ensures numerical stability. All tokens in a response share its advantage. A group with identical rewards has zero advantages.

GRPO. The standard GRPO loss (Shao et al., 2024) clips the policy ratio to limit the contribution of large changes from the rollout policy:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big(-\min\Big\{r_{i,t}\widehat{A}_i, \text{clip}(r_{i,t}, 1-\epsilon, 1+\epsilon)\widehat{A}_i\Big\} + \beta d_{i,t}\Big)\right]. \tag{12}$$

The expectation is over training examples and sampled response groups. Here ϵ is the clipping range and β controls regularization toward a fixed reference policy π_{ref}. A token-level KL estimator is

$$d_{i,t} = q_{i,t} - \log q_{i,t} - 1, \qquad q_{i,t} = \frac{\pi_{\text{ref}}(o_{i,t} \mid x, o_{i,<t})}{\pi_\theta(o_{i,t} \mid x, o_{i,<t})}. \tag{13}$$

We give the standard loss to describe the GRPO-style update underlying CamRM-G; its optimizer settings and length control are managed by the provider.

SAPO. CamRM-Q uses SAPO (Gao et al., 2025b), which replaces the clipped term with a smooth function of the same token-level ratio. Following the SAPO formulation and its ms-swift implementation,[1] define

$$g_{\tau_i}(r) = \frac{4}{\tau_i}\sigma(\tau_i(r-1)), \qquad \tau_i = \begin{cases} 1.0, & \widehat{A}_i > 0, \\ 1.05, & \widehat{A}_i \le 0, \end{cases} \qquad \sigma(z) = \frac{1}{1+e^{-z}}. \tag{14}$$

The loss is

$$\mathcal{L}_{\text{SAPO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big(-g_{\tau_i}(r_{i,t})\widehat{A}_i + \beta d_{i,t}\Big)\right], \qquad \beta = 0.02. \tag{15}$$

At $r = 1$, $g'_{\tau_i}(r) = 1$. As the ratio moves away from 1, this derivative decreases smoothly, reducing the update without a hard clipping boundary. The larger temperature for negative advantages makes their updates decay faster. Both objectives use only generated response tokens in the loss; prompt tokens and padding are excluded. For the open model, responses truncated at the completion limit are also excluded from the loss.

[1] ms-swift SAPO documentation; Qwen SAPO technical note.

A.5 HUMAN GRADING PROTOCOL

We use human ratings to evaluate whether the returned video follows the requested camera motion. For individual-clip grading, the interface presents clips in a 2×4 multi-select grid alongside the motion request. Raters mark each clip as aligned, misaligned, or a scene-change failure.

To measure consistency, we repeat pages with the clips in a different order. Per-clip repeat agreement is 87.5% (70 of 80), while whole-page exact agreement is 5 of 10. Agreement is lower on Omni text-to-video than on the other splits. These measurements show that the grading task includes ambiguous cases and that reliability varies across video sources.

For every loop experiment, we define baseline alignment by a majority vote over the original dataset annotations and two independent re-graders. An item is baseline-aligned when at least two of the three sources call it aligned. We use these labels to define the subsets for fix and regression rates, keeping the starting labels consistent across loop configurations.

A.5.1 GRADING INTERFACE

We use the grading interface to present the requested camera motion, videos, and motion definitions together. Figure 5 shows its A/B comparison view. The two clips appear side by side as Option A and Option B under the same target motion. Graders can select the better match, mark both clips as good, or indicate that neither matches. These choices allow a grader to record cases where both candidates succeed or both fail without having to prefer one.

The reference panel below the clips illustrates the camera motions and distinguishes commonly confused categories, including dolly versus zoom and truck versus pan. Graders can consult this panel while evaluating the videos. The example in Figure 5 asks for a roll and highlights that motion in the reference panel.

A.6 REFINEMENT-LOOP AND PASS@k PROTOCOLS

We evaluate two aspects of test-time control: the alignment of the video returned by the motion-guided refinement loop and the probability that its candidate set contains an aligned video. We use the former to measure delivered performance and the latter to compare search strategies under equal generation budgets.

Deployed loop. All controlled configurations allow up to three refinement rounds after the initial clip, with the same stopping rule and scene-change check. The RM decides whether to accept each candidate or continue refinement. The no-rewriter configuration runs the same loop implementation in rejection-sampling mode. It retains CamRM-G and rates every candidate. After a rejection, it independently samples the original prompt again. This comparison measures the contribution of rewriting with the RM held fixed. Both scene-change redraws and generations following a rewrite count towards the budget and the average generation cost.

Pass@k evaluation. We evaluate 334 items across the four generated splits, using a frozen test split verified to be disjoint from RM fine-tuning data. Each item has a budget of 8 generations. Scene-change retries count towards this budget; auxiliary RM calls do not, because they produce no new candidate. We share the first candidate between the two strategies, using the same prompt and seed, so candidate identity is matched at $k = 1$. A candidate counts as successful only when it is aligned and free of scene changes, following the human rubric.

These protocols support the separate results in Sections 4.4 and 4.5. The candidate-set oracle measures whether search produced a successful candidate. Delivered alignment also depends on the RM's acceptance decisions and the stopping rule, so the oracle rate does not by itself describe the video returned to the user.

A.7 GEOMETRY BASELINES: COST AND COVERAGE

We compare geometry baselines at 2 and 24 fps to test whether denser input improves alignment decisions and to measure the additional computation. We measure runtime per clip on a single H100 over the $1{,}064$-clip evaluation set.

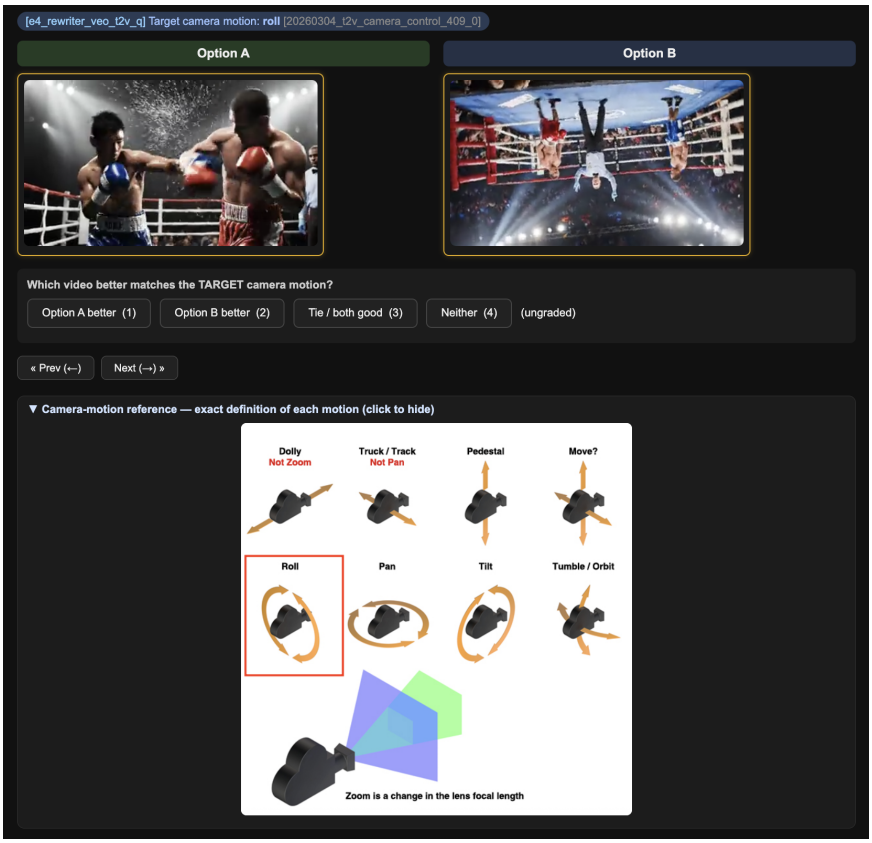


Figure 5: **Human grading interface.** The A/B comparison view presents two clips for the requested camera motion, here a roll. Graders choose Option A, Option B, both good, or neither. The motion-reference panel remains visible below the clips.

CUT3R takes 0.7 s per clip at 2 fps and 6.7 s at 24 fps. Its recurrent processing visits every frame, so increasing frame density raises runtime by roughly ten times. MegaSAM takes 75 s and 155 s, respectively. Its runtime is dominated by monocular depth estimation and global bundle adjustment, so the twelvefold increase in frame count approximately doubles its cost. Over the full evaluation set, these measurements correspond to 0.24 and 2.3 GPU-hours for CUT3R, and 26 and 53 GPU-hours for MegaSAM.

As reported in Section 4.2, denser input improves MegaSAM's balanced accuracy from 63.82% to 66.47%, while CUT3R shows no accuracy gain. Thus, the value of additional frames depends on the reconstruction method: MegaSAM improves at a substantial computational cost, whereas the added cost brings no measured benefit for CUT3R.

We also account for reconstruction failures when evaluating COLMAP. Under the acc-all convention, a failed reconstruction counts as a false negative for an aligned clip and a false positive for a misaligned clip. This measures performance over all requested ratings. Restricting evaluation to converged clips would give a higher accuracy but would omit the clips for which the RM cannot provide a decision.

A.8 Ablation details

We use the open-model ablation to test whether the cue design also helps Qwen3.5-9B and to distinguish the supervised cue comparison from the reinforcement-learning results. The four supervised configurations in Table 7 share the recipe in Appendix A.2 and differ only in their cue inputs.

The full text cue set reaches 74.91% bAcc after SFT, compared with 69.10% for flow text, 72.51% for global motion, and 70.87% for flow arrow maps. The combined cues perform best in this

Table 7: Rating-model ablation for Qwen3.5-9B on the frozen 1,064-clip set. Rates are in %; Δ is the bAcc gain in % over the untuned base. The SFT rows share the training recipe and vary the cue input. The final SAPO model uses the full motion cues; the GRPO control starts from the base model without cues. Bold and underline mark the best and second-best tuned results.

	Components				5 datasets weighted (%)				
Variant	Flow	Geom	SFT	RL	Acc	Spec	Rec	bAcc	Δ
Base Qwen3.5-9B (no tuning)†					61.89	19.45	92.88	56.16	—
Supervised fine-tuning; same recipe, cue is the only variable									
+ flow, text	✓		✓		70.68	**74.59**	63.61	69.10	+12.9
+ global motion		✓	✓		75.85	68.49	76.52	72.51	+16.3
+ flow, arrow maps	✓		✓		72.56	<u>73.40</u>	68.33	70.87	+14.7
+ flow + parallax + GM	✓	✓	✓		<u>77.44</u>	71.28	<u>78.53</u>	<u>74.91</u>	+18.7
Reinforcement learning									
GRPO from base				✓	74.41	71.56	74.02	72.79	+16.6
CamRM-Q (SAPO, warm-started)	✓	✓	✓	✓	**77.63**	72.46	**78.99**	**75.72**	**+19.6**

† The untuned base is the Δ reference and is excluded from the column ranking: it answers "matches" to almost everything, so its recall is high for a degenerate reason.

matched comparison. Flow arrow maps outperform flow text alone, so these results do not establish that text is always the better representation.

The GRPO control starts from the untuned base model without motion cues. The final CamRM-Q starts from a binary-SFT checkpoint and uses SAPO with optical-flow and global-motion cues, reaching 75.72% bAcc. Because these configurations differ in their starting checkpoint, cues, and RL method, their comparison evaluates the complete training configurations. We report the cue-SFT models and the binary-SFT classification variant separately in Table 6.

A.9 Classification protocol

We evaluate motion classification to test whether binary alignment training improves recognition of the motion itself. Models choose among 20 categories without additional classification training, except for the explicitly labelled classification-tuned comparisons in Table 8.

We use multi-label scoring for CameraBench because 30.9% of its positive clips contain two or more annotated motions. A top-1 prediction is correct when it names any annotated motion. Scoring the long-format rows independently would cap CameraBench top-1 accuracy at $207/279 = 74.2\%$ and understate performance for every model. The four generated splits contain one row per video and one motion label per row. Their labels do not fully describe compound motions such as orbits, 360 spins, and dolly zooms, which account for 6 to 25% of rows. We score the recorded label without granting additional entailment credit.

Binary alignment training improves bAcc@1 from 38.54% to 46.54% for Gemini and from 17.72% to 30.71% for Qwen3.5-9B. The binary-SFT Qwen model also exceeds the three camera-motion specialists, whose bAcc@1 scores are 29.23%, 22.42%, and 15.38%. These results support transfer from alignment training to motion recognition in both model families.

The three specialist models emit one category rather than a ranked list, so their top-5 accuracy equals their top-1 accuracy by construction; we omit their @5 entries. Direct classification training reaches 54.25% bAcc@1, indicating further room to improve recognition. However, 25.6% of the CameraBench classification training examples have contradictory single-label targets. This issue affects the classification-tuned comparison, while the binary-training transfer results do not use those targets.

Table 8: Camera-motion classification over 20 categories. We report five-dataset averages (%); chance bAcc@1 is 5%. Binary alignment training improves bAcc@1 by 8.0% for CamRM-G and 13.0% for Qwen3.5-9B. Top-1 predictions count as correct when they match any annotated motion in the multi-label evaluation. ‡ These runs emit a single category rather than a ranked list, so their Acc@5 equals their Acc@1 by construction; we do not report an @5 number for them.

Model	Acc@1	bACC@1	Macro-F1	Acc@5	bACC@5
Closed frontier models					
Gemini 3.5 Flash (vanilla)	41.70	38.54	35.27	77.43	74.85
Gemini 3.1 Pro (vanilla)	41.35	39.38	36.10	78.61	76.83
+ v2 prompt + CoT + optical flow	45.54	42.06	39.51	82.31	80.02
CamRM-G (tuned for *binary* only)	**50.52**	**46.54**	43.05	82.51	79.59
Open weights					
Qwen3.5-9B (untuned base)	18.40	17.72	13.55	56.90	55.85
Qwen3.6-27B (untuned base)	32.99	30.14	25.52	69.62	66.19
Qwen3.5-9B (binary SFT)	**33.11**	**30.71**	24.41	66.87	63.75
Camera-motion specialists					
ACaM (Qwen3-VL-8B)	30.34	29.23	24.76	–‡	–‡
CameraBench-Qwen (2.5-VL-7B)	24.54	22.42	21.68	–‡	–‡
CamReasoner-7B	16.44	15.38	10.19	–‡	–‡
Models trained for classification					
Best classification-tuned, top-1 decoding	**57.29**	**54.25**	**51.45**	–‡	–‡
Best classification-tuned, top-5 decoding	50.92	47.36	45.06	**87.74**	**86.66**

A.10 Specificity and recall

We examine specificity and recall to distinguish two errors that affect the refinement loop: accepting a mismatched clip and rejecting an aligned one. Figure 6 complements the aggregate benchmark metrics by showing both rates for each RM.

Untuned open models and several proprietary models have high recall but low specificity, indicating that they accept many mismatched clips. Geometry methods and the camera-motion specialists show the opposite pattern, rejecting many aligned clips. The three CamRM variants provide a more balanced combination of specificity and recall. This balance matters in the loop because false acceptance can end generation too early, while false rejection can discard a successful candidate.

A.11 Frame-rate ablation

We vary the input frame rate to determine how much temporal information the VLM needs for alignment rating. Table 9 evaluates the untuned Gemini 3.5 Flash at 1, 2, 4, and 8 fps with the remaining settings fixed.

Balanced accuracy is highest at 2 fps (67.72%), compared with 62.28% at 1 fps, 64.91% at 4 fps, and 63.51% at 8 fps. Specificity also peaks at 2 fps in this comparison. We therefore use 2 fps for the VLM rating input. Sparse sampling can omit temporal evidence, while denser input increases the visual context without improving accuracy here. We evaluate geometry baselines separately at 2 and 24 fps in Table 1.

Table 9: Frame-rate sweep on the untuned Gemini 3.5 Flash, 5-dataset size-weighted.

Sampling rate	bAcc	Spec
FPS = 1	62.28	60.67
FPS = 2 (Ours)	**67.72**	**61.33**
FPS = 4	64.91	59.67
FPS = 8	63.51	59.33

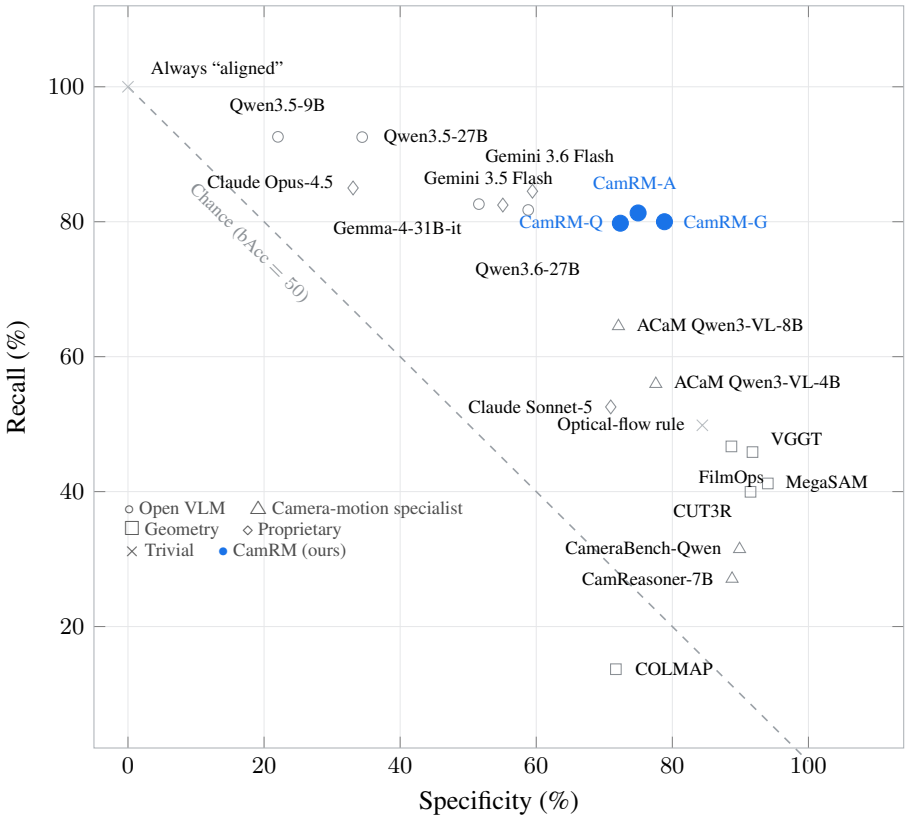


Figure 6: **Specificity and recall of the rating models.** Higher specificity means fewer mismatched clips are accepted, and higher recall means fewer aligned clips are rejected. The dashed line marks chance balanced accuracy. Untuned open models and several proprietary models favour acceptance, while geometry methods and camera-motion specialists favour rejection. The three CamRM variants, including the untuned CamRM-A, provide a more balanced combination.

A.12 The agentic variant: tool ablation

We develop CamRM-A to test whether explicit motion evidence improves rating without parameter updates. An untuned Gemini 3.5 Flash can request the cues in Section 3.2 through tools before returning a verdict. We provide four tools, listed below. Bracketed arguments are optional. The server binds the video path for each rollout, so the model does not receive the path. We cache each call within the rollout and reuse its result when the model repeats the request.

- `compute_optical_flow(mode={text,visual}, [start_time], [end_time])` returns a 3×3 grid of flow magnitudes and directions and the background-versus-foreground parallax statistics. The model can restrict the calculation to a time window. The `visual` mode also returns arrow-map images. The prompt instructs the model to examine the outer cells for camera motion and use higher parallax for truck, pedestal, and dolly and lower parallax for pan, tilt, and zoom.
- `estimate_camera_motion([start_time], [end_time])` returns net global translation, scale change, and in-plane rotation over the clip or a selected window, together with a ranking of the dominant component.
- `sample_video_frames(start_time, end_time, [fps])` returns a denser sequence of frames over a short window. It allows the model to inspect fast or subtle motion missed at the base rate of 2 fps. These calls use a per-rollout image budget.
- `detect_scene_change()` runs the automated shot-cut detector. A confirmed hard cut to unrelated content requires a negative verdict, following the human rubric.

Table 10 reports two comparisons: removing individual inputs and incrementally adding cues. Removing frames gives the lowest bAcc among the three removal ablations (67.19%), followed by removing the motion estimate (67.77%) and optical flow (69.43%). The incremental comparison starts with the motion estimate and adds flow and an SfM trajectory. We treat these as separate comparisons because they use different starting configurations.

The all-tools configuration reaches 78.13% bAcc in the tool ablation. The benchmark evaluation in Table 1 reports 77.97% for CamRM-A; these values come from separate evaluation records. The removal ablations show that frames remain useful alongside the explicit cues. Each agentic verdict also requires a multi-turn tool-calling episode, whereas the deployed loop uses the single-call CamRM-G.

Table 10: Tool ablation for CamRM-A, 5-dataset size-weighted.

Configuration	Acc	bAcc
No frames (cues only)	65.56	67.19
No motion estimate	65.50	67.77
No flow cue	67.04	69.43
Motion estimate only	69.18	70.85
Flow + motion	69.09	70.79
+ SfM trajectory	70.42	71.19
CamRM-A (all tools)	**80.13**	**78.13**

A.12.1 Agentic reasoning examples

We inspect individual rollouts to understand how CamRM-A uses the tools when making a decision. Figures 7 to 9 show three examples from the run used for Table 10. Each figure includes six sampled frames, the instruction, the model's reasoning and tool response at each turn, and the final verdict beside the human label. Recomputing performance from the rollout dump reproduces the table within 0.1%. We shorten long tool responses and reasoning passages, mark every omission, and retain the measurements cited by the model beside its reasoning.

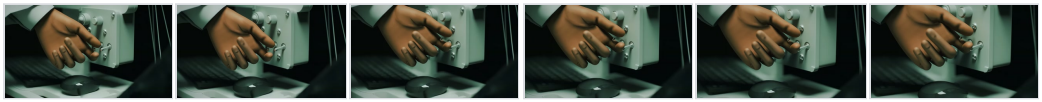
We select examples with the most common call sequence. Among 493 CameraBench items, 385 use one call to each of three tools over four turns, and 467 call all three tools. The mean is 2.98 calls per item. The source notes record the less common sequences that are not shown. The three examples use the parallax summary in different ways.

In Figure 7, the estimated scale increases by 26.3% and the flow diverges radially. These cues are compatible with either zoom or dolly. The outer cells average 1.17 px and the center 1.14 px, so the model follows the cue's rule for similar regional magnitudes and predicts zoom. In Figure 8, all cells move left, the global shift is 54.0% of frame width, and the outer flow is approximately uniform. The model interprets this as a pan and accepts the clip.

In Figure 9, the outer cells average 1.19 px while the center averages 3.92 px. The model attributes the stronger central motion to the speaker and rejects the requested pedestal-down motion. This example shows how regional flow can help separate subject motion from camera motion, as in the CamRM-G example in Figure 13.

The tool responses also influence how the model gathers evidence. The global-motion response warns that it cannot distinguish pan from truck or zoom from dolly and instructs the model to check parallax. A subsequent flow call therefore follows guidance already present in the tool output. The traces show evidence use, but they do not isolate independent planning ability. The three tools called in these examples provide the same types of evidence available to CamRM-G in one call. Since the two models differ by less than 1.3% in bAcc, we use CamRM-G in the deployed loop to avoid a multi-turn episode for every rating.

CameraBench · real video | instruction to the rater: "**The camera zooms in.**" | CamRM-A · Gemini 3.5 Flash, untuned



TURN 0 — REASONING

My Thought Process for Evaluating Camera Motion Against Scene Changes

My primary objective here is to rigorously determine whether the observed camera movement, specifically a zoom-in action (categorized as 'zoom_in'), aligns with the

[... 2028 further characters of reasoning elided for space ...]

→ **detect_scene_change()**

```
The 'simple' scene-change detector found no shot cuts / scene
changes. Treat the clip as a single continuous shot unless you
clearly see otherwise.
backend: simple
scene_cuts: []
```

TURN 1 — REASONING

My Thought Process for Handling Camera Motion and Optical Flow

Alright, so the immediate priority is clear: we're staying within the current scene, no transitions needed. My next logical step is to dive into the core processing tasks. This means I need to invoke the `estimate_camera_motion` function. This is where I'll be crunching the data to figure out how the camera has moved between frames – translation, rotation, all of it. Following that, and intrinsically linked to understanding that motion, I'll be calling `compute_optical_flow`.

[... 455 further characters of reasoning elided for space ...]

→ **estimate_camera_motion()**

```
Global camera-motion estimate over 0.0s-3.8s (15 intervals):
- Horizontal: camera pans/trucks RIGHT (content shifted left 11.6%
of width).
- Vertical: camera tilts/pedestals DOWN (content shifted up 11.4% of
height).
- Depth/lens: camera ZOOMS IN or DOLLIES IN (content scaled up
26.2%).
- Roll: no significant roll.
- Dominant motion component(s): scale (zoom/dolly), translation,
rotation (roll).
[... fixed usage instructions elided ...]
net_horizontal_frac: -0.1159
net_vertical_frac: -0.1138
net_scale_delta: 0.2619
net_rotation_deg: 0.25
intervals_used: 15
```

TURN 2 — REASONING

My Thought Process on Disentangling Camera Motion

Right, my primary objective here is to definitively determine the nature of the camera's movement within this sequence. I need to differentiate between a straightforward zoom-in and a dolly-in, and also rule out simpler motions like a static shot or a basic pan. To achieve this, my immediate focus is on analyzing the output of the `compute_optical_flow` function.

Specifically, I'm looking for the parallax statistics it provides. These statistics are crucial because they offer distinct signatures for zoom versus dolly movements. A zoom, for instance, would exhibit a different parallax pattern compared to the uniform inward motion characteristic of a dolly.

[... 602 further characters of reasoning elided for space ...]

→ **compute_optical_flow()**

```
[Optical Flow Analysis]: automated 3x3 grid motion summary +
parallax stats, sampled at 2.0 FPS. Directions describe how image
CONTENT moves; camera motion is the opposite (content moving LEFT =>
camera pans/trucks RIGHT).
--- Time interval: 0.0s to 0.5s ---
  Top-Left: Magnitude 1.7 px, Dominant Direction LEFT
  Top-Center: Magnitude 0.5 px, Dominant Direction NONE
  Top-Right: Magnitude 0.8 px, Dominant Direction NONE
  Center-Left: Magnitude 0.4 px, Dominant Direction NONE
  Center: Magnitude 0.7 px, Dominant Direction NONE
  Center-Right: Magnitude 0.7 px, Dominant Direction NONE
  Bottom-Left: Magnitude 1.9 px, Dominant Direction DOWN-LEFT
  Bottom-Center: Magnitude 1.5 px, Dominant Direction DOWN
  Bottom-Right: Magnitude 0.2 px, Dominant Direction NONE
[... 7 further 1 s intervals elided for space ...]
[Parallax summary over window] background(outer) mean magnitude =
1.17 px; center(foreground) mean magnitude = 1.14 px; horizontal
divergence = +1.85; vertical divergence = +1.18; mean background
translation = (-0.17, -0.09).
[Interpretation] Outer flow diverges radially
(expanding/contracting) rather than translating uniformly ->
consistent with ZOOM or DOLLY. If the background magnitude is much
smaller than foreground/edge magnitude, favor DOLLY (depth
parallax); if uniform across depth, favor ZOOM.
mode: text
```

FINAL ANSWER

<think>
The user's prompt is "The camera zooms in.", which corresponds to the category `zoom_in`.

First, I ran `detect_scene_change` which returned no shot cuts.
Second, I analyzed the global motion estimate via `estimate_camera_motion` which estimated:

[... 667 further characters of answer elided for space ...]

Therefore, the video matches the requested camera motion.
</think><answer>1</answer>

CamRM-A verdict: **aligned** human label: **aligned** **match**

Figure 7: **Agentic rating of a zoom.** We show the inputs, tool calls, responses, and verdict for one CamRM-A rollout. The model calls each of three tools once over four turns. Scale increases by 26.3%, while outer and centre flow magnitudes are 1.17 and 1.14 px. The model uses their similarity to interpret the radial flow as zoom under the cue's parallax rule.

CameraBench · real video | instruction to the rater: "**The camera pans right.**" | CamRM-A · Gemini 3.5 Flash, untuned



TURN 0 — REASONING

My Thought Process on Evaluating Video Against Camera Motion

Alright, so the core task here is to rigorously verify if a given video segment accurately reflects a specific, stated camera movement.

[... 1245 further characters of reasoning elided for space ...]

→ `detect_scene_change()`

```
The 'simple' scene-change detector found no shot cuts / scene
changes. Treat the clip as a single continuous shot unless you
clearly see otherwise.
backend: simple
scene_cuts: []
```

TURN 1 — REASONING

My Thought Process on Camera Motion Estimation

Alright, so the first thing I'm noticing is the complete lack of scene changes. This is crucial information because it means whatever motion we're tracking is happening *within* a single, static environment. There's no jumping from one shot to another, which simplifies things considerably.

Given this constant environment, my immediate instinct is to move towards analyzing the actual movement itself. The next logical step for me is to decide which tool is best suited for this.

[... 1556 further characters of reasoning elided for space ...]

→ `estimate_camera_motion()`

```
Global camera-motion estimate over 0.0s-4.4s (18 intervals):
- Horizontal: camera pans/trucks RIGHT (content shifted left 54.0%
of width).
- Vertical: camera tilts/pedestals UP (content shifted down 17.4% of
height).
- Depth/lens: camera ZOOMS IN or DOLLIES IN (content scaled up
9.5%).
- Roll: no significant roll.
- Dominant motion component(s): translation, scale (zoom/dolly),
rotation (roll).
[... fixed usage instructions elided ...]
net_horizontal_frac: -0.5404
net_vertical_frac: 0.174
net_scale_delta: 0.0948
net_rotation_deg: 1.09
intervals_used: 18
```

TURN 2 — REASONING

My Thought Process on Camera Motion Analysis

Alright, so the global camera-motion estimate is giving me some interesting feedback here. It's reporting a "Horizontal: camera pans/trucks RIGHT (content shifted left 54.0% of width)," and identifying "Dominant motion component(s): translation, scale (zoom/dolly), rotation (roll)."

My immediate thought is to clarify the terminology: is it a "pan_right" or a "truck_right," or is it genuinely a pan to the right? I need to ground this abstract motion in the visual reality of the frames.

Let me dive into the frames, starting from 00:00.000. I see a man in a distinctive green and yellow shirt with "ZINK" visible on the back, and then I catch "Troy Lee" on his sleeve.

[... 1888 further characters of reasoning elided for space ...]

→ `compute_optical_flow()`

```
[Optical Flow Analysis]: automated 3x3 grid motion summary +
parallax stats, sampled at 2.0 FPS. Directions describe how image
CONTENT moves; camera motion is the opposite (content moving LEFT =>
camera pans/trucks RIGHT).
--- Time interval: 0.0s to 0.5s ---
  Top-Left: Magnitude 2.4 px, Dominant Direction DOWN-LEFT
  Top-Center: Magnitude 1.8 px, Dominant Direction DOWN
  Top-Right: Magnitude 9.9 px, Dominant Direction DOWN-LEFT
  Center-Left: Magnitude 9.9 px, Dominant Direction DOWN-LEFT
  Center: Magnitude 4.7 px, Dominant Direction DOWN-LEFT
  Center-Right: Magnitude 10.5 px, Dominant Direction LEFT
  Bottom-Left: Magnitude 7.6 px, Dominant Direction DOWN-LEFT
  Bottom-Center: Magnitude 3.5 px, Dominant Direction LEFT
  Bottom-Right: Magnitude 9.0 px, Dominant Direction LEFT
[... 8 further 1 s intervals elided for space ...]
[Parallax summary over window] background(outer) mean magnitude =
5.15 px; center(foreground) mean magnitude = 3.89 px; horizontal
divergence = +2.98; vertical divergence = -0.03; mean background
translation = (-3.95, +1.45).
[Interpretation] Outer flow is a fairly uniform translation ->
consistent with PAN/TILT (pure rotation, low parallax) or
TRUCK/PEDESTAL (translation with parallax; check whether near
objects move faster than far ones).
mode: text
```

FINAL ANSWER

An analysis of the video's camera motion reveals the following:

1. **Camera Motion Identification**: The prompt specifies "The camera pans right" which corresponds to the **pan_right** category (camera rotates horizontally to the right, causing the visual con

[... 449 further characters of answer elided for space ...]

3. **Verdict**: The camera motion is dominated by a steady rightward pan following the subjects in the landscape.

<answer>1</answer>

CamRM-A verdict: **aligned** human label: **aligned** **match**

Figure 8: **Agentic rating of a pan.** Content moves left in every grid cell, with a global shift of 54.0% of frame width. The outer flow is a uniform translation. The model cites these measurements in the order returned by the tools and accepts the requested pan.

CameraBench · real video instruction to the rater: "**The camera pedestals down.**" CamRM-A · Gemini 3.5 Flash, untuned



TURN 0 — REASONING

My Thought Process on Evaluating Camera Pedestals

Okay, so the core task here is to determine if the camera is indeed "pedestaling down" in the provided video.

[... 452 further characters of reasoning elided for space ...]

→ `detect_scene_change()`

```
The 'simple' scene-change detector found no shot cuts / scene
changes. Treat the clip as a single continuous shot unless you
clearly see otherwise.
backend: simple
scene_cuts: []
```

TURN 1 — REASONING

Initial Assessment of Video Stasis and Planned Motion Analysis

My immediate observation is a complete lack of scene changes. This suggests a potentially static shot, which, given my expertise, immediately leads me to consider the possibility of no significant camera motion. To confirm this hypothesis, my next step is to rigorously compute optical flow and estimate camera motion.

[... 746 further characters of reasoning elided for space ...]

→ `estimate_camera_motion()`

```
Global camera-motion estimate over 0.0s-2.2s (9 intervals):
- Horizontal: camera pans/trucks LEFT (content shifted right 8.2% of
width).
- Vertical: no significant vertical camera motion.
- Depth/lens: no significant scale change.
- Roll: camera ROLL detected (~5.1 deg; horizon tilts).
- Dominant motion component(s): translation, rotation (roll), scale
(zoom/dolly).
[... fixed usage instructions elided ...]
net_horizontal_frac: 0.0816
net_vertical_frac: -0.0352
net_scale_delta: -0.0187
net_rotation_deg: 5.08
intervals_used: 9
```

TURN 2 — REASONING

My Internal Monologue on Optical Flow Analysis

Alright, so I'm looking at this `net_vertical_frac` value, and it's coming in at -0.0352. My initial reaction is, "That's incredibly small." But then I'm registering the negative sign. My first thought is about the convention: does a negative vertical fraction mean the content is shifting *up* or *down* relative to the camera's perspective? Usually, I associate a positive fraction with the camera moving upwards (and thus content appearing to move downwards), but the absolute magnitude is so negligible that it immediately leads me to the overarching conclusion the system has already drawn: "no significant vertical camera motion." This makes sense; such a tiny deviation is likely noise or an artifact, not a meaningful camera movement.

[... 689 further characters of reasoning elided for space ...]

→ `compute_optical_flow()`

```
[Optical Flow Analysis]: automated 3x3 grid motion summary +
parallax stats, sampled at 2.0 FPS. Directions describe how image
CONTENT moves; camera motion is the opposite (content moving LEFT =>
camera pans/trucks RIGHT).
--- Time interval: 0.0s to 0.5s ---
  Top-Left: Magnitude 1.9 px, Dominant Direction RIGHT
  Top-Center: Magnitude 4.8 px, Dominant Direction UP-RIGHT
  Top-Right: Magnitude 0.3 px, Dominant Direction NONE
  Center-Left: Magnitude 1.4 px, Dominant Direction RIGHT
  Center: Magnitude 3.4 px, Dominant Direction UP-RIGHT
  Center-Right: Magnitude 0.2 px, Dominant Direction NONE
  Bottom-Left: Magnitude 0.9 px, Dominant Direction NONE
  Bottom-Center: Magnitude 1.5 px, Dominant Direction RIGHT
  Bottom-Right: Magnitude 0.8 px, Dominant Direction NONE
[... 3 further 1 s intervals elided for space ...]
[Parallax summary over window] background(outer) mean magnitude =
1.19 px; center(foreground) mean magnitude = 3.92 px; horizontal
divergence = -0.49; vertical divergence = +0.05; mean background
translation = (+0.63, +0.30).
[Interpretation] Outer flow is a fairly uniform translation ->
consistent with PAN/TILT (pure rotation, low parallax) or
TRUCK/PEDESTAL (translation with parallax; check whether near
objects move faster than far ones).
mode: text
```

FINAL ANSWER

<think>
The user prompt asks if the camera pedestals down in the given video.
We ran the scene change detector and found no scene cuts.
We ran the camera motion estimation and optical flow tools.
The camera motion estimation shows no significant vertical camer
[... 415 further characters of answer elided for space ...]
Thus, the expected motion "pedestal down" is absent.
The final alignment rating must be 0.
</think>
<answer>0</answer>

CamRM-A verdict: **misaligned** human label: **misaligned** **match**

Figure 9: **Separating subject and camera motion.** Outer cells average 1.19 px, compared with 3.92 px in the centre. The model attributes the central motion to the speaker, reports no vertical camera movement, and rejects the pedestal-down request, agreeing with the human label. The grid preserves regional differences that a single global estimate cannot show.

A.13 PER-CATEGORY BREAKDOWN

We examine category-level accuracy to identify camera motions that remain difficult for CamRM-G. Table 11 reports accuracy for each source label, averaged over the datasets in which that label appears.

Pan-left, zoom, and static exceed 84% accuracy. Orbit-under reaches 45.0% and orbit-right reaches 66.4%, while the two roll directions also differ substantially. These results suggest that motions requiring interpretation of a 3D camera path remain difficult.

We use an unweighted mean because the recorded category rates do not include per-category clip counts. The column n gives the number of datasets contributing to each mean, and we exclude datasets in which a label is absent. The source vocabularies differ across tasks, so we retain the labels listed in Appendix A.1. The untuned baseline has aggregate results only; this table therefore describes CamRM-G's category-level performance without a category-level baseline comparison.

Table 11: Per-category accuracy of CamRM-G (%), averaged *unweighted* over the n datasets in which the category appears.

Category	n	Acc	**Category**	n	Acc
pan_left	5	89.5	orbit_over	2	76.7
dolly_zoom	1	87.5	truck_left	5	76.2
zoom_out	5	86.6	truck_right	4	75.9
zoom_in	4	85.6	roll	2	75.0
static	5	84.7	roll_clockwise	1	75.0
tilt_up	5	84.3	orbit_left	4	70.8
dolly_out	4	83.3	360_spin	2	70.0
dolly_in	4	82.6	tilt_down	5	68.1
pedestal_up	3	82.2	orbit_right	4	66.4
pedestal_down	5	81.5	roll_counterclockwise	1	63.6
pan_right	5	80.9	orbit_under	2	45.0
move_up	1	80.0			

A.14 QUALITATIVE EXAMPLES OF LOOP BEHAVIOUR

We inspect successful repairs and regressions to understand how RM feedback changes the generated video. Of the 336 deployed-loop items, 145 have human grades for both the dataset originals and the returned clips. The loop repairs 82, leaves 54 misaligned, and worsens 9.

We select repairs in which the RM rejected the initial loop candidate for motion misalignment. We also check whether that candidate is the graded dataset original, so the displayed comparison can be interpreted correctly.

Figures 10 and 11 show seven repairs: three from Veo 3.1 and four from Omni, covering both generation tasks and seven target categories. At least two of the three original graders judged each dataset original misaligned, and at least four of the five final graders judged the returned clip aligned, above the $2/5$ threshold used to group cases. We select examples whose motion can be interpreted from six still frames.

The Veo loops start from the graded dataset originals. For Omni, the top row shows the graded dataset original, while the loop starts from a different sample of the same prompt that the RM also judges misaligned.

The examples show several types of motion error. The dolly_out original moves inward; the tilt_down original holds the horizon while the mist moves; and the orbit_right original stays nearly static. In the pan_right original, the camera stays fixed while the runners leave the frame. The orbit_left original moves towards the fresco without circling it, the tilt_up original moves across the fire escape without turning towards the sky, and the roll original has no frame rotation. Four categories, orbit_right, tilt_down, orbit_left, and roll, are in the lowest third of the per-category results in Table 11. The examples show that the RM can identify

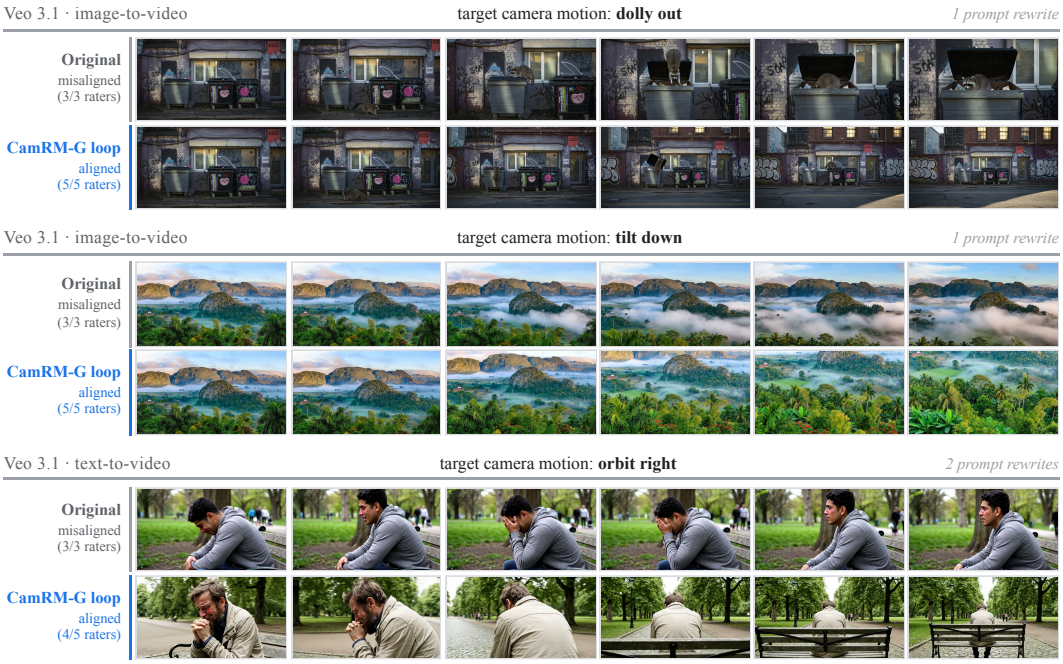


Figure 10: **Successful repairs on Veo 3.1.** We show six uniformly sampled frames from each original clip and the clip returned by the CamRM-G loop. All three original graders judged each starting clip misaligned.

errors even in these difficult categories, consistent with its use of scene content alongside the flow signature.

In each repair, the rewriter expands the short motion request into explicit instructions about camera movement and what should remain fixed. For `tilt_up`, it asks for a stationary pivot with the view rotating upward. For `pan_right`, it fixes the camera position and describes the runners moving towards the left edge as the camera pans. These examples illustrate how the RM's diagnosis can guide a more specific prompt without changing the generator's weights.

Figure 12 shows both regressions that remain after the dataset-original check. A static clip acquires a slow drift, and an orbit-left clip that circles a hologram becomes nearly static. Both change from $3/3$ original graders judging them aligned to $0/5$ final graders. The failure mechanisms differ. For orbit-left, the RM accepts the rewritten clip despite all five final graders rejecting it. For static, the RM rejects all three rewrites, but the loop returns the last candidate when the budget is exhausted.

The static example suggests a possible change to the stopping rule: retaining the original when no candidate is accepted would avoid this regression. It would not address the false acceptance in the orbit-left example. These cases therefore point to separate limitations in rating and fallback selection. The 54 unrepaired items are more common than the 9 regressions in the graded subset. We illustrate regressions because missing motion is difficult to show in still frames, so the figure should not be interpreted as a comparison of failure frequencies.

A.15 Rating-model reasoning examples

We inspect CamRM-G's reasoning to understand how it relates the motion cues to individual alignment decisions. Figures 13 to 15 show the sampled frames, the user's prompt, the computed cues, the model's reasoning, and its verdict beside the human label. We place the cue measurements next to the reasoning so that readers can check the evidence used for each decision.

We replay cue text from the evaluation run's optical-flow cache, exactly matching the text received by the model. The prompts, reasoning, and verdicts are verbatim, including the third example's error in assigning the same bib number to both runners. We shorten only the cue panels by omitting

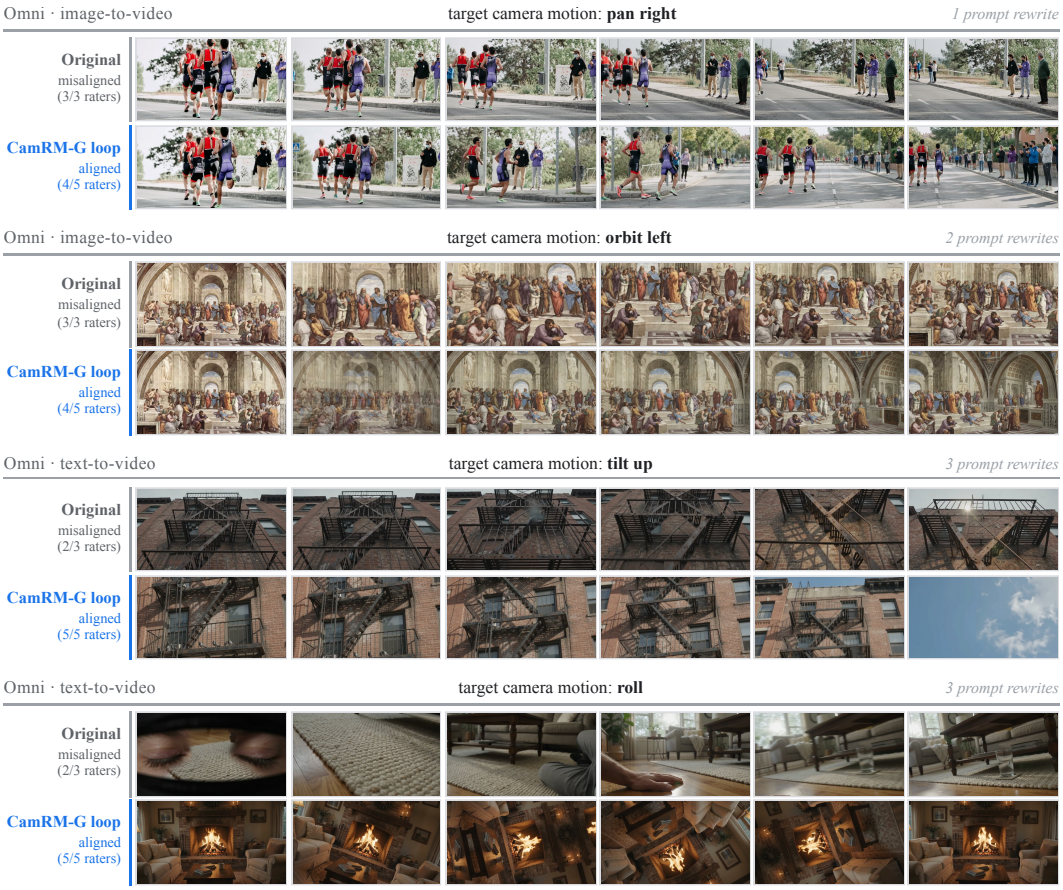


Figure 11: **Successful repairs on Omni.** Each loop started from a new sample of the same prompt. The top rows show the graded dataset originals; the loop's starting clips, which the RM also judged misaligned, are not shown. Text-to-video examples have no shared conditioning frame, so their scene content also differs.

repeated per-second flow grids and shared usage instructions, with every omission marked. The traces come from a fresh inference pass because the original evaluation stored only verdicts. We use the same model and configuration as in Table 1, but the model is not deterministic at temperature 0, so the decisions do not match item for item. We report no accuracy from this pass.

We select the examples to examine different uses of the cues. In Figure 13, the eight outer cells move right at roughly 4–10 px while the center moves by about 1.8 px. The model interprets this as the camera trucking left while keeping the jogger centerd. The grid makes the difference between the subject and surrounding motion explicit.

In Figure 15, the runners become larger in successive frames, which can resemble the requested `zoom_in`. The global-motion estimate reports no significant scale change. The model uses this cue to infer that the runners approach a camera that pans and tilts to follow them, and rejects the clip. The human graders agree. This example illustrates how the global cue can help distinguish subject approach from lens magnification, which is difficult to judge from subject size alone.

Figure 14 shows a correct verdict with a weakness in its explanation. The request is `dolly_out`, and the model cites the estimated 69.7% reduction in content scale. This supports an apparent retreat, but scale change alone cannot distinguish camera translation from zoom. The model then appeals to the parallax summary. Its rule associates dolly motion with a lower background magnitude than foreground magnitude, yet the outer cells measure 5.54 px and the center 2.11 px, the opposite ordering.

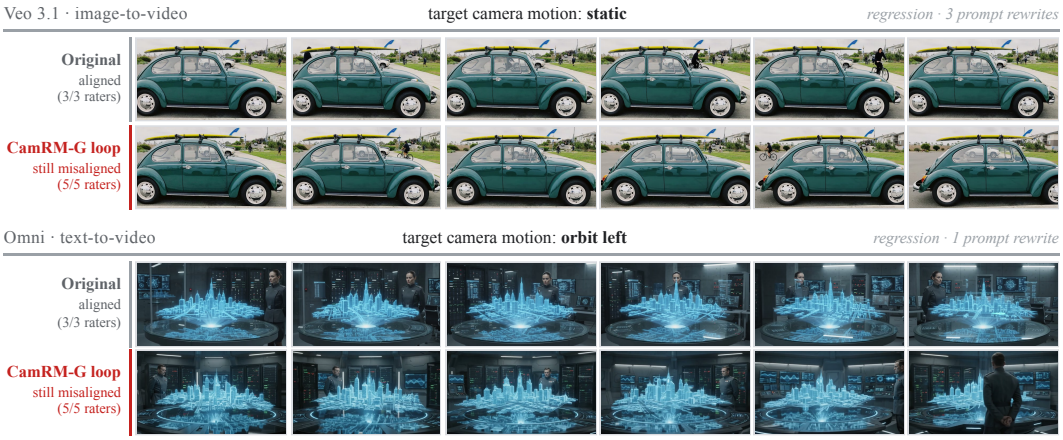
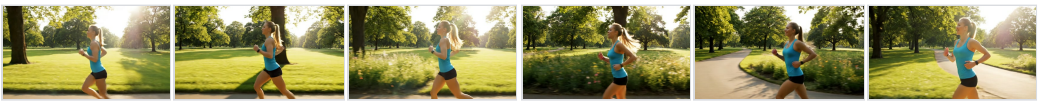


Figure 12: **Regressions after feedback.** We show both regressions among cases where the loop started from the graded dataset original. All three original graders judged the starting clips aligned, while all five final graders judged the returned clips misaligned.

This mismatch exposes a limitation of the cue description. The outer cells are labelled as background, but spatial position is only a proxy for depth. A radial flow field around a centerd foreground subject can violate that assumption. Although the verdict agrees with the human label, its explanation is not fully supported by the parallax values it cites. The example motivates clearer wording about the relation between spatial regions and scene depth when presenting these cues to the RM.

Veo 3.1 · text-to-video | target camera motion: **truck left** | CamRM-G · FT step 395



USER INPUT — GENERATION PROMPT

Side profile view of a woman jogger running on a park path, camera trucks left at the same speed keeping her centered

MOTION CUES COMPUTED FROM PIXELS

```
[Optical Flow Textual Analysis]: Below is an automated 3x3
grid motion summary sampled at 1.0 FPS.
--- Time interval: 0.0s to 1.0s ---
  Top-Left: Magnitude 5.4 px, Dominant Direction RIGHT
  Top-Center: Magnitude 3.9 px, Dominant Direction RIGHT
  Top-Right: Magnitude 8.1 px, Dominant Direction RIGHT
  Center-Left: Magnitude 10.2 px, Dominant Direction RIGHT
  Center: Magnitude 1.8 px, Dominant Direction RIGHT
  Center-Right: Magnitude 10.7 px, Dominant Direction RIGHT
  Bottom-Left: Magnitude 5.8 px, Dominant Direction RIGHT
  Bottom-Center: Magnitude 1.8 px, Dominant Direction UP-
LEFT
  Bottom-Right: Magnitude 6.9 px, Dominant Direction RIGHT
[... 15 further 1 s intervals elided for space ...]
[Parallax summary over window] background(outer) mean
magnitude = 4.87 px; center(foreground) mean magnitude =
3.77 px; horizontal divergence = -1.76; vertical divergence
= -1.04; mean background translation = (+3.94, +0.10).
[Interpretation] Outer flow is a fairly uniform translation
-> consistent with PAN/TILT (pure rotation, low parallax)
or TRUCK/PEDESTAL (translation with parallax; check whether
near objects move faster than far ones).

[Global Camera-Motion Estimate]
Global camera-motion estimate over 0.0s-8.0s (31
intervals):
- Horizontal: camera pans/trucks LEFT (content shifted
right 56.3% of width).
- Vertical: no significant vertical camera motion.
- Depth/lens: camera ZOOMS OUT or DOLLIES OUT (content
scaled down 13.1%).
- Roll: camera ROLL detected (~3.4 deg; horizon tilts).
- Dominant motion component(s): translation, scale
(zoom/dolly), rotation (roll).
[... fixed usage instructions elided ...]
```

CAMRM-G CHAIN OF THOUGHT

The user prompt asks for a video showing a "Side profile view of a woman jogger running on a park path, camera trucks left at the same speed keeping her centered".
This corresponds to the camera motion category 'truck_left'.

Let's analyze the video:
1. **Scene Continuity**: The video is a single continuous shot from 00:00.000 to 00:16.000 with no cuts or sudden jumps.
2. **Subject and Setting**: A woman is jogging on a paved path in a park with green grass and trees. She is shown in a side profile view.
3. **Camera Motion**: The woman is running towards the left of the frame. The camera moves along with her to the left (trucking left) at the same speed, keeping her centered in the frame.
4. **Optical Flow Analysis**: The optical flow shows that the background (outer grid cells) consistently moves to the right (dominant direction: RIGHT), which indicates that the camera is moving to the left (trucking left). The center grid cell (the woman) has a much lower motion magnitude, confirming she is kept stable and centered in the frame.

The video perfectly matches the user's request.

CamRM-G verdict: **aligned** human label: **aligned** **match**

Figure 13: **Rating a truck-left motion.** We show the inputs and response for one CamRM-G rating. The outer flow cells move right while the centre cell has much weaker motion. The model interprets this pattern as the camera trucking left while keeping the subject centred.

Veo 3.1 · image-to-video | target camera motion: **dolly out** | CamRM-G · FT step 395



USER INPUT — GENERATION PROMPT

The camera dollies out. A small bird flies with remarkable agility through the dense cacti.

MOTION CUES COMPUTED FROM PIXELS

```
[Optical Flow Textual Analysis]: Below is an automated 3x3
grid motion summary sampled at 1.0 FPS.
--- Time interval: 0.0s to 1.0s ---
  Top-Left: Magnitude 2.4 px, Dominant Direction DOWN-RIGHT
  Top-Center: Magnitude 2.4 px, Dominant Direction DOWN-
RIGHT
  Top-Right: Magnitude 2.3 px, Dominant Direction DOWN-
RIGHT
  Center-Left: Magnitude 2.4 px, Dominant Direction DOWN-
RIGHT
  Center: Magnitude 2.4 px, Dominant Direction DOWN-RIGHT
  Center-Right: Magnitude 2.3 px, Dominant Direction DOWN-
RIGHT
  Bottom-Left: Magnitude 2.4 px, Dominant Direction DOWN-
RIGHT
  Bottom-Center: Magnitude 2.4 px, Dominant Direction DOWN-
RIGHT
  Bottom-Right: Magnitude 2.4 px, Dominant Direction DOWN-
RIGHT
[... 15 further 1 s intervals elided for space ...]
[Parallax summary over window] background(outer) mean
magnitude = 5.54 px; center(foreground) mean magnitude =
2.11 px; horizontal divergence = -8.96; vertical divergence
= -5.91; mean background translation = (-0.36, +1.37).
[Interpretation] Outer flow diverges radially
(expanding/contracting) rather than translating uniformly -
> consistent with ZOOM or DOLLY. If the background
magnitude is much smaller than foreground/edge magnitude,
favor DOLLY (depth parallax); if uniform across depth,
favor ZOOM.

[Global Camera-Motion Estimate]
Global camera-motion estimate over 0.0s-8.0s (31
intervals):
- Horizontal: camera pans/trucks LEFT (content shifted
right 49.4% of width).
- Vertical: camera tilts/pedestals UP (content shifted down
71.5% of height).
- Depth/lens: camera ZOOMS OUT or DOLLIES OUT (content
scaled down 69.7%).
- Roll: no significant roll.
- Dominant motion component(s): translation, scale
(zoom/dolly), rotation (roll).
[... fixed usage instructions elided ...]
```

CAMRM-G CHAIN OF THOUGHT

The user prompt states: "The camera dollies out. A small bird flies with remarkable agility through the dense cacti."
This corresponds to the camera motion category 'dolly_out'.

Let's analyze the video:
1. **Scene Continuity**: The video is a single continuous shot of a hummingbird flying around a cactus bush in a desert environment. There are no scene cuts or jump cuts.
2. **Camera Motion**: The video starts with a close-up shot of the cactus. As the hummingbird flies around, the camera steadily moves backward (dollies out), revealing the entire cactus bush and the surrounding desert landscape.
3. **Optical Flow Analysis**: The optical flow summary indicates a strong radial divergence consistent with zooming out or dollying out (content scaled down by 69.7%). The presence of depth parallax (the foreground cactus shifts and scales differently compared to the distant desert background) confirms a physical dolly out rather than a simple lens zoom.

The camera motion is a smooth, continuous dolly out, which perfectly matches the user's prompt.

CamRM-G verdict: **aligned** human label: **aligned** **match**

Figure 14: **Distinguishing dolly from zoom.** The global-motion cue reports a 69.7% reduction in content scale, which alone cannot distinguish camera translation from zoom. The model then uses the parallax cue. Its verdict agrees with the human label, but the cited parallax values do not fully support its explanation, as discussed in Appendix A.15.

Omni · image-to-video | target camera motion: **zoom in** | CamRM-G · FT step 395



USER INPUT — GENERATION PROMPT

The camera zooms in. In the outdoor cross-country race, the runner in pink speeds up and overtakes the runner in front of him.

MOTION CUES COMPUTED FROM PIXELS

```
[Optical Flow Textual Analysis]: Below is an automated 3x3
grid motion summary sampled at 1.0 FPS.
--- Time interval: 0.0s to 1.0s ---
  Top-Left: Magnitude 4.8 px, Dominant Direction RIGHT
  Top-Center: Magnitude 4.7 px, Dominant Direction RIGHT
  Top-Right: Magnitude 4.9 px, Dominant Direction RIGHT
  Center-Left: Magnitude 4.9 px, Dominant Direction RIGHT
  Center: Magnitude 4.2 px, Dominant Direction UP-RIGHT
  Center-Right: Magnitude 5.4 px, Dominant Direction RIGHT
  Bottom-Left: Magnitude 4.9 px, Dominant Direction RIGHT
  Bottom-Center: Magnitude 5.1 px, Dominant Direction RIGHT
  Bottom-Right: Magnitude 4.7 px, Dominant Direction RIGHT
[... 19 further 1 s intervals elided for space ...]
[Parallax summary over window] background(outer) mean
magnitude = 5.44 px; center(foreground) mean magnitude =
2.93 px; horizontal divergence = +1.64; vertical divergence
= -0.23; mean background translation = (-1.05, -1.39).
[Interpretation] Outer flow is a fairly uniform translation
-> consistent with PAN/TILT (pure rotation, low parallax)
or TRUCK/PEDESTAL (translation with parallax; check whether
near objects move faster than far ones).

[Global Camera-Motion Estimate]
Global camera-motion estimate over 0.0s-10.0s (39
intervals):
- Horizontal: camera pans/trucks LEFT (content shifted
right 15.0% of width).
- Vertical: camera tilts/pedestals DOWN (content shifted up
30.2% of height).
- Depth/lens: no significant scale change.
- Roll: no significant roll.
- Dominant motion component(s): translation, scale
(zoom/dolly), rotation (roll).
[... fixed usage instructions elided ...]
```

CAMRM-G CHAIN OF THOUGHT

The user prompt states: "The camera zooms in. In the outdoor cross-country race, the runner in pink speeds up and overtakes the runner in front of him." This corresponds to the camera motion category 'zoom_in'.

Let's analyze the video:
1. **Runner Motion**: At the beginning of the video (00:00), the runner in blue (bib 168) is in front of the runner in pink (bib 168). As the video progresses, the runner in pink speeds up, runs alongside the blue runner, and eventually overtakes him, becoming the lead runner by around 00:12.
2. **Camera Motion**: The camera tracks the runners as they run towards it. The runners get larger in the frame because they are physically moving closer to the camera. However, the camera itself does not zoom in. The background elements (such as the trees and the path behind the runners) do not show any magnification or zoom-in effect. The optical flow analysis confirms this, stating "no significant scale change" for the depth/lens motion. The camera is primarily panning and tilting to keep the approaching runners in the center of the frame.

Since the camera does not zoom in, the video does not match the requested camera motion.

CamRM-G verdict: **misaligned** human label: **misaligned** **match**

Figure 15: **Distinguishing subject approach from zoom.** The runners become larger across the sampled frames. The model cites the global-motion estimate of no significant scale change, attributes the increase in subject size to their approach, and rejects the requested zoom.