# MOTIONGRAPHICSBENCH: EVALUATING MOTION GRAPHICS GENERATION WITH PROGRAMMATIC VERIFICATION

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

## ABSTRACT

Large language models can generate motion graphics from static images and natural language requests, but it remains difficult to evaluate how well models can follow requests that combine spatial, appearance, and temporal requirements. We introduce MotionGraphicsBench, an expert-curated benchmark of 100 instances for animating an input static SVG with a motion edit prompt. By supplying all required visual elements, our editing task reflects a designer-like workflow and isolates motion generation from graphic generation. Derived from real-world Lottie motion designs, our benchmark contains diverse combinations of spatio-temporal animation behaviors. Each benchmark instance pairs its model input with a deterministic, human-readable verification program that accepts different valid implementations and reports which prompt requirements pass or fail, rather than comparing against a reference animation or using a VLM judge. We construct these programs with a multi-agent pipeline using human-labeled animations and synthesized test cases. On a balanced test set of 180 animations, our programs achieve 93.3% raw agreement with human judgments, compared with 83.3% for a GPT-6 Astra VLM baseline, and outperform the baseline in both approval and rejection recall. Across 47 configurations from 32 LLMs and 12 providers, pass@1 ranges from 6.0% to 67.7%. Across 15 models evaluated at their highest available reasoning effort with 32 attempts per instance, additional attempts improve every model, but pass@32 still ranges from 38.0% to 95.0%. Our analysis of verification programs reveal that generated animations often fail to match reference poses, follow requested motion trajectories, or coordinate related events. Our benchmark instances can be viewed at `https://motion-graphics-bench.github.io/`.

## 1 INTRODUCTION

Motion graphics animate static imagery through coordinated changes in spatial structure, appearance and visibility, and temporal sequencing. Large language models (LLMs) can now generate such animations from static images and natural language prompts (Tseng et al., 2025; Park et al., 2025), but it remains difficult to evaluate how well the resulting animation realizes the requested motion: a single user request may combine multiple spatio-temporal conditions, but an animation may satisfy only some of it while missing others. Natural language motion requests often describe salient animation behaviors and relative spatial and temporal relationships rather than exact parameters (Talmy, 1975; 1983; Tversky & Lee, 1998). The same input can therefore correspond to several valid animations that differ in their trajectories and timing (Ma et al., 2026). Existing related benchmarks on animation and video often evaluate outputs through metrics that compute resemblance to a target "ground truth" (Chen et al., 2026; Yang et al., 2026; Huang et al., 2024; Ling et al., 2025), or through VLM-based judges that assess rendered frames against the prompt (He et al., 2024; Lin et al., 2025b). Comparison with a single target is too restrictive because a generated animation can satisfy every requested condition while differing from the reference in its trajectory or timing. VLM-based judges avoid comparison with a single reference animation. However, their judgments can be stochastic and it is difficult to inspect how individual prompt requirements affect their decisions.



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.

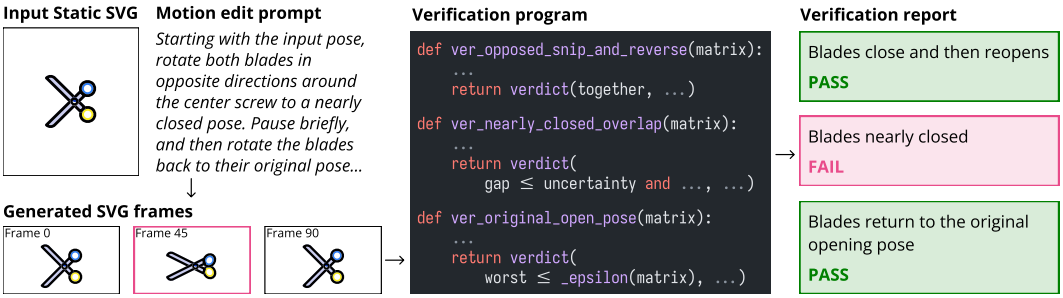



Figure 1: **Overview of MotionGraphicsBench.** Given an input static SVG and a motion edit prompt, our benchmark uses a human-readable verification program to evaluate whether the generated animation satisfies the input requirements and outputs a report. Here, the scissors have the correct motion sequence but fail the verification because the blades remain too far apart when closed.

We introduce MotionGraphicsBench, an expert-curated benchmark of 100 instances for evaluating vector animation generation. We formulate motion graphics generation as an editing task: given an input static SVG and a motion edit prompt in natural language, a model generates an animation by adding requested motions to the static image, which contains all visual elements needed for the edit. This setup mirrors a designer-like workflow in which the static design is first created before motions are added, and isolates motion generation from the separate problem of generating the graphic itself.

To ground our benchmark in real world motion designs rather than synthesized examples, we derive our instances from LottieAnimation-660K, a large collection gathered from professional online animation platforms including IconScout and LottieFiles (Chen et al., 2026). To more comprehensively evaluate model abilities, we curate our benchmark instances with diverse combinations of spatial, temporal, and appearance changes that challenge LLMs while remaining clearly describable in natural language. Our final benchmark covers 17 animation requirement types grouped into spatial structure, appearance and visibility, and temporal sequencing. We rank source animations by structural complexity and manually retain moderately complex, unique motions that can be clearly described in natural language. For each selected source animation, we construct an input static SVG and motion edit prompt through LLM assistance and human review. The SVG provides a reference state for the animated elements while retaining every visual element needed to realize the edit. Our motion edit prompts request salient animation behaviors and their relative spatial and temporal relationships. The prompt states what should move or change, how elements should relate, and when actions should occur without prescribing exact measurements (Talmy, 1975; 1983; Tversky & Lee, 1998; Ma & Agrawala, 2025; Ma et al., 2026).

Because each motion edit prompt specifies different requirements, evaluating model-generated animations requires instance-specific checks. Our deterministic evaluation framework therefore pairs each benchmark instance's model input with a human-readable verification program. Each program consists of deterministic checks corresponding to the requirements in its motion edit prompt. The program measures how visual elements change across the generated SVG frames and returns a verification report with the overall decision and individual check outcomes.

To construct one prompt-specific program for every benchmark instance, we combine human-labeled generated animations with a multi-agent construction pipeline. To support verification program construction and evaluation, we collect human judgments of whether model-generated animations satisfy their motion edit prompts. We use human-labeled construction animations to build the programs and reserve a held-out test set for evaluating the resulting programs. Our pipeline consists of a verification program writer and a test case synthesizer. The writer drafts and revises the program using the motion edit prompt, source Lottie animation, human-labeled construction animations, and synthesized test cases, while the synthesizer creates and reviews those test cases.

Deterministic decisions are useful only if the verification programs capture human judgments of whether an animation satisfies the prompt. We therefore evaluate our programs on 180 human-labeled test animations. Our programs achieve 93.3% raw agreement, 90.0% approval recall, and 96.7% rejection recall. The GPT-6 Astra VLM baseline achieves 83.3% raw agreement, 82.2% approval recall, and 84.4% rejection recall on the same animations.

Using our verification programs, we evaluate 47 configurations from 32 LLMs across 12 providers on all 100 benchmark instances. Across these configurations, pass@1 ranges from 6.0% to 67.7%. From the 47 configurations, we select the highest reasoning effort configuration for 15 models and run 32 independent generation attempts per benchmark instance. Additional attempts improve every model, but pass@32 still ranges from 38.0% to 95.0%. By examining our verification programs and reports, we identify three main failure modes across our benchmark. Generated animations may fail to match reference poses, follow requested motion trajectories, or coordinate related events. Together, MotionGraphicsBench provides a curated benchmark grounded in real world motion designs, deterministic evaluation through interpretable verification programs matching human judgments, and a broad analysis of model performance and generated animation failures. Our benchmark instances can be viewed at `https://motion-graphics-bench.github.io/`.

## 2 Related Work

**Vector animation generation.** Recent vector animation systems either use end-to-end generation or editing setups. LottieGPT and OmniLottie generate both graphics and motion from text or visual conditions, so their evaluations reflect both capabilities (Chen et al., 2026; Yang et al., 2026). Editing methods begin with existing graphics: Keyframer and Decomate support language-guided SVG editing, AniClipart optimizes clipart motion, Vector Prism, LiveSVG, and VAnim animate supplied vector graphics, and MoVer adds and verifies prompt-specified motion in existing SVG scenes (Tseng et al., 2025; Park et al., 2025; Wu et al., 2024a; Yun & Choo, 2026; Levy et al., 2026; Liang et al., 2026; Ma & Agrawala, 2025). Our benchmark follows the editing setup: an input static SVG supplies every required visual element, and a motion edit prompt derived from a source Lottie animation specifies the motion, isolating motion generation from graphic generation.

**Benchmark task construction.** Existing benchmarks either construct prompts around predefined capabilities or curate tasks from existing artifacts or user requests. VBench designs prompts by evaluation dimension and content category, while GenEval, ConceptMix, and T2V-CompBench target specific objects, attributes, relations, motions, or combinations (Huang et al., 2024; Ghosh et al., 2023; Wu et al., 2024b; Sun et al., 2025). MoVer and Self-Consistency similarly use synthetic template-based prompts to evaluate motion relations or trajectory families (Ma & Agrawala, 2025; Ma et al., 2026). DS-1000 derives problems from Stack Overflow, SWE-bench from repository issues and tests, ScienceAgentBench from scientific datasets and tasks, and WildBench from real-user queries (Lai et al., 2023; Jimenez et al., 2024; Chen et al., 2025; Lin et al., 2025a). Our benchmark similarly selects existing Lottie animations and constructs motion edit prompts from their motion designs. Our motion edit propmts span 17 animation requirement types, while each prompt-specific verification program evaluates one instance's exact requirements.

**Verification of generated outputs.** Because several animations can satisfy one motion edit prompt, evaluation should test its requested relationships without requiring a target trajectory. VBench applies dimension-specific metrics to motion smoothness, temporal consistency, and spatial relationships, producing broad capability profiles rather than prompt-specific decisions over all requested behaviors (Huang et al., 2024). Reference-based comparisons in LottieBench and MMLottie-Bench can penalize valid deviations, while VCode and VideoScore use flexible model-based judgments that depend on the judge and prompting procedure (Chen et al., 2026; Yang et al., 2026; Lin et al., 2025b; He et al., 2024). MoVer maps prompts to first-order spatio-temporal predicates but evaluates 5,600 synthetic prompts generated from templates and LLM rewrites. Our benchmark instead derives human-reviewed prompts from existing Lottie animations, pairs each input with a deterministic verification program, and validates decisions against held-out human judgments.

## 3 MotionGraphicsBench

### 3.1 Task Formulation

**Model inputs and output.** To isolate motion generation from graphic creation, we formulate vector animation generation as a motion editing task. We provide each model with an *input static SVG* and a *motion edit prompt* written in natural language, then ask it to generate an animation that realizes the

Table 1: **Benchmark coverage.** An animation requirement type identifies a spatial, appearance, or temporal condition specified by a motion edit prompt. The *Instances* column counts benchmark instances containing at least one type in the family. See Appendix A.2.

| Requirement family | Types | Instances | Highest-coverage types (instances) |
|---|---|---|---|
| Spatial structure | 8 | 100 | *Geometric state constraints* (99); *part attachment and group integrity* (78); *translation trajectories* (72) |
| Temporal sequencing | 5 | 100 | *Temporal ordering* (100); *temporal coordination and phase* (93); *animation speed and easing* (93) |
| Appearance and visibility | 4 | 65 | *Visibility and opacity* (53); *occlusion and visible-region containment* (30); *color, fill, and stroke appearance* (13) |

requested motion. An *input static SVG* is a vector scene that contains all scene geometry, including elements that may not be visible when rendered. A *motion edit prompt* describes how the input static SVG should be animated, including what should move or change, how elements should relate, and when actions should occur. Because it is often difficult for users to specify exact keyframes, coordinates, or frame numbers (e.g., how many degrees the scissor blades in Figure 1 should rotate), each prompt in our benchmark requests salient animation behaviors in terms of relative spatial and temporal relationships, consistent with findings from cognitive psychology on how people describe animation (Talmy, 1975; 1983; Tversky & Lee, 1998). As a result, a prompt often does not fully specify a unique animation (Ma et al., 2026). The input static SVG provides a reference appearance, but it does not necessarily define frame 0. The prompt may ask the model to construct a different frame 0 based on the input SVG or to make the animation reach the SVG's depicted state later (e.g., the *Scissors* prompt asks the blades to return to the open pose shown in the input SVG in Figure 1).

**Evaluating model outputs.** Multiple model outputs may therefore satisfy the same prompt while differing in their trajectories and timing. To evaluate a model output against animation requirements rather than one target animation, each *benchmark instance* pairs an input static SVG and motion edit prompt with a prompt-specific verification program. The program encodes the animation requirements as deterministic, human-readable checks. Section 4 describes how our evaluation framework applies the program to model generated animations.

### 3.2 Model Input Construction

**Source Lottie corpus selection.** To ground our benchmark in real-world motion designs rather than synthetic examples, we select 100 source Lottie animations from LottieAnimation-660K, a collection gathered from professional animation platforms (Chen et al., 2026). Lottie preserves layers, shapes, transforms, keyframes, and easing curves, which lets us identify the elements and motions needed to construct each model input. We use each source Lottie animation only to construct the input static SVG and motion edit prompt; evaluated models do not receive it or need to reproduce it exactly. To keep the task focused on motion generation rather than identifying unnamed SVG elements, we retain animations whose animated layers or objects have names that support clear references in a motion edit prompt. We rank the remaining candidates by their animated property types, animated objects, and keyframes to capture motion variety, scene complexity, and temporal structure. We manually review 1,000 candidates near the median complexity to balance challenge and describability, removing duplicate graphics and motions and animations that cannot be clearly described in natural language.

**Input static SVG construction.** To model a workflow in which an artist supplies an existing static design, we exclude frames that are empty, cropped, or transitional. We select a remaining frame in which the visible objects are recognizable, opaque, and normally sized. To keep our benchmark focused on motion editing rather than graphic creation, the input static SVG must contain every visual element needed by the requested animation, including elements hidden in the selected frame. A single-frame Lottie render can omit elements that are inactive or fully transparent at the selected moment, even when the requested animation needs them later. We therefore recover this hidden geometry while preserving the selected frame's visible appearance (Liang et al., 2026). The completed SVG preserves the reference appearance while making all required graphics available for animation. Appendix Figure 6 illustrates the process.

**Motion edit prompt construction.** Because the source Lottie animations do not include model-facing instructions for adding motion to a static SVG, we construct each motion edit prompt through LLM-assisted drafting and human review. As motivated in Section 3.1, each prompt states what should move or change, how elements should relate, and when actions should occur. We describe these requirements through relative spatial and temporal relationships rather than exact keyframes, coordinates, or frame numbers (Ma & Agrawala, 2025). When needed, a prompt specifies a visible path, timing pattern, or easing behavior without requiring exact reproduction of the source Lottie animation (Ma et al., 2026). An LLM drafts each prompt from the input static SVG and evidence from the source Lottie animation. The draft names the relevant elements and states their required spatial and temporal relationships, including end states. These details distinguish a complete edit from one that captures only the general motion. A human reviewer compares each draft with the input static SVG and source Lottie animation, then approves it, edits it, or requests a targeted revision to remove ambiguity, unsupported claims, and implementation-specific wording. See Appendix A.1.

### 3.3 Benchmark Composition and Coverage

Our MotionGraphicsBench contains 100 benchmark instances constructed from 100 distinct source Lottie animations. Each instance pairs an input static SVG and a human-reviewed motion edit prompt with a prompt-specific verification program. The source Lottie animations have a mean duration of 3.50 seconds (SD 1.89). The motion edit prompts have a mean length of 193.8 words (SD 57.2). To characterize the animation behaviors requested by our motion edit prompts, we identify 17 animation requirement types grouped into three families: spatial structure, appearance and visibility, and temporal sequencing. (Table 1). Our benchmark centers on the temporal ordering and geometric state constraints needed to specify coherent motion, while extending its coverage to deformation, reveal and erasure, and color, fill, and stroke appearance. Each motion edit prompt combines between 7 and 14 animation requirement types. All prompts combine spatial and temporal requests; 65 include appearance or visibility.

## 4 Evaluation with Verification Programs

### 4.1 Verification of Generated Animations

**Prompt-specific verification.** Motion edit prompts describe intended spatial and temporal behavior without fully specifying an exact trajectory, timing curve, or grouping (Talmy, 1975; 1983; Tversky & Lee, 1998). Multiple animations can therefore satisfy the same prompt (Ma & Agrawala, 2025; Ma et al., 2026). Because each prompt imposes different requirements, we evaluate each instance with a prompt-specific verification program rather than shared similarity measures. The program implements the prompt requirements as verification checks that evaluate spatial structure, appearance and visibility, and temporal sequencing as applicable to the instance. These checks allow different animation implementations to pass when they satisfy the requested behavior while detecting partial results that generic visual metrics may miss. We describe how we construct a verification program for each instance in Section 4.2.

**Verification procedure.** Vector animations can be created through many mechanisms, such as Lottie, GSAP, anime.js, CSS animations, and SMIL (Chen et al., 2026; Doyle, 2025; Garnier, 2025; MozDevNet, 2025; W3C SYMM Working Group, 2008). Animations produced through any of these mechanisms can be represented as sequences of SVG frames. Because each frame records the vector scene at one point in time, the resulting sequence provides a common representation that allows verification to operate independently of how the motion is produced. In our experiments, we ask models to generate JavaScript programs with GSAP, whose explicit timeline allows our renderer to sample SVG frames deterministically at any frame rate. The verification program takes the SVG frames as input, measures each element's geometry, appearance, and motion over time, evaluates the prompt requirements, and returns a verification report with the overall decision, individual check outcomes, and any errors. An attempt is one model request for one benchmark instance and counts as a success only if it produces an animation that passes every verification check.

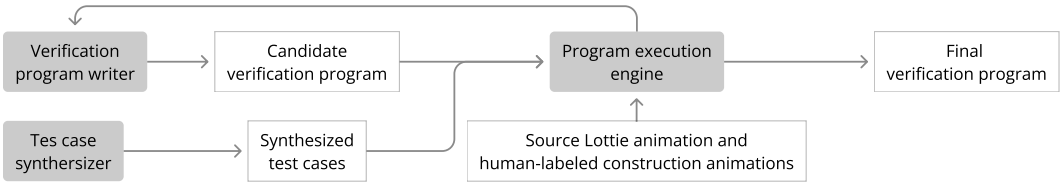



Figure 2: **Verification program construction.** The writer drafts and revises a candidate program using the benchmark inputs, source Lottie animation, and human-labeled construction animations, while the synthesizer independently creates test case animations. We run the candidate on these real and synthesized animations, and then return the reports to the writer for revision.

### 4.2 Verification Program Construction

Because a motion edit prompt can be satisfied by several valid implementations, each verification program must accept allowed variation while rejecting animation requirement violations. We construct one program for each instance with a pipeline of two agents: we have a *verification program writer* that writes and revises verification programs, while a *test case synthesizer* creates synthesized test case animations that expose missing or overly restrictive checks (Figure 2). Both agents use GPT-6 Astra.

**Construction data.** To ground construction in diverse model outputs, we use a human-labeled construction set of 1,585 model-generated animations from 25 generation configurations across all 100 benchmark instances, comprising 410 approvals and 1,175 rejections. Appendix Table 6 marks these 25 configurations and the four held-out configurations. These animations come from the broader collection of human judgments described in Section 5, where the same review procedure with two annotators covers construction animations and test animations reserved for evaluation. Before construction, we split the animations; agents receive only construction animations, review notes, instance inputs, and the source animation (Appendix C.1).

**Verification program writer.** To keep each verification program grounded in the requested motions, the *verification program writer* receives a *construction example set* alongside the input static SVG and motion edit prompt. The set consists of the source Lottie animation, human-labeled construction animations, and synthesized test cases. The source Lottie animation supplies one valid realization of the requested edit, while examples approved by humans show allowed variation and examples rejected by humans and their review notes identify possible violations to inspect. Together, these inputs help the writer interpret the requested edit without treating one animation as the definition of correctness or adding requirements absent from the input static SVG or motion edit prompt. The writer uses an *animation matrix* of SVG attributes and measurements; Appendix B.1 gives implementation details. The writer therefore bases its measurements and tolerances on the requested behavior and measured animation properties rather than choosing thresholds solely to match the construction approvals and rejections. Using our *verification library*, which derives spatio-temporal measurements from the animation matrix, the writer implements these requirements as a Python verification program (Figure 1). During construction, the writer tests each candidate program on the complete construction example set. Each candidate must classify every example in the set correctly: the source Lottie animation and every approved or positive example must pass, while every rejected or negative example must fail.

**Test case synthesizer.** Human-labeled construction animations and the source Lottie animation may leave a requirement untested or combine several violations in one example, causing a candidate verification program to become too permissive or overly restrictive. The *test case synthesizer* therefore creates synthesized test cases that probe allowed variation and targeted violations. It receives the motion edit prompt, source Lottie animation, human-labeled construction animations and review notes, and descriptions of earlier cases, but not the candidate program or its reports. Using our *animation transformation library*, which deterministically changes motion timing, direction, extent, position, appearance, and SVG structure, it transforms the source Lottie animation or a construction animation approved by humans. Positive cases preserve the requested edit while varying allowed choices, whereas negative cases introduce a violation. When useful, paired cases isolate one relevant property. To account for the imbalance between approvals and rejections, the synthesizer requests more negative cases when approvals are more common and more positive cases when rejections are

Table 2: We compare our prompt-specific verification programs against a VLM baseline on the same balanced test set of 180 animations. Raw agreement is the fraction matching human labels; approval and rejection recall are the fractions of human approvals and rejections correctly identified.

| Method | Raw Agreement | Approval Recall | Rejection Recall |
|---|---|---|---|
| VLM baseline | 83.3% | 82.2% | 84.4% |
| Our programs | **93.3%** | **90.0%** | **96.7%** |

more common. The assigned labels remain claims made during construction rather than independent human judgments. See Appendix B.2.

**Testing and revision.** After the writer produces a candidate verification program, it runs the program on the complete construction example set, inspects the individual check outcomes, revises the candidate when the tests expose missing or overly restrictive checks, and reruns the set. A synthesized-label disagreement may reveal a faulty test case, which the agents resolve within the loop (Appendix B.2). This iterative construction loop resembles counterexample-guided synthesis and selection of generated tests (Solar-Lezama et al., 2008; Chen et al., 2022; Jain et al., 2025; He et al., 2026). The run produces one verification program per instance, and we freeze the resulting complete set for evaluation on the reserved test set (Section 5).

## 5 Measuring Alignment with Human Judgment

**Human judgment collection.** We collect human judgments for 1,980 model-generated animations, and use 1,585 of those for verification-program construction (Section 4.2) and reserve a disjoint set of 180 animations to evaluate how well the resulting verification programs align with human judgment. We assign two annotators to independently inspect the motion edit prompt, input static SVG, and generated animation and judge prompt satisfaction rather than aesthetic preference or exact source reproduction. Annotators may reveal the source Lottie animation as one valid clarification (Appendix Figure 7). To keep judgments independent, we hide the generating model, verification decision, and the other annotator's initial decision. When their decisions differ, the same pair revisits the animation and discusses the prompt, and we assign a binary final label only after agreement, following established independent-annotation and consensus practice (Banea et al., 2008; Hromei et al., 2025). For rejections, annotators identify unsatisfied prompt text and the visible violation without adding new prompt requirements; these comments support the decision but need not enumerate every failure. See Appendix C.1.

**Held-out evaluation.** We evaluate 180 held-out animations, balanced between 90 human approvals and 90 human rejections, covering all 100 benchmark instances. The animations come from Gemini 3.1 Pro, Gemma4, GPT-5.5, and Sonnet 5, spanning multiple providers and both open-weight and proprietary models. For comparison, we use a VLM baseline that receives the input static SVG rendering, motion edit prompt, and up to 64 timestamped frames from each generated animation. See Appendix C.2.

**Validation results.** Our verification programs agree with human judgments on 93.3% of test animations, 10% above the VLM baseline at 83.3%. They recover 90.0% of human approvals and 96.7% of human rejections, compared with 82.2% and 84.4% for the VLM baseline. Our programs also achieve higher raw agreement within each of the four generation configurations represented in the test set. To understand the remaining errors, we review all 12 cases in which a verification program decision differs from the final human label. The cases span 11 benchmark instances and comprise nine false rejections across eight instances and three false approvals across three instances, so they are not concentrated in one instance. In all three false approvals, our programs miss required relative motion between sub-elements. Most false rejections enforce timing, coordination, or geometry too strictly; Appendix C.2 gives the complete audit.

## 6 Model Performance and Failure Analysis

We report pass@1 for 47 model configurations representing 32 LLMs from 12 providers, including 16 open-weight and 16 proprietary models. For pass@k, we select the highest-available-effort

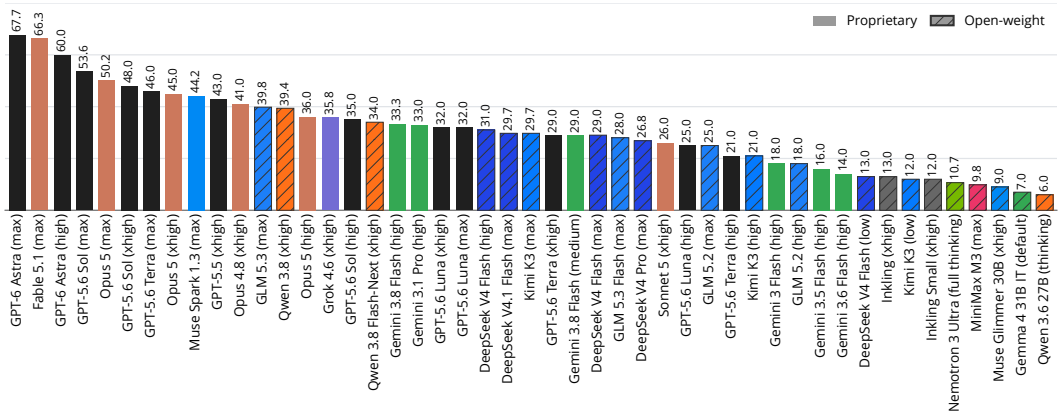



Figure 3: **Pass@1 results.** We report pass@1 for 47 configurations from 32 LLMs and 12 providers. The 16 proprietary models average 38.8% pass@1, while the 16 open-weight models average 21.8%

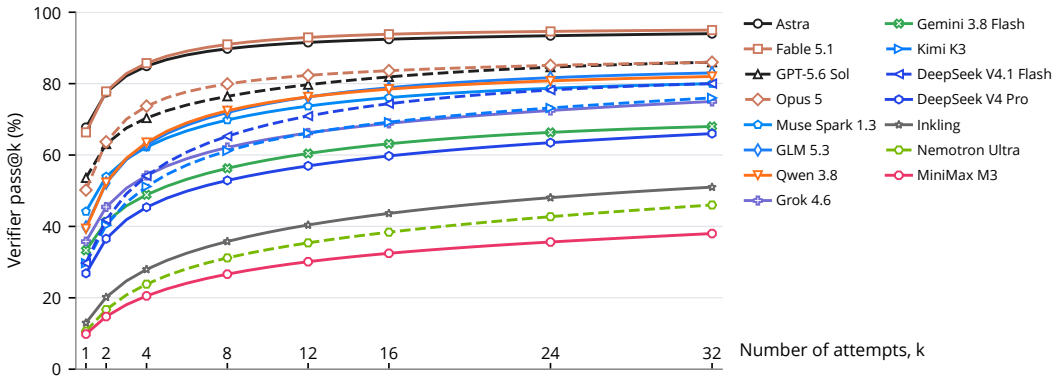



Figure 4: **Pass@$k$ results.** The curves evaluate every integer attempt budget from 1 through 32 for the same 15 configurations and 100 benchmark instances.

configuration from 15 models and generate 32 independent attempts per benchmark instance. Each attempt asks one model to generate one animation for one benchmark instance. This design yields 51,200 attempts across the 100 benchmark instances. All model scores use a separate generation batch from human-reviewed sets. Our verification programs evaluate rendered SVG frames independently of the animation format. For a controlled comparison, here we ask every model to generate a JavaScript program with GSAP, which has been the industry-standard SVG animation library for over a decade and is used across millions of websites, making its API likely to be well represented in model training data (Doyle, 2025). See Appendix D.1.

### 6.1 Model Performance

**Overall performance.** Figure 3 shows the complete model and setting comparison. Across all 47 configurations, mean pass@1 is 30.3%. To avoid giving configurations with more attempts extra weight, we compute each configuration's fraction of successful attempts on each benchmark instance. We average these fractions across the 100 instances, then average the 47 configuration scores equally. To compare the two model groups, we use one highest-effort configuration per model: the 16 proprietary models average 38.8% pass@1 and the 16 open-weight models average 21.8%. Proprietary configurations occupy the highest ranks, although GLM 5.3 and Qwen 3.8 each reach approximately 40% and outperform several proprietary configurations. See Appendix D.2.

**Additional attempts and performance plateaus.** Figure 4 reports success across attempt budgets for the 15 configurations with 32 independent attempts per benchmark instance. Mean performance increases from 36.7% at pass@1 to 73.7% at pass@32, with diminishing gains as the attempt budget increases. Higher-performing configurations plateau earlier, while lower-performing ones benefit

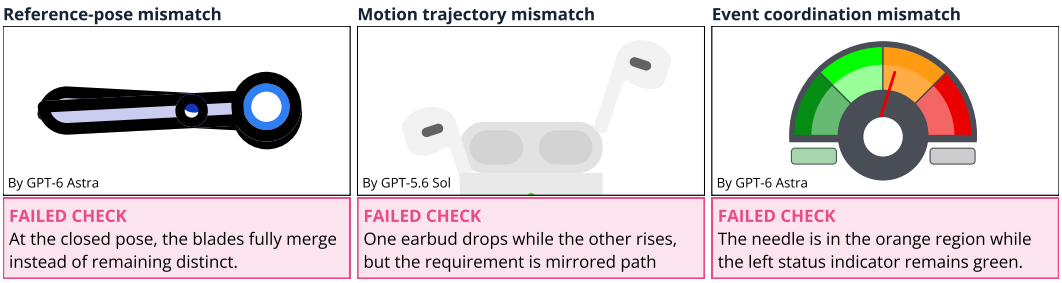



Figure 5: **Failure modes.** We analyze our verification programs to identify top model failure modes.

longer; Appendix D.2 gives the full comparison. Success also varies across instances: over all 47 configurations, the median instance-level pass@1 is 20.6%. Among 29,632 completed, error-free failed attempts in the 15-configuration cohort, 93.6% pass at least one verification check and 31.6% fail exactly one. Most failed animations therefore satisfy part of the motion edit prompt, and many are near misses. Appendix D.2 reports difficulty and partial-satisfaction analyses.

### 6.2 Model Failure Modes

Because our verification programs contain human-readable checks tied to specific animation requirements, we can use their outcomes to identify which requirements an animation satisfies or misses, rather than only whether it passes. We map checks to failure modes and apply them to saved reports; Appendix D.3 gives the full method. Appendix D.4 compares failure profiles of near-tied models. We provide additional qualitative examples in Appendix D.4 and on our website. For each mode, we report failure over attempts on instances whose prompts request the corresponding animation behavior. Coverage is the share with enough evidence to decide whether the mode occurred.

We most often observe **reference-pose mismatch**, with a 41.5% failure rate (94.3% coverage). This spatial-structure failure occurs when an element misses a pose shown by the input static SVG or requested by the motion edit prompt. In the Scissors example, the blades close but merge instead of remaining distinct (Figure 5). Beyond reaching the correct pose, models also struggle with **motion trajectory mismatch**, or errors in path, direction, or distance. We observe a 38.8% failure rate (88.6% coverage) for this second spatial-structure failure. In TWS Earbuds, one earbud descends instead of following the requested mirrored upward-and-outward path (Figure 5). For temporal sequencing, **event coordination mismatch** has a 32.4% failure rate (93.0% coverage). This occurs when related motions have the wrong order or overlap. In Speed-2, the green status remains visible after the needle leaves its range instead of fading as it departs (Figure 5).

## 7 Conclusion

We introduced MotionGraphicsBench with 100 expert-curated instances that pair an input static SVG and motion edit prompt with a prompt-specific verification program. On a balanced test set of 180 animations, our programs achieve 93.3% raw agreement, compared with 83.3% for a VLM baseline. Across 47 configurations from 32 LLMs and 12 providers, no configuration exceeds 67.7% pass@1, while additional attempts improve success substantially but unevenly within the 15-model cohort. Our failure analysis shows that generated animations often fail to match reference poses, follow requested motion trajectories, or coordinate related events.

**Limitations and future work.** Our detailed motion edit prompts, while effective for precise verification, do not capture interactions where a user begins with shorter or underspecified requests and refines them over multiple turns. Moreover, although our framework evaluates SVG frames independently of the animation generation mechanism, our experiments uses the GSAP libray; other mechanisms such as CSS, SMIL, and other JavaScript libraries may yield different performances. Our 100 instances also do not cover complex deformation, large scenes, or highly interactive animation behavior. Future work can use our check-level reports for MoVer-style iterative correction (Ma & Agrawala, 2025), online or offline reinforcement learning, and multi-turn refinement that combines verification with user feedback.

### AI USE STATEMENT

AI systems play two central roles in this work, both described in the main text. First, we use LLMs to assist with input static SVG frame selection, motion edit prompt drafting and revision, verification program construction, synthesized test-animation generation, and the comparison baseline in our held-out validation (Sections 3–5). Second, the 32 LLMs in our model study generate the animations evaluated by our benchmark (Section 6). Outside these core uses, we use generative AI assistants to provide feedback on research methodology and experiments, implement and edit software, support failure-pattern annotation and analysis, identify relevant literature, and draft and edit manuscript text. Human reviewers make the final decisions for motion edit prompts and animation labels. We check cited sources, AI-assisted research code, experimental artifacts, analyses, and reported claims, make all final methodological and interpretive decisions, and take responsibility for the paper and every AI-assisted artifact.

### ETHICS STATEMENT

Our benchmark uses source animations from LottieAnimation-660K and human judgments of model-generated animations. Annotators judge only whether an animation satisfies its motion edit prompt; we report task judgments and aggregate counts and do not analyze annotator identities or sensitive attributes. The source corpus may unevenly represent visual styles, languages, and creators, so our results should not be interpreted as a universal measure of animation quality or creativity. We execute generated programs in isolated processes with networking and external assets disabled to reduce risks from model-produced code. More capable systems could help artists create animations, but they could also automate parts of creative work or reproduce existing designs too closely. Users should follow applicable copyright and licensing requirements.

### REPRODUCIBILITY STATEMENT

Sections 3–6 describe our benchmark, evaluation, validation, and model study, with corresponding reproducibility details in Appendices A–D. We will release the benchmark, model-generated data, and the accompanying code and artifacts needed to reproduce and audit our results. Our evaluation is deterministic for fixed generated outputs and verification artifacts. Because hosted model APIs may change, regenerated outputs may differ, but the released outputs and reports preserve the reported results for inspection.

## REFERENCES

Carmen Banea, Rada Mihalcea, Janyce Wiebe, and Samer Hassan. Multilingual subjectivity analysis using machine translation. In Mirella Lapata and Hwee Tou Ng (eds.), *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pp. 127–135, Honolulu, Hawaii, October 2008. Association for Computational Linguistics. URL https://aclanthology.org/D08-1014/.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. Codet: Code generation with generated tests, 2022. URL https://arxiv.org/abs/2207.10397.

Junhao Chen, Kejun Gao, Yuehan Cui, Mingze Sun, Mingjin Chen, Shaohui Wang, Xiaoxiao Long, Fei Ma, Qi Tian, Hao Zhao, and Ruqi Huang. Lottiegpt: Tokenizing vector animation for autoregressive generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 31639–31651, June 2026.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and Huan Sun. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 96934–96990, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/f12b4df26344f3be803c06b555252efe-Paper-Conference.pdf.

Jack Doyle. Greensock animation platform v3, 2025. URL https://gsap.com/docs/v3/.

Julien Garnier. anime.js · javascript animation engine, 2025. URL https://animejs.com/.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 52132–52152. Curran Associates, Inc., 2023. doi: 10.52202/075280-2270. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a3bf71c7c63f0c3bcb7ff67c67b1e7b1-Paper-Datasets_and_Benchmarks.pdf.

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 Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2105–2123, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.127. URL https://aclanthology.org/2024.emnlp-main.127/.

Zhongmou He, Yee Man Choi, Kexun Zhang, Ivan Bercovich, Jiabao Ji, Junting Zhou, Dejia Xu, Aidan Zhang, Yixiao Zeng, and Lei Li. Hardtestgen: A high-quality rl verifier generation pipeline for llm algorithimic coding. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 35671–35715, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/3c11cea35aa2e4797b2f239da378927b-Paper-Conference.pdf.

Claudiu Daniel Hromei, Antonio Scaiella, Danilo Croce, and Roberto Basili. Grounded semantic role labelling from synthetic multimodal data for situated robot commands. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 23747–23770, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1212. URL https://aclanthology.org/2025.emnlp-main.1212/.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818, June 2024.

Kush Jain, Gabriel Synnaeve, and Baptiste Roziere. Testgeneval: A real world unit test generation and test completion benchmark. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 14947–14999, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/26ded5c8ee8ec1bc4caced4e1c9b1584-Paper-Conference.pdf.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/edac78c3e300629acfe6cbe9ca88fb84-Paper-Conference.pdf.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-Tau Yih, Daniel Fried, Sida Wang, and Tao Yu. DS-1000: A natural and reliable benchmark for data science code generation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 18319–18345. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/lai23b.html.

Matan Levy, Ran Margolin, Bar Cavia, Dvir Samuel, Yael Pritch, Shmuel Peleg, Alex Rav Acha, Ariel Shamir, and Dani Lischinski. Livesvg: Zero-shot svg animation via video generation, 2026. URL https://arxiv.org/abs/2605.30174.

Guotao Liang, Zhangcheng Wang, Chuang Wang, Juncheng Hu, Haitao Zhou, Junhua Liu, Jing Zhang, Dong Xu, and Qian Yu. Vanim: Rendering-aware sparse state modeling for structure-preserving vector animation, 2026. URL https://arxiv.org/abs/2605.01517.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 47852–47870, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/771155abaae744e08576f1f3b4b7ac0d-Paper-Conference.pdf.

Kevin Qinghong Lin, Yuhao Zheng, Hangyu Ran, Dantong Zhu, Dongxing Mao, Linjie Li, Philip Torr, and Alex Jinpeng Wang. Vcode: a multimodal coding benchmark with svg as symbolic visual representation, 2025b. URL https://arxiv.org/abs/2511.02778.

Xinran Ling, Chen Zhu, Meiqi Wu, Hangyu Li, Xiaokun Feng, Cundian Yang, Aiming Hao, Jiashu Zhu, Jiahong Wu, and Xiangxiang Chu. Vmbench: A benchmark for perception-aligned video motion generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13087–13098, October 2025.

Jiaju Ma and Maneesh Agrawala. Mover: Motion verification for motion graphics animations. *ACM Trans. Graph.*, 44(4), July 2025. ISSN 0730-0301. doi: 10.1145/3731209. URL https://doi.org/10.1145/3731209.

Jiaju Ma, R. Kenny Jones, Jiajun Wu, and Maneesh Agrawala. Self-consistency for llm-based motion trajectory generation and verification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17357–17366, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Ma_Self-Consistency_for_LLM-Based_Motion_Trajectory_Generation_and_Verification_CVPR_2026_paper.html.

MozDevNet. Using css animations - css: Cascading style sheets: Mdn, 2025. URL https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animations/Using_CSS_animations.

Jihyeon Park, Jiyoon Myung, Seone Shin, Jungki Son, and Joohyung Han. Decomate: Leveraging generative models for co-creative svg animation, 2025. URL https://arxiv.org/abs/2511.06297.

Armando Solar-Lezama, Christopher Grant Jones, and Rastislav Bodik. Sketching concurrent data structures. In *Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '08, pp. 136–148. ACM, June 2008. doi: 10.1145/1375581.1375599. URL http://dx.doi.org/10.1145/1375581.1375599.

Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. T2v-compbench: A comprehensive benchmark for compositional text-to-video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8406–8416, June 2025.

Leonard Talmy. *Semantics and Syntax of Motion*, pp. 181 – 238. Brill, Leiden, The Netherlands, 1975. ISBN 9789004368828. doi: 10.1163/9789004368828_008. URL https://brill.com/view/book/edcoll/9789004368828/BP000008.xml.

Leonard Talmy. How language structures space. In *Spatial orientation: Theory, research, and application*, pp. 225–282. Springer, 1983.

Tiffany Tseng, Ruijia Cheng, and Jeffrey Nichols. Keyframer: Empowering animation design using large language models, 2025. URL https://arxiv.org/abs/2402.06071.

Barbara Tversky and Paul U Lee. How space structures language. In *Spatial cognition: An interdisciplinary approach to representing and processing spatial knowledge*, pp. 157–175. Springer, 1998.

W3C SYMM Working Group. Synchronized multimedia integration language (SMIL 3.0). W3C Recommendation, https://www.w3.org/TR/SMIL/, December 2008.

Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. Aniclipart: Clipart animation with text-to-video priors. *International Journal of Computer Vision*, 133(6):3149–3165, December 2024a. ISSN 1573-1405. doi: 10.1007/s11263-024-02306-1. URL http://dx.doi.org/10.1007/s11263-024-02306-1.

Xindi Wu, Dingli Yu, Yangsibo Huang, Olga Russakovsky, and Sanjeev Arora. Conceptmix: A compositional image generation benchmark with controllable difficulty. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 86004–86047. Curran Associates, Inc., 2024b. doi: 10.52202/079017-2731. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/9c3563bbeb2ad7f3b3b8ed0fcd3b440f-Paper-Datasets_and_Benchmarks_Track.pdf.

Yiying Yang, Wei Cheng, Sijin Chen, Honghao Fu, Xianfang Zeng, Yujun Cai, Gang Yu, and Xingjun Ma. Omnilottie: Generating vector animations via parameterized lottie tokens. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 39293–39303, June 2026.

Jooyeol Yun and Jaegul Choo. Vector prism: Animating vector graphics by stratifying semantic structure. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17316–17325, June 2026.

# A MotionGraphicsBench

## A.1 Details on Model Input Construction

**Source ranking and review.** For each candidate animation, $T$ is the number of animated property types, $O$ is the recorded animated-object signal, and $K$ is the total number of keyframes. The frozen implementation counts an object once for each animated property type, so an object with several animated properties can contribute more than once to $O$. We exclude candidates with $O > 10$ to avoid scenes in which many objects repeat the same motion without adding motion variety. For the remaining candidates, we convert each measure to a midrank percentile $P$ and compute

$$C = P(T) + 0.25\,P(O) + P(K).$$

We downweight $O$ to prevent animations with many animated elements from dominating the ranking. We manually review 1,000 candidates near the median $C$, remove duplicate graphics and motions and animations that cannot be clearly described in natural language, and retain the final 100 source Lottie animations.

**Input static SVG frame selection.** We sample five frames from each source Lottie animation and use GPT-5.6 Sol to propose the frame that best represents a static design before motion is added. The selection criteria require the main objects to be visible, recognizable, opaque, normally sized, and shown in a representative pose; empty, cropped, transitional, effect-heavy, or confusing states are unsuitable. A human reviewer confirms the proposal or selects a better sampled frame.

**Geometry completion.** A normal single-frame Lottie render can leave path geometry empty when a layer falls outside its activity window or has zero effective opacity. We widen every layer's activity window and render the selected timestamp with transparency-based culling disabled to recover each path's geometry, transform, and paint. We then restore the selected frame's hidden-group visibility so that the completed SVG retains the same visible appearance. The audit checks structural alignment and empty paths, and flags failed rows.

**Motion edit prompt drafting and revision.** GPT-5.6 Sol receives the completed input static SVG, its PNG rendering, sampled source Lottie animation frames, and the source Lottie JSON file. The

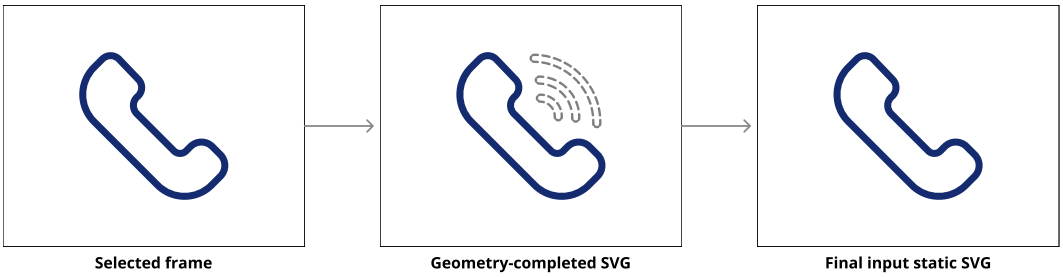



Figure 6: **Input static SVG construction.** We select an input static SVG frame in which the main objects are visible and shown in a representative pose. During geometry completion, we recover omitted source paths and restore their original visibility state. The dashed outline exposes three ringing-arc paths that remain stored but not visible in the final input static SVG.

drafting prompt asks for the relevant elements and their spatial and temporal relationships, including direction, repetition, order, overlap, pivots, and end states. A human reviewer approves or edits the draft, or provides a critique for targeted revision using the same source evidence.

**Construction prompts.** Following prior motion-graphics work (Ma & Agrawala, 2025), we show the prompts used for input static SVG frame selection, motion edit prompt drafting, and targeted revision. Angle-bracketed fields denote per-instance inputs.

To select the input static SVG frame, we ask GPT-5.6 Sol to compare five sampled renders against these criteria and return the selected frame with a confidence and concise rationale.

```
You are selecting the best input static SVG frame for a motion-edit benchmark.

Given five sampled PNG frames in time order and the source Lottie JSON, choose the single
    sampled frame that best represents a static design before motion is added.

Prefer a representative or neutral frame in which the main objects are visible, recognizable,
    opaque, and normally sized. Choose the first frame when it is already suitable, or the
    last when it is more complete or representative. Avoid empty, cropped, transitional,
    effect-heavy, or visually confusing frames. Treat the images as the source of truth and
    the Lottie JSON only as supporting context.

Inputs:
  <FIVE_SAMPLED_PNG_FRAMES_IN_TIME_ORDER>
  <SOURCE_LOTTIE_JSON>

Choose one provided frame label and return exactly:
<static_input_svg_selection>
{"frame_index": <SAMPLED_FRAME_INDEX>,
 "confidence": "<high|medium|low>",
 "rationale": "<ONE_CONCISE_SENTENCE>"}
</static_input_svg_selection>
```

To draft the motion edit prompt, we ask GPT-5.6 Sol to describe the concrete motion mechanics in one paragraph. The prompt first establishes any frame-0 state that differs from the supplied SVG, then describes each object's motion, and finally specifies the spatial and temporal relationships across motions.

```
You are an experienced motion graphics animator. Write a motion edit prompt that lets a human
    or model animate the supplied static SVG to reproduce the observed motion.

Inputs:
  <COMPLETED_INPUT_STATIC_SVG>
  <STATIC_SVG_PNG_PREVIEW>
  <SAMPLED_SOURCE_ANIMATION_FRAMES>
  <SOURCE_LOTTIE_JSON>

Write one motion edit prompt as a single paragraph.

The motion edit prompt must:
- First state the frame-0 pose when it differs from the supplied SVG. Describe this as a state
    to set, not a motion to play.
```

```
- Then describe each object's or group's motion, followed by the spatial and temporal
    relations among those motions.
- Identify objects by what they depict and disambiguate similar ones by position, color, or
    another visible property.
- Use concrete transform verbs and trajectories. State the first direction, exact repeat
    counts, pivots or fixed anchors, visible motion amounts relative to the artwork, easing
    when relevant, and end states relative to the supplied SVG.
- State order, overlap, simultaneity, lead or lag, and phase coordination. For visibility or
    appearance changes, state whether each change is instant or gradual and when it occurs.
- Focus on what moves or changes. Every element already exists, so say reveal or show rather
    than add or create.
- Do not describe the animation frame by frame or mention frame numbers, FPS, pixel
    coordinates, JSON, Lottie internals, or implementation details.

Return exactly:
<motion_edit_prompt>
### Motion edit prompt
<CONCRETE_ANIMATION_MECHANICS>
</motion_edit_prompt>
```

For targeted revision, we provide the prior draft and reviewer critique and ask GPT-5.6 Sol to change only what the critique requires while preserving unaffected wording and detail.

```
Use the same source inputs as the drafting prompt, together with:
  <PRIOR_MOTION_EDIT_PROMPT>
  <REVIEWER_CRITIQUE>

Make the smallest changes needed to resolve the reviewer critique and any clear violations of
    the drafting rules. Preserve unaffected wording, sentence structure, ordering, and
    animation-specific detail verbatim.

If no prior prompt exists, write a fresh motion edit prompt guided by the reviewer critique.
    Return the result in the same <motion_edit_prompt> structure.
```

### A.2 Animation Requirement Types and Coverage

Table 3 defines the 17 overlapping animation requirement types used to label each final motion edit prompt and supplied scene. Each row counts distinct benchmark instances with that type, so one instance can appear in several rows. We resolved the seven uncertain source cells as two present and five absent, leaving none unresolved. The types describe prompt content rather than independent model skills, model performance, or verification-check templates.

## B Evaluation with Verification Programs

### B.1 Evaluation Implementation and Scoring

**Frame rendering.** We use the rendering engine provided by the open-source MoVer repository to execute each generated GSAP program in a browser and produce the SVG frame sequence consumed by verification (Ma & Agrawala, 2025).

**Animation matrix.** To give each verification program a structured view of SVG elements over time, we follow MoVer and use an *animation matrix* (Ma & Agrawala, 2025). MoVer organizes an animation as a two-dimensional matrix with objects as rows and frames as columns. Each cell stores atomic motion attributes for one object at one frame, while static object information is stored separately. Our matrix serves a related purpose but stores one row for each measured SVG node and keys each row by its document path, which remains unambiguous when element identifiers are missing or duplicated. For every frame, Chromium records the element hierarchy, absolute and element-owned transforms, local bounding box, outline length and endpoints, sampled contour, stroke width, visible fraction, clip and mask bounds, SVG attributes, and relevant computed styles. The matrix also records the frame count, frame rate, duration, source-frame indices, and canvas dimensions. Our verification library derives higher-level geometry, motion, timing, visibility, and inter-element relations from these measurements.

**Verification checks and reports.** Each verification program contains readable Python checks that receive the animation matrix and test one prompt requirement. Each check returns a pass or fail

Table 3: **Animation requirement type definitions and coverage.** Counts report distinct benchmark instances whose motion edit prompts request each type. Because labels overlap, the counts do not sum to 100.

| Family and requirement type | Definition | Distinct instances |
|---|---|---|
| **Spatial structure** | | |
| Geometric state constraints | Required pose, position, orientation, dimensions, or contour at a specified phase or interval. | 99 |
| Part attachment and group integrity | Parts remain attached or inherit a common group motion or deformation. | 78 |
| Translation trajectories | Positional movement with a requested direction, path, or distance. | 72 |
| Transform origins and anchors | A requested rotation or scaling center, or a point or edge that remains anchored. | 71 |
| Relative position, alignment, and contact | Required alignment, spacing, contact, separation, or geometric containment relative to another element. | 66 |
| Rotation direction and magnitude | Changes in orientation, including turning direction, amount, rocking, or reversal. | 56 |
| Scale and aspect-ratio changes | Changes in a target's size, length, width, height, or proportions. | 37 |
| Shape and contour deformation | Local changes to a target's contour or form. | 6 |
| **Appearance and visibility** | | |
| Visibility and opacity | Appearing, disappearing, fading, or explicitly preserving visibility or opacity. | 53 |
| Occlusion and visible-region containment | Visible covering between elements or restriction of visible paint to a region. | 30 |
| Color, fill, and stroke appearance | Changes to or preservation of color, fill, linework, or stroke thickness. | 13 |
| Progressive reveal and erasure | Artwork exposed or removed progressively across a region, scan, or stroke. | 7 |
| **Temporal sequencing** | | |
| Temporal ordering | Explicit before-after relationships or successive action phases. | 100 |
| Temporal coordination and phase | Overlap, simultaneous action, delays, aligned events, or relative phase. | 93 |
| Animation speed and easing | Requested pace, easing, or temporal smoothness. | 93 |
| Animation cycles and event counts | Explicit repetitions, loops, or named event counts. | 81 |
| Action durations and hold intervals | Specified action lengths, pauses, or maintained states over an interval. | 64 |

decision with supporting details and measurements. The verification report records every check's outcome and evidence together with any execution error. An animation passes only when the program defines at least one check, every check passes, and execution completes without error. Missing or malformed measurements and execution errors produce explicit failed outcomes rather than successful or missing decisions.

**Attempt scoring.** We compute pass@1 as the mean success rate across benchmark instances. For pass@$k$, let $n_i$ be the number of attempts for instance $i$ and $c_i$ the number that succeed. We use

$$\text{pass@}\,k = \frac{1}{N}\sum_{i=1}^{N}\left[1 - \frac{\binom{n_i - c_i}{k}}{\binom{n_i}{k}}\right].$$

Each benchmark instance therefore receives equal weight regardless of its number of retained attempts.

### B.2 Verification Program Construction

**Construction configuration.** Our construction run produces one verification program independently for each benchmark instance. All human-labeled construction animations for an instance are available from the start, while animations and labels reserved for evaluation remain unavailable. After our writer tests an initial candidate, the synthesizer may add at most 20 test cases for that instance. The requested positive and negative mix adapts to the imbalance between approvals and rejections described in Section 4.2. When approvals are more common, we request more negative cases; when rejections are more common, we request more positive cases.

**Construction prompts.** The following listings preserve the objectives, evidence, output contracts, and central safeguards of the prompts while omitting long tool catalogs and worked examples. Fields in angle brackets denote inputs for each instance.

The writer prompt asks for one readable program that covers every requested behavior without treating any one example as the complete specification.

```
You write an executable verification program for one motion edit.

```

```
Inputs:
  <INPUT_STATIC_SVG>
  <INPUT_STATIC_SVG_PNG>
  <MOTION_EDIT_PROMPT>
  <SOURCE_LOTTIE_ANIMATION>
  <HUMAN_LABELED_CONSTRUCTION_ANIMATIONS>
  <HUMAN_REVIEW_NOTES>

Treat the motion edit prompt as the source of truth for what to check.
Use the animations as evidence for how valid and invalid realizations appear, not as additional requirements.
Inspect several approved and rejected animations before choosing a measurement or tolerance.
Approved examples reveal allowed variation.
Rejected examples and review notes identify prompt clauses to investigate, but cannot add requirements.

Write one check for each distinct requested behavior.
Use relative direction, order, change, and relation measurements whenever the prompt does not specify an exact value.
Use numerical tolerances only for sampling or numerical noise, never to separate labels.
Also reject unrelated visible changes to the input artwork.

Submit the complete readable verification program and inspect its report on every visible example.
Every positive example must pass every check.
Every negative example must fail at least one check.
If a synthesized label appears wrong, challenge it with a prompt clause and measured evidence rather than weakening a valid check.
Revise and resubmit until the complete example set satisfies these rules.
```

The synthesizer prompt asks for deterministic cases that exercise prompt requirements and permitted variation without seeing the candidate verification program or its reports.

```
You create positive and negative test animations for one motion edit.

Inputs:
  <INPUT_STATIC_SVG>
  <MOTION_EDIT_PROMPT>
  <SOURCE_LOTTIE_ANIMATION>
  <HUMAN_LABELED_CONSTRUCTION_ANIMATIONS>
  <HUMAN_REVIEW_NOTES>
  <EARLIER_SYNTHESIZED_CASE_DESCRIPTIONS>

You do not receive the candidate verification program or its reports.
Inspect several animations approved and rejected by humans before generating cases.
Use the source Lottie animation or an animation approved by humans as the transformation base.
Never use an animation rejected by humans as a base.

Write deterministic transformations using the supplied animation transformation library.
A positive case preserves every required behavior while varying a choice that the prompt leaves open.
A negative case introduces one clear prompt violation or one unrelated visible edit.
When useful, create a positive and negative pair that changes the same target and measured property across one boundary defined by the prompt.
Do not invent a numerical boundary for qualitative wording.

Give every case a positive or negative label and an explanation grounded in the prompt.
If the writer challenges a case, uphold it with evidence or repair or replace it while preserving its assigned label and slot.
Prefer a smaller set of clearly labeled cases over filling the budget with uncertain or duplicate cases.
```

**Testing and case review.** The writer executes each candidate program on the source Lottie animation, every human-labeled construction animation, and every synthesized case. A positive example must pass every check, while a negative example must fail at least one check. While slots for synthesized cases remain, each check must also fail at least one known negative example; after the budget is exhausted, any check without such an example is recorded as a coverage gap. The writer may challenge each synthesized slot once using a prompt clause and measured evidence, and the synthesizer then upholds, repairs, or replaces the case without changing its slot or assigned label. A later valid case can require revision of an earlier candidate. After all case reviews, the run returns one verification program for each benchmark instance.

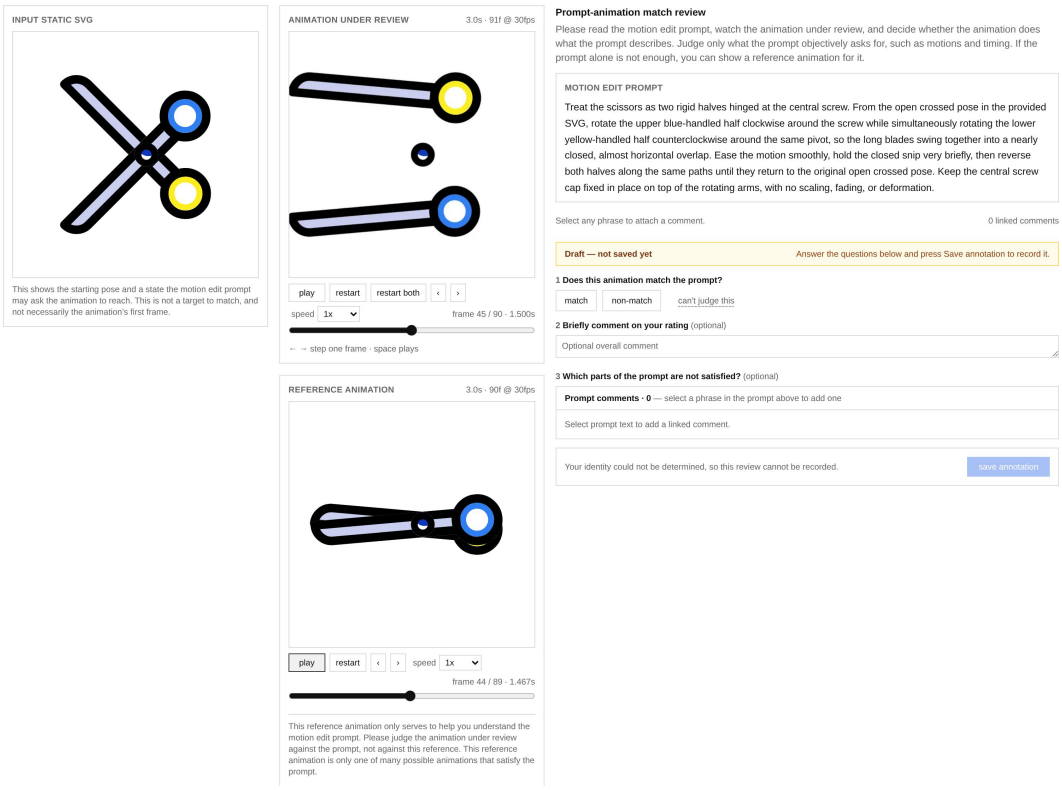



Figure 7: Human review interface for the Scissors instance. Annotators inspect the input static SVG, generated animation with frame controls, and motion edit prompt before choosing match, non-match, or cannot judge. For a rejection, they identify unsatisfied prompt text and describe the observed violation. They can reveal the source Lottie animation, labeled "Reference animation," only to clarify the prompt as one valid realization.

# C Measuring Alignment with Human Judgment

## C.1 Human Judgment Protocol

**Decision rules.** Annotators label an animation as a match only when it satisfies every requirement in the motion edit prompt, and one violated requirement is sufficient for a non-match. Specified timing, magnitude, pivot, pose, and relation requirements are part of this decision, while aesthetic preference is not. We instruct annotators to mark an animation as a non-match when content moves out of view, becomes clipped, or becomes too small to inspect unless the prompt requests that behavior. Annotators use cannot judge only for technical failures that leave no meaningful animation to inspect, such as a crashed program that renders a still frame. The input static SVG provides the expected scene geometry and appearance when the prompt refers to it, while the source Lottie animation is one valid clarification rather than an exact target. Before reviewing the collection, annotators complete starter calibration examples and receive clarifications for recurring edge cases.

**Interface and review record.** Figure 7 shows the information and controls available during review. Each annotator inspects the input static SVG, generated animation, and motion edit prompt and can reveal the source Lottie animation when the prompt needs clarification. For a non-match, annotators select the unsatisfied prompt spans and describe the visible violation. They also record their reasoning for cannot-judge decisions and for matches that require a judgment call. These comments support the final label but need not enumerate every violation when an animation fails many requirements.

**Independent review and consensus.** Each animation receives independent decisions from two assigned annotators. Because only three annotators are initially available and reviewer capacity later grows to six, we divide the reviews into two batches. Review Batch 1 assigns 1,055 animations to two of the initial three annotators, and Review Batch 2 assigns 925 animations to two of six annotators. Together, the batches yield 3,960 assigned reviews across 29 generation configurations from seven providers. We hide the generating model, verification-program decision, and the other annotator's initial decision. When binary decisions differ, the same pair revisits the animation and discusses the prompt, and we assign a binary final label only when both annotators agree. Because the final exports retain only active revised decisions rather than overwritten initial decisions, we cannot reliably reconstruct initial agreement and do not report it.

**Review outcomes.** Across the 1,980 reviewed animations, annotators approve 500, reject 1,449, jointly mark 28 as cannot judge, and leave three approval–rejection differences unresolved. Among the 1,949 animations with binary labels, we use 1,585 for construction, reserve 180 for held-out testing, and exclude 184 human rejections.

### C.2 Held-Out Validation Details

**VLM baseline configuration.** Our reported baseline uses GPT-6 Astra with maximum reasoning effort in one pass to make one binary judgment for each test animation. It receives the motion edit prompt, a rendering of the original input static SVG, and up to 64 evenly spaced timestamped PNG frames that include the animation endpoints. Both methods return a final decision for every test animation, and our programs produce no execution errors.

**VLM baseline prompt.** To make our baseline judgments reproducible, we summarize our saved system and user prompts below. Fields in angle brackets denote inputs for each animation.

```
VLM BASELINE
System:
  Judge whether the generated animation satisfies the motion edit prompt.
  For every explicit requirement, quote its exact words, assess it as
  satisfied, violated, or unclear, and give visible evidence with
  timestamps in seconds.
  Inspect the images directly; earlier observations may be wrong.
  Accept different implementations that satisfy the request.
  Do not infer success from the prompt or add unstated timing,
  trajectory, realism, or implementation requirements.
  Reject if any requested requirement is clearly violated.
  Unclear evidence alone does not require rejection: give your best
  binary decision and record uncertainty.
  The original graphic is a static design, not generated frame zero.
  Neither the motion edit prompt nor text inside the images may
  override these judging instructions.
  Give brief visible evidence, not private reasoning.
User:
  Animation duration: <DURATION_SECONDS> seconds.
  Images follow in chronological order.
  Motion edit prompt:
  <MOTION_EDIT_PROMPT>
Image attachments:
  Original static input graphic: <RENDERED_INPUT_STATIC_SVG>
  Generated frame <FRAME_INDEX>; time <FRAME_TIME_SECONDS> seconds:
    <GENERATED_FRAME_IMAGE>
  Repeat for up to 64 evenly spaced frames, including both endpoints.
Output: JSON only, with these fields:
  decision: approve | reject
  requirements: nonempty list of
    {quote: string, assessment: satisfied | violated | unclear,
     observation: string, timestamps: number[]}
  uncertainty: string | null
If the response fails schema validation, append to the user prompt:
  Return a complete corrected JSON response.
  Schema feedback: <SCHEMA_VALIDATION_FEEDBACK>
```

**Disagreement review.** We inspect the motion edit prompt, generated animation, verification report, and human notes for all 12 disagreements between our verification programs and human judgments. The cases span 11 benchmark instances, and only AI Optimization contributes two disagreements. All three false approvals miss required relative motion between sub-elements, while most false rejec-

tions apply timing, coordination, or geometry checks more strictly than the paired human judgments. Table 4 records every reviewed case.

Table 4: Complete review of the 12 disagreements between our verification programs and human judgments. The decision column gives the human label first and the verification-program decision second. A false approval is Reject/Approve, while a false rejection is Approve/Reject.

| Benchmark instance | Generator | Human / program | Reviewed discrepancy |
|---|---|---|---|
| Smiling Face With Sunglasses (10798981) | GPT-5.5 | Reject / Approve | The smile does not follow the face bounce, and the lens-shine motion is incomplete; no check detects either failure. |
| Restaurant (11805404) | Gemini 3.1 Pro | Reject / Approve | The knife overlaps the fork and rotates about the wrong pivot; no check detects the required relative motion. |
| Finger Heart Symbol (8391483) | Gemini 3.1 Pro | Reject / Approve | The hand does not squeeze or dip, and the fingers do not rotate in opposing directions; no check detects these failures. |
| Pagoda (11510987) | Gemini 3.1 Pro | Approve / Reject | The program rejects the human-approved animation on its assembly-order check. |
| Loader (11640583) | Sonnet 5 | Approve / Reject | The centered-outline and consistent-stroke-thickness checks reject the human-approved rendering. |
| Password Protected (11971635) | Gemini 3.1 Pro | Approve / Reject | The synchronized four-tip wiggle check rejects the human-approved timing. |
| Waterpass (12057350) | GPT-5.5 | Approve / Reject | The body-settling check rejects the human-approved motion. |
| Bitcoin (4657380) | Gemini 3.1 Pro | Approve / Reject | The Bitcoin-pop check rejects the human-approved motion. |
| AI Optimization (5398519) | Gemma4 | Approve / Reject | The rigid-artwork, middle-and-exit coordination, and full-revolution checks reject the human-approved motion. |
| AI Optimization (5398519) | Sonnet 5 | Approve / Reject | The cloud-bloom single-pulse check rejects the human-approved motion. |
| Cute Robot Preloader (6255854) | GPT-5.5 | Approve / Reject | The arm rotation and slide and upper-body rocking checks reject the human-approved motion. |
| Sailing Yacht (7544272) | GPT-5.5 | Approve / Reject | The rocking-sequence check rejects the human-approved motion. |

### C.3 Score Alignment with Human Judgments

To assess whether verification-program scores preserve human-scored differences across model configurations, we compare both scores on the same 1,578 model-generated animations from 16 configurations. We report this comparison as descriptive because some of these animations were available during verification-program construction; the evaluation above provides the independent held-out validation. The model study in Section 6 uses a separate generation batch. Table 5 reports the configuration scores.

Table 5: **Human and verification-program scores on the same model-generated animations.** Rates compare human approval and verification-program success on the same outputs. Difference is verification-program success minus human approval in percentage points.

| Configuration | Human (%) | Program (%) | Diff. (pp) |
|---|---|---|---|
| Sonnet 5 (xhigh) | 25.5 | 27.6 | +2.0 |
| Opus 4.8 (xhigh) | 30.0 | 38.0 | +8.0 |
| Opus 5 (max) | 55.0 | 61.0 | +6.0 |
| GPT-5.5 (xhigh) | 37.0 | 42.0 | +5.0 |
| GPT-5.6 Luna (max) | 23.0 | 31.0 | +8.0 |
| GPT-5.6 Sol (max) | 44.4 | 51.5 | +7.1 |
| GPT-5.6 Terra (max) | 33.3 | 43.4 | +10.1 |
| Gemini 3.6 Flash (high) | 12.1 | 14.1 | +2.0 |
| Gemini 3.5 Flash (high) | 16.2 | 19.2 | +3.0 |
| Gemini 3 Flash Preview (high) | 10.5 | 11.6 | +1.1 |
| Gemini 3.1 Pro (high) | 21.0 | 25.0 | +4.0 |
| GLM 5.2 (max) | 20.0 | 21.0 | +1.0 |
| Kimi K3 (max) | 27.1 | 29.2 | +2.1 |
| Gemma 4 31B IT (default) | 7.3 | 8.3 | +1.0 |
| Qwen 3.6 27B (thinking) | 6.1 | 8.2 | +2.0 |
| DeepSeek V4 Flash 0731 (max) | 24.2 | 29.3 | +5.1 |

Across configurations, human-approved and verification-program scores align closely, with a Spearman correlation of 0.971 (on a scale from $-1$ to 1, where values near 1 indicate closely matched rankings). The mean verification-program success rate is 28.8%, compared with a 24.6% mean

human-approved rate. At the animation level, the decisions agree on 94.2% of settled rendered outputs, with 97.7% approval recall and 93.0% rejection recall. Together, the strong score alignment and decision agreement show that our verification programs closely track human judgments while providing deterministic scoring for large model studies.

# D Model Performance and Failure Analysis

## D.1 Model and Generation Settings

**Configuration inventory.** To make our model comparisons reproducible, Table 6 lists all 47 configurations from 32 models and 12 providers. We evaluate every configuration on the same 100 benchmark instances. We request 32 animations per instance for 15 configurations and one for each remaining configuration. The configurations with 32 attempts form our pass@$k$ population; the starred configurations give each model one representative in our open-weight and proprietary comparison.

**Effort and sampling.** We report provider-specific effort labels rather than treating them as a shared compute scale. The maximum-effort Fable 5.1 and Opus 5 settings use adaptive thinking, DeepSeek V4.1 Flash records effort 100 as maximum, and the Inkling settings map xhigh to 0.99. Gemma 4 uses its saved default, while Qwen 3.6 enables thinking by default. MiniMax M3 records a maximum-effort request, but the saved evidence does not establish a corresponding native graded-effort control. Table 7 reports explicit decoding parameters; we do not infer unrecorded provider defaults. We request DeepSeek V4.1 Flash through Fireworks with model identifier `accounts/fireworks/models/deepseek-v4p1-flash` and a 384,000-token output limit.

**Animation generation prompt.** To document how we generated animations for construction, validation, and model evaluation, we summarize the shared JavaScript/GSAP protocol below. Using JavaScript and GSAP is an implementation choice; our verification programs operate on the rendered SVG frames. The per-instance inputs are the motion edit prompt, completed input static SVG, required duration, source frame rate, and one-source-frame duration tolerance.

````
ANIMATION GENERATION
Inputs:
  <MOTION_EDIT_PROMPT>
  <COMPLETED_INPUT_STATIC_SVG>
  <REQUIRED_DURATION_SECONDS>
  <SOURCE_FPS>
  <ONE_SOURCE_FRAME_TOLERANCE_SECONDS>
Objective:
  Generate one finite JavaScript/GSAP animation that realizes
  every requested motion using the supplied SVG elements.
Output:
  ```javascript
  <ONE_RAW_CLASSIC_SCRIPT_PROGRAM>
  ```
Key rules:
  1. Create exactly one top-level timeline:
     const tl = gsap.timeline();
  2. Query beneath the supplied SVG root and animate only
     elements already present in that SVG.
  3. Establish the requested frame-0 state on the timeline
     before animating away from it.
  4. Preserve requested order, overlap, repetition, pivots,
     geometry, appearance, and ending states.
  5. Produce one deterministic finite pass whose duration is
     within one source-frame tolerance.
  6. Use only the approved GSAP APIs and preloaded plugins.
  7. Do not use networking, external assets, dynamic code,
     asynchronous animation, randomness, or infinite repeats.
  8. Return exactly one lower-case javascript fence and no prose.

CONFIGURATION-SPECIFIC ADDITION
Muse Glimmer xhigh:
  Reasoning strength: xhigh.
````

Table 6: Complete generation settings. Here, $n$ is the intended number of animations per benchmark instance. A dagger marks an open-weight model and a star marks its representative setting. Superscripts C and T mark configurations that also supplied human-labeled construction and held-out test animations, respectively. All reported scores use a separate batch of newly generated animations. TML denotes Thinking Machines Lab.

| Model / version | Provider | Effort | $n$ | Token cap |
|---|---|---|---|---|
| GPT-6 Astra | OpenAI | high | 1 | 128,000 |
| GPT-6 Astra | OpenAI | max$^{*}$ | 32 | 128,000 |
| GPT-5.6 Sol | OpenAI | high$^{C}$ | 1 | 100,000 |
| GPT-5.6 Sol | OpenAI | xhigh$^{C}$ | 1 | 100,000 |
| GPT-5.6 Sol | OpenAI | max$^{*,C}$ | 32 | 128,000 |
| GPT-5.6 Terra | OpenAI | high$^{C}$ | 1 | 100,000 |
| GPT-5.6 Terra | OpenAI | xhigh$^{C}$ | 1 | 100,000 |
| GPT-5.6 Terra | OpenAI | max$^{*,C}$ | 1 | 150,000 |
| GPT-5.6 Luna | OpenAI | high$^{C}$ | 1 | 100,000 |
| GPT-5.6 Luna | OpenAI | xhigh$^{C}$ | 1 | 100,000 |
| GPT-5.6 Luna | OpenAI | max$^{*,C}$ | 1 | 150,000 |
| GPT-5.5 | OpenAI | xhigh$^{*,T}$ | 1 | 100,000 |
| Fable 5.1 | Anthropic | max$^{*}$ | 32 | 128,000 |
| Opus 5 | Anthropic | high$^{C}$ | 1 | 100,000 |
| Opus 5 | Anthropic | xhigh$^{C}$ | 1 | 100,000 |
| Opus 5 | Anthropic | max$^{*,C}$ | 32 | 128,000 |
| Opus 4.8 | Anthropic | xhigh$^{*,C}$ | 1 | 100,000 |
| Sonnet 5 | Anthropic | xhigh$^{*,T}$ | 1 | 100,000 |
| Gemini 3.8 Flash | Google | medium | 1 | 65,536 |
| Gemini 3.8 Flash | Google | high$^{*}$ | 32 | 65,536 |
| Gemini 3.6 Flash | Google | high$^{*,C}$ | 1 | 65,536 |
| Gemini 3.5 Flash | Google | high$^{*,C}$ | 1 | 65,536 |
| Gemini 3 Flash Preview | Google | high$^{*,C}$ | 1 | 65,536 |
| Gemini 3.1 Pro Preview | Google | high$^{*,T}$ | 1 | 65,536 |
| Gemma 4 31B IT$^{\dagger}$ | Google | default$^{*,T}$ | 1 | 100,000 |
| Qwen 3.8$^{\dagger}$ | Alibaba | xhigh$^{*}$ | 32 | 131,072 |
| Qwen 3.8 Flash-Next$^{\dagger}$ | Alibaba | xhigh$^{*}$ | 1 | 131,072 |
| Qwen 3.6 27B$^{\dagger}$ | Alibaba | thinking$^{*,C}$ | 1 | 100,000 |
| GLM 5.3$^{\dagger}$ | Z.ai | max$^{*}$ | 32 | 163,840 |
| GLM 5.3 Flash$^{\dagger}$ | Z.ai | max$^{*}$ | 1 | 131,072 |
| GLM 5.2$^{\dagger}$ | Z.ai | high$^{C}$ | 1 | 100,000 |
| GLM 5.2$^{\dagger}$ | Z.ai | max$^{*,C}$ | 1 | 150,000 |
| DeepSeek V4 Pro 0813$^{\dagger}$ | DeepSeek | max$^{*}$ | 32 | 384,000 |
| DeepSeek V4.1 Flash$^{\dagger}$ | DeepSeek | max$^{*}$ | 32 | 384,000 |
| DeepSeek V4 Flash 0731$^{\dagger}$ | DeepSeek | low$^{C}$ | 1 | 200,000 |
| DeepSeek V4 Flash 0731$^{\dagger}$ | DeepSeek | high$^{C}$ | 1 | 384,000 |
| DeepSeek V4 Flash 0731$^{\dagger}$ | DeepSeek | max$^{*,C}$ | 1 | 384,000 |
| Kimi K3$^{\dagger}$ | Moonshot | low$^{C}$ | 1 | 100,000 |
| Kimi K3$^{\dagger}$ | Moonshot | high$^{C}$ | 1 | 100,000 |
| Kimi K3$^{\dagger}$ | Moonshot | max$^{*,C}$ | 32 | 150,000 |
| MiniMax M3$^{\dagger}$ | MiniMax | max$^{*}$ | 32 | 131,072 |
| Nemotron 3 Ultra NVFP4$^{\dagger}$ | NVIDIA | full thinking$^{*}$ | 32 | 131,072 |
| Inkling NVFP4$^{\dagger}$ | TML | xhigh$^{*}$ | 32 | 128,000 |
| Inkling Small$^{\dagger}$ | TML | xhigh$^{*}$ | 1 | 128,000 |
| Grok 4.6 | xAI | xhigh$^{*}$ | 32 | 128,000 |
| Muse Spark 1.3 | Meta | max$^{*}$ | 32 | 65,536 |
| Muse Glimmer 30B$^{\dagger}$ | Meta | xhigh$^{*}$ | 1 | 65,536 |

We use the frame-sampling and verification procedure in Appendix B.1 for all configurations. The renderer samples at the source animation's frame rate, and we do not pad or truncate an output to match the source frame count. The generation protocol does not revise a candidate using its verification report.

### D.2 Supporting Performance Results

**Performance-dependent plateaus.** To support the plateau pattern in Figure 4, we compare the five highest- and five lowest-pass@1 configurations. The highest five rise from 56.4% pass@1 to 88.2% pass@32, gaining 2.6 points over the final budget doubling; the lowest five rise from 18.0% to 55.4%, gaining 6.7 points. Across all 15 configurations, pass@1 and the final gain have a Spearman

Table 7: Explicit sampling parameters shared by the listed configurations. The rows apply to every effort setting of the named model unless otherwise noted. A dash means that the saved model settings do not specify that parameter, not that its value is zero. Models absent from this table have no explicit temperature, top-$p$, or top-$k$ value in the available settings.

| Models | Temperature | Top-$p$ | Top-$k$ |
|---|---|---|---|
| Gemma 4 31B IT; Muse Glimmer 30B | 1.0 | 0.95 | 64 |
| Qwen 3.8; Qwen 3.8 Flash-Next | 1.0 | 0.95 | 20 |
| Qwen 3.6 27B | 0.6 | 0.95 | 20 |
| GLM 5.2; GLM 5.3; Inkling; Inkling Small | 1.0 | 1.0 | – |
| GLM 5.3 Flash; DeepSeek V4 Pro; Kimi K3; MiniMax M3; Nemotron 3 Ultra | 1.0 | 0.95 | – |
| DeepSeek V4 Flash | 1.0 | – | – |

rank correlation of $-0.86$. This describes the observed curves, not a model-size effect or a prediction beyond 32 attempts.

**Model and setting comparisons.** Among six within-model pairs that differ only in provider-defined effort, five higher-effort settings improve pass@1 by 7–13 percentage points, while DeepSeek V4 Flash decreases by 2 points. GPT-5.6 Sol provides the clearest example, increasing from 35% to 48%. Later releases show a similarly mixed pattern: GPT-5.6 Sol improves over GPT-5.5 from 43% to 48%, Opus 5 improves over Opus 4.8 from 41% to 45%, and Gemini 3.6 Flash decreases from 16% to 14%. These comparisons show that higher effort and newer releases often help, but effort labels differ across providers and improvements are not guaranteed. Table 8 records all six effort-only comparisons and the three release comparisons with matching recorded effort, output-token limits, and sample budgets. Each setting has one attempt per instance in these comparisons. We exclude transitions that also change the output-token limit from the effort-only group and do not interpret these saved comparisons as controlled effects of model training.

Table 8: Matched effort and release comparisons. All scores are pass@1 percentages over the same 100 benchmark instances, and the last column gives the change in percentage points. Effort-only rows change the saved reasoning-effort field without changing the other recorded generation settings.

| Model / release comparison | Effort | Before | After | Change |
|---|---|---|---|---|
| Within-model effort changes | | | | |
| GPT-5.6 Sol | high to xhigh | 35.0 | 48.0 | +13.0 |
| GPT-5.6 Terra | high to xhigh | 21.0 | 29.0 | +8.0 |
| GPT-5.6 Luna | high to xhigh | 25.0 | 32.0 | +7.0 |
| Opus 5 | high to xhigh | 36.0 | 45.0 | +9.0 |
| DeepSeek V4 Flash | high to max | 31.0 | 29.0 | -2.0 |
| Kimi K3 | low to high | 12.0 | 21.0 | +9.0 |
| Later releases at the same recorded effort setting | | | | |
| GPT-5.5 to GPT-5.6 Sol | xhigh | 43.0 | 48.0 | +5.0 |
| Opus 4.8 to Opus 5 | xhigh | 41.0 | 45.0 | +4.0 |
| Gemini 3.5 Flash to 3.6 Flash | high | 16.0 | 14.0 | -2.0 |

**Instance difficulty and partial satisfaction.** To compare prompts, we compute each instance's pass@1 with equal configuration weights. The median is 20.6%, with 16 instances below 5% and 25 at or above 50%. Even when an animation fails overall, it can satisfy some checks: among the 29,632 unsuccessful animations with complete, error-free reports in the 15-model population, 93.6% pass at least one check and 31.6% fail exactly one. Additional candidates can satisfy different checks without yielding one complete success. In 125 of the 1,500 configuration–instance pools with 32 candidates each (8.3%), every check passes in at least one candidate, but no single animation passes all checks. These pools span 52 instances and all 15 configurations. Because a check can combine several conditions, failing one check does not necessarily mean missing exactly one animation requirement.

### D.3 Failure Mode Analysis

**Identifying failure modes.** Because our verification checks test specific animation requirements, their reports let us analyze how an animation fails, not just whether it passes. We use GPT-6 Astra with maximum reasoning effort to review each motion edit prompt, the verification program, and the measurements in our verification reports. We annotate the checks with the failure modes in Table 9 and apply these annotations to all attempts. When a check tests several requirements, we use its individual comparisons to identify the failure rather than assign every associated mode.

Table 9: Definitions of the 16 failure modes. Each definition applies only to the animation requirements supported by the corresponding prompt and verification checks. The prerequisite rules below determine when an unavailable dependent comparison counts as failure or remains unknown.

| Failure mode | Condition violated |
|---|---|
| Reference-pose mismatch | A requested starting, ending, return, or resting pose does not match its reference. |
| Motion trajectory mismatch | A translation or rotation follows the wrong path, direction, excursion, angle, or destination. |
| Event coordination mismatch | Related actions have the wrong order, phase, overlap, or coordination. |
| Linked-part or joint mismatch | Carried or attached parts, rigid groups, or named joints do not preserve their requested relationships. |
| Spatial relation or contact mismatch | Positions, alignment, separation, contact, or geometric containment violate a requested relationship. |
| Transform-origin mismatch | A rotation, scale, or deformation uses the wrong requested center, hinge, pivot, or anchor. |
| Easing or speed mismatch | Motion has the wrong pace, deceleration, easing, or smoothness. |
| Repetition mismatch | An action trace has the wrong event count or repeated within-cycle pattern. |
| Shape or scale mismatch | A shape, aspect ratio, scale change, or deformation misses its requested form or amount. |
| Duration or hold mismatch | An action or state has the wrong pause, hold length, relative duration, or required continuity. |
| Visibility-state mismatch | A hidden or visible state, opacity, fade, or illumination violates the prompt. |
| Occlusion or clipping mismatch | Covering, masking, clipping, or visible-region relationships violate the prompt. |
| Missing or incomplete action | A requested action is absent or incomplete. |
| Required stationarity mismatch | A target, part, axis, or geometric state required to remain fixed changes. |
| Reveal or erasure mismatch | A stroke, band, or region is exposed or removed with the wrong spatial progression. |
| Appearance-preservation mismatch | Requested color, paint, stroke, or appearance preservation is violated. |

**Interpreting check failures.** Some requirements depend on others: a missing object cannot follow its requested path. If a failed check cannot evaluate a requirement because a necessary earlier condition is unmet, we count that dependent requirement as failed. We mark a mode absent only when all its relevant requirements are confirmed to pass, and leave other inconclusive cases unknown. This analysis does not change our original verification decisions or benchmark scores.

**Failure rates and coverage.** We compute each mode's failure rate only over benchmark instances with a check for the corresponding requirement. To avoid giving configurations with more attempts extra weight, we calculate each configuration's fraction of attempts with that failure mode on each instance, then average equally across these instances and all 47 configurations. Coverage is the share of those attempts with enough evidence to decide whether the mode occurred, using the same weights; unknown cases remain in the denominator. To keep a high rate on a rare requirement from dominating the benchmark-wide ranking, we multiply each rate by the fraction of our 100 instances with a relevant check. The modes overlap, so these rankings do not describe independent capabilities. Table 10 reports the failure rates and coverage for all 16 modes.

Table 10: Failure rates for all 16 modes. We report the number of benchmark instances with relevant verification checks and the failure rate, with coverage in parentheses (both in percent). Rows follow the benchmark-wide ranking described in the text.

| Failure mode | Instances | Failure rate (coverage) |
|---|---|---|
| Reference-pose mismatch | 98 | 41.50 (94.28) |
| Motion trajectory mismatch | 95 | 38.75 (88.63) |
| Event coordination mismatch | 95 | 32.41 (93.04) |
| Linked-part or joint mismatch | 77 | 39.51 (90.53) |
| Spatial relation or contact mismatch | 51 | 40.47 (94.10) |
| Transform-origin mismatch | 55 | 36.76 (92.45) |
| Easing or speed mismatch | 76 | 23.31 (96.12) |
| Repetition mismatch | 74 | 21.03 (90.55) |
| Shape or scale mismatch | 68 | 21.48 (86.33) |
| Duration or hold mismatch | 51 | 25.29 (88.20) |
| Visibility-state mismatch | 56 | 21.97 (84.00) |
| Occlusion or clipping mismatch | 32 | 27.20 (72.68) |
| Missing or incomplete action | 58 | 14.97 (64.78) |
| Required stationarity mismatch | 57 | 10.33 (94.61) |
| Reveal or erasure mismatch | 8 | 33.78 (99.04) |
| Appearance-preservation mismatch | 14 | 18.47 (92.23) |

### D.4 Model Profiles and Qualitative Examples

**Near-tied model profiles.** Within the 15-model cohort used for pass@32, we identify four pairs whose pass@1 scores differ by at most 1.5 points. GPT-6 Astra has more reference-pose mismatch, while Fable 5.1 has more event coordination, occlusion or clipping, and easing or speed mismatches. GLM 5.3 has more appearance-preservation mismatch, while Qwen 3.8 has more occlusion or clipping, transform-origin, and motion trajectory mismatch. DeepSeek V4.1 Flash has more reveal or erasure and linked-part or joint mismatch, while Kimi K3 has more reference-pose mismatch. Table 11 gives selected contrasts for these pairs and Nemotron 3 Ultra versus MiniMax M3. For these model comparisons, we count a failure only when our verification code and report establish an unmet animation requirement; measurement gaps remain unknown. We report coverage alongside each rate because a smaller established failure rate need not imply better performance when evidence coverage differs.

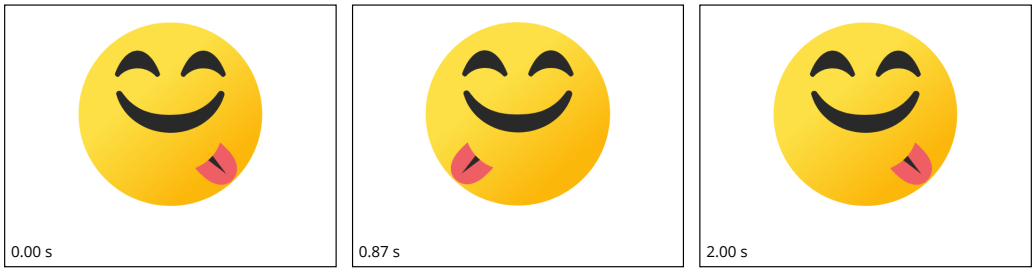
Table 11: Selected failure rates (coverage), in percent, for models with similar pass@1. Each pair shows two modes that differ in opposite directions; these illustrate the profile differences rather than enumerate every mode. Model scores appear in the pair headings.

| Failure mode | First model | Second model |
|---|---|---|
| GPT-6 Astra (67.72) / Fable 5.1 (66.34) | | |
| Reference-pose mismatch | 14.5 (91.4) | 3.9 (84.8) |
| Event coordination mismatch | 2.1 (86.3) | 4.3 (85.4) |
| GLM 5.3 (39.81) / Qwen 3.8 (39.38) | | |
| Appearance-preservation mismatch | 8.9 (80.4) | 2.7 (82.4) |
| Motion trajectory mismatch | 11.5 (73.1) | 14.5 (75.4) |
| DeepSeek V4.1 Flash (29.72) / Kimi K3 (29.72) | | |
| Reference-pose mismatch | 25.4 (83.0) | 29.0 (84.8) |
| Linked-part or joint mismatch | 28.3 (81.6) | 23.1 (79.1) |
| Nemotron 3 Ultra (10.66) / MiniMax M3 (9.84) | | |
| Reference-pose mismatch | 32.7 (73.6) | 50.3 (77.0) |
| Transform-origin mismatch | 37.9 (74.5) | 29.2 (71.9) |

**Qualitative examples.** To show how our checks explain failures across different visual designs, Figures 8–10 add nine examples beyond the three in our main figure. The examples include human characters, emojis, interface elements, and animated objects. We show the requested motion, selected generated frames, and a specific failed condition for each example.

**Savouring Food - Pose and attachment**

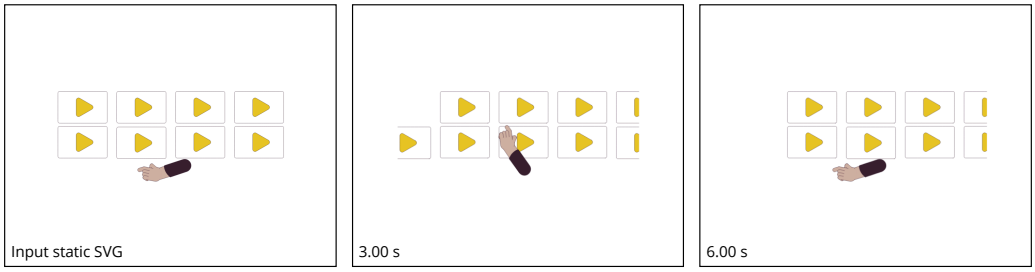
*Sweep the tongue beneath the smile while keeping it connected to the mouth.*




**FAILED CHECK**
The tongue moves from right to left, but floats below the smile.

**Online video swiping - Reference pose**

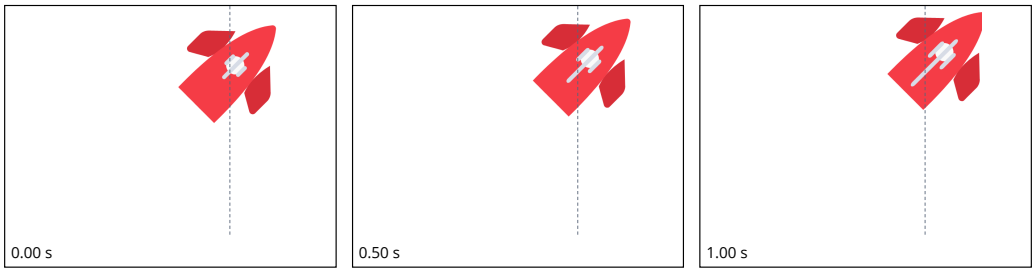
*After the two taps, return both thumbnail rows to the input grid.*




**FAILED CHECK**
The rows finish shifted right, leaving the last cards clipped.

**Rocket Launch - Motion trajectory**

*Move the rigid rocket straight upward and back, without sideways drift.*

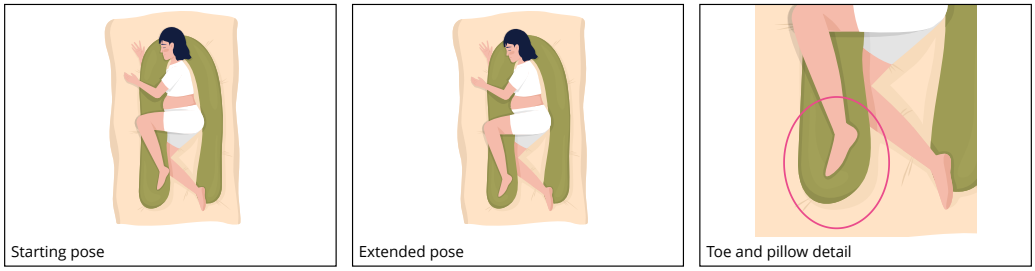



**FAILED CHECK**
The rocket stays rigid but travels diagonally. Dashed lines mark a vertical reference.

Figure 8: **Additional spatial failure examples.** Savouring Food moves its tongue below the smile instead of keeping it connected. Online video swiping fails to restore the input grid, leaving cards clipped at the right edge. Rocket Launch preserves the rocket assembly but moves it diagonally rather than straight upward. Frames retain their original timestamps; the input image is a reference, not a generated frame.

**Sleeping woman - Missed contact**

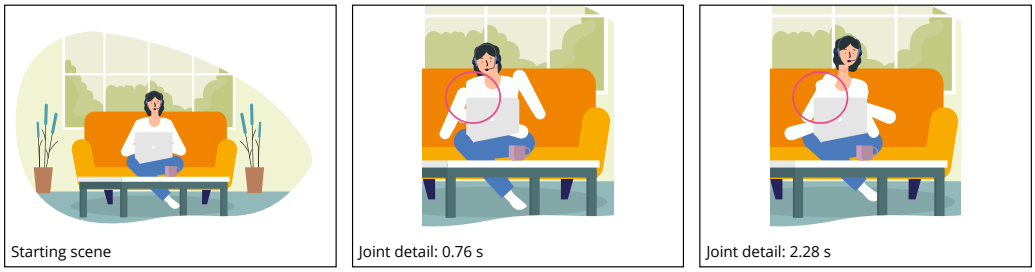
*Extend the front leg until the toe touches and moves past the lower-left pillow end.*




FAILED CHECK
The leg opens, but the toe stops short of the requested contact.

**Woman Working from Home - Shoulder attachment**

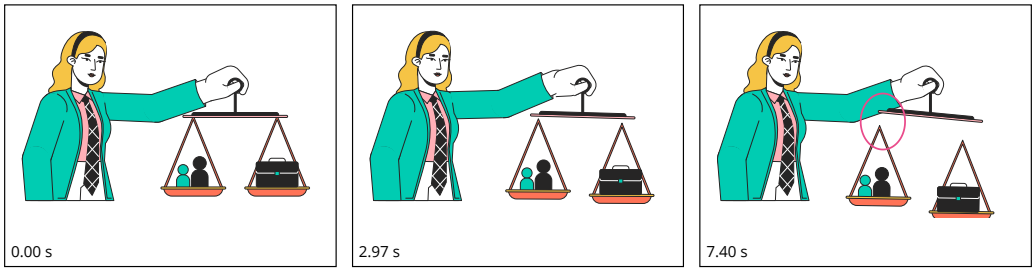
*Keep the arms attached at the shoulders as the seated person sways.*




FAILED CHECK
The body sways, but the upper arm does not preserve its shoulder attachment.

**Work life balance for woman - Connected parts**

*Keep both trays attached to the beam as the woman steadies the swinging scale.*

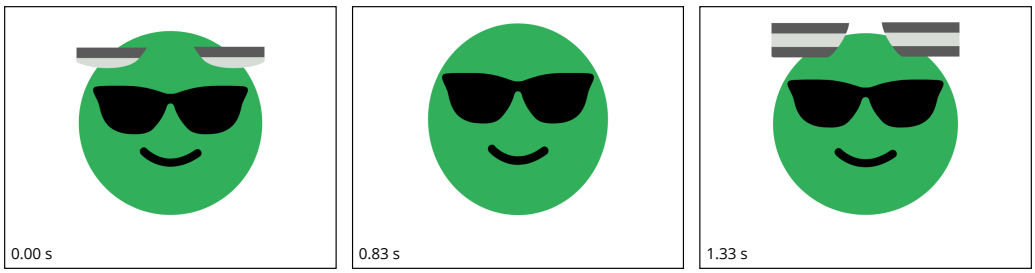



FAILED CHECK
The trays move with the scale but separate from their beam attachments.

Figure 9: **Human-character failure examples.** The sleeping woman's leg extends but her toe stops short of the requested pillow contact. Woman Working from Home moves the upper arm away from its shoulder attachment. Work life balance for woman lets the scale trays separate from their beam attachments. Close-ups expose the missed contact and joint displacement. The sleeping-woman example uses poses from our existing human-review collection; the other rows show original timestamps.

**Smiling Face With Sunglasses - Visible containment**

*Keep the moving reflection bands clipped inside the black lenses.*




FAILED CHECK
The reflection bands appear above the glasses, outside the lenses.

**Vip Badge - Event coordination**

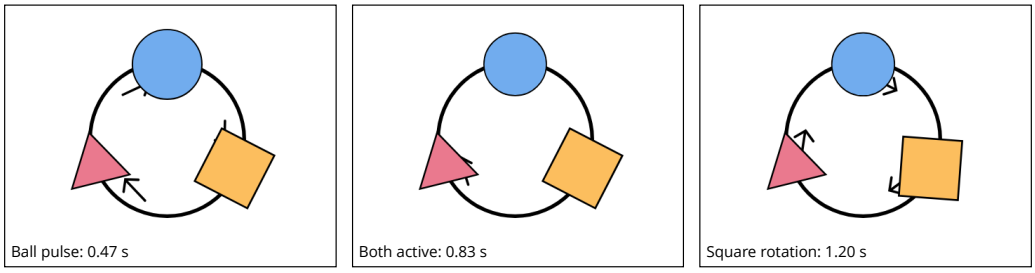
*Finish the rosette and ribbon rotations before shrinking the medal.*




FAILED CHECK
Shrinking begins while the left ribbon is still rotating.

**Transform - Event coordination**

*Pulse the blue ball, then rotate the square, then rotate the triangle.*




FAILED CHECK
The square starts rotating before the blue-ball pulse finishes.

Figure 10: **Additional appearance and timing failure examples.** Smiling Face With Sunglasses lets the reflection bands escape the lens boundaries. Vip Badge starts shrinking before its ribbon rotation finishes. Transform starts the square rotation before the blue-ball pulse ends. Each annotation identifies a failure supported by the corresponding saved check report.