Self-Consistency for LLM-Based Motion Trajectory Generation and Verification

Jiaju Ma
Stanford University

R. Kenny Jones
Stanford University

Jiajun Wu
Stanford University

Maneesh Agrawala
Stanford University

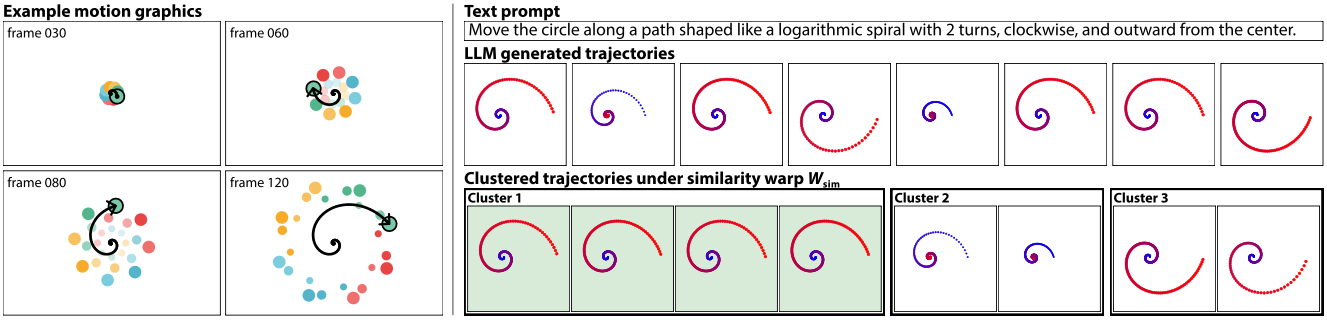


Figure 1. Complex motion graphics animations are often composed of trajectories in the form of geometric shapes (left). While LLMs can generate motion graphics animations from a prompt describing the shape of an object's trajectory, the resulting animation does not always follow the prompt specification (right, motions move from blue to red). We present a self-consistency method that enables more accurate LLM-based trajectory generation without supervision and show that it can be used for trajectory verification. We ask the LLM to generate multiple trajectory samples, cluster the samples using a hierarchy of geometric transformation groups, and choose the largest cluster as the most self-consistent set. We choose the centroid of the largest cluster as the most self-consistent generation, and verify a new trajectory by checking whether it can be added to the largest cluster (i.e. its distance to this centroid falls within a threshold τ).

Abstract

We study visual concept discovery for motion trajectories, where prompts often describe mid-level shape concepts such as spirals, figure-8s, and parabolas. We represent these concepts as shape families: a prototype trajectory paired with a geometric transformation group that captures allowable variation. To recover a shape family we propose an adaption of self-consistency to the visual domain, by clustering samples under warp-invariant transformation groups. Our approach yields an unsupervised, training-free method that supports both improved trajectory generation and concept-based verification. On a benchmark of motion-trajectory prompts, our approach improves generation accuracy by 4–6% and verification precision by 11% over strong baselines.

1. Introduction

Verification is a critical component of generative AI workflows for visual content creation. A user specifies a desired output in natural language, a generative model produces a visual result, and a verifier checks whether the result matches the specification. When a mismatch is detected, the system can either alert the user or use the verification signal to automatically refine the generation. In many visual domains, this verification problem reduces to *concept matching*: determining whether the concepts described in the prompt are present in the generated output.

We study this problem in the domain of motion graphics, where prompts often describe trajectories using shape concepts such as spirals, figure-8s, and parabolas. These concepts are naturally interpretable, but they are also underspecified: a prompt such as "move the circle in a spiral path" does not refer to one exact trajectory, but to a family of trajectories. As a result, direct output matching is not suitable for trajectory verification or for aggregating multiple candidate generations.

To address this, we represent trajectory concepts as *shape families*, defined by a prototype trajectory together with a geometric transformation group that captures the allowable variations of that shape. This representation is compact, structured, and interpretable: the prototype provides a canonical instance of the concept, while the transformation group specifies which geometric changes preserve its identity. Shape families therefore provide a natural representation for mid-level visual concepts in motion trajectories.

To recover shape families that correspond with a user prompt, we propose an adaptation of *self-consistency* to the visual domain. Given a prompt, we first sample diverse motion trajectories from an LLM. We then cluster these sam-

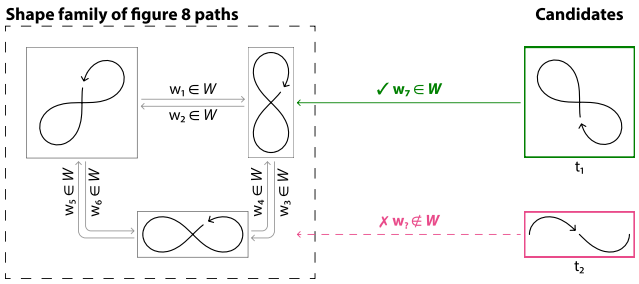


Figure 2. We define a shape family $\mathcal{F}(o, W)$ with a prototype trajectory o and a Lie transformation group W. The shape family is then the set of all trajectories that can be warped into one another $w(o)$ by all warps $w \in W$ (left). In this example, any of the figure 8's in the family can serve as the prototype since they can all be warped into one another. We can check whether a trajectory t_i is a member of the family by checking if there is a warp $w \in W$ that can transform t_i into o (right).

ples using warp-invariant distances defined by candidate transformation groups, and recover the shape family that is most consistent with the sampled outputs. In contrast to standard self-consistency methods, where agreement is often defined by exact matching over discrete answers, our approach defines consistency through membership in a shared geometric shape family.

The recovered shape family supports two downstream tasks. First, it improves generation by selecting a sampled trajectory that is consistent with the inferred shape family. Second, it supports concept-based verification by checking whether a query trajectory belongs to the discovered shape family. Because this procedure relies only on diverse LLM samples and geometric clustering, it is entirely unsupervised and training-free. We evaluate our approach on a benchmark describing shapes common in motion graphics. Our method improves generation accuracy by 4–6% over direct LLM generation, and improves verification precision by 11% over VLM baselines.

2. Preliminaries

Self-Consistency. Self-consistency was introduced as a decoding strategy for improving language model performance on reasoning tasks [27]. The core idea is to sample multiple candidate outputs from a model and then aggregate them by selecting the answer that is most consistent across samples. In its standard form, this consistency is typically defined by exact agreement over discrete outputs such as strings or numerical answers. In this paper, we adapt self-consistency to a visual domain where exact matching is no longer appropriate. Instead of asking whether two outputs are identical, we ask whether they belong to the same underlying geometric concept. Our key idea is to define consistency for motion trajectories through membership in a shared *shape family*.

Motion Trajectories. In motion graphics, a graphical element (e.g., a circle, rectangle, triangle, etc.) is animated by transformations as a function of time. We define the element's *trajectory* as its center point position over time. As shown in Figure 1, motion trajectories commonly found in various motion graphics animations often fall into geometrically well-defined shape families.

Shape Family. Informed by the Erlangen program [16] and Lie transformation groups [8, 15], we define a shape family $\mathcal{F}(o, W)$ as the set of trajectories that can be generated by transforming a prototype trajectory o using any warp w in a transformation group W (Figure 2). That is,

$$\mathcal{F}(o, W) = \{w(o) \mid w \in W\} \tag{1}$$

where the warp w applies transformations such as translation, rotation, scaling, and shear to the prototype o. For example, a text prompt such as *"Move in a figure 8-shaped path"* corresponds to a shape family where o is a figure 8 shape (lemniscate of Bernoulli) at any scale, position, orientation, or reflection. Thus W is the Lie group of similarity transformations with reflections or $\mathrm{Sim}(2)$ [8, 15]. A warp w in this group can apply any combination of translation, rotation, uniform scale, and reflection to o to generate the trajectories of the shape family

Warp-Invariant Distance Metrics. In this work we develop shape families using a set of Lie transformation groups that people commonly consider when describing a family of shapes [21]. As we illustrate in the appendix, we identify a set of Lie transformation groups that form a partially ordered hierarchy. For each transformation group W, we define a corresponding *distance metric* d_W that measures the distance between two trajectories while remaining invariant to warps in W. Specifically, given two trajectories t_1 and t_2, the W-invariant distance is:

$$d_W(t_1, t_2) = \min_{w \in W} \frac{1}{n} \sum_{i=1}^{n} \|w(t_{1,i}) - t_{2,i}\|^2 \tag{2}$$

where $t_{1,i}$ and $t_{2,i}$ denote the i-th points on trajectory t_1 and t_2 respectively. We optimize over all warps $w \in W$ to find the best alignment between the transformed t_1 and t_2. This distance equals zero if and only if $t_2 \in \mathcal{F}(t_1, W)$.

3. Method

Given a prompt describing a desired motion trajectory, our goal is to recover the shape family that best matches the prompt from a set of diverse LLM-generated samples. This recovered shape family is then used both to select a self-consistent generation and to verify whether a query trajectory matches the prompt.

Sampling. We first sample N candidate motion trajectories from an LLM conditioned on the input prompt. We obtain these trajectories through program synthesis: the model

generates motion graphics programs, which are then executed to produce trajectories. To encourage diversity, we prompt the model to generate multiple distinct candidates spanning plausible variations of the described trajectory.

Clustering. Given the sampled trajectories, we compute pairwise distances under the warp-invariant metric d_W for each candidate transformation group W. For a fixed W, this yields a distance matrix over trajectories, which we cluster using DBSCAN with threshold τ. The largest cluster corresponds to the set of trajectories that are most self-consistent under W, and we take its centroid trajectory as the prototype of the corresponding shape family.

Selecting the transformation group. Because the correct transformation group is not known a priori, we estimate it automatically from a hierarchy of candidate groups. We consider two decision criteria: *Majority-Consensus*, which selects the most restrictive group whose largest cluster contains a strict majority of samples, and *Hierarchical-Consistency*, which selects the most restrictive group that preserves the dominant cluster found under less restrictive warps. Intuitively, Majority-Consensus is more conservative and tends to favor precision, while Hierarchical-Consistency better preserves recall. Full details are provided in the appendix.

Membership Check. Once a shape family $\mathcal{F}(o, W)$ has been recovered, we verify a query trajectory t by checking whether it belongs to that family. Concretely, we compute the warp-invariant distance $d_W(o, t)$ between the recovered prototype o and the query trajectory, and accept the trajectory if this distance is below threshold τ. This same recovered family can also be used to select a self-consistent generation by returning a sample from the largest cluster.

4. Results

To evaluate our extension of self-consistency for the domain of motion graphics, we develop a benchmark of prompts associated with ground-truth shape families (Section 4.1). We use this benchmark to validate that our approach provides performance boosts over baseline alternatives for both generation (Section 4.2) and verification (Section 4.3) settings.

4.1. Constructing a Motion Trajectory Benchmark

Comparing the performance of methods that generate or verify motion graphics trajectories requires a dataset that contains text prompts describing motion trajectories and their corresponding ground truth shape families. As no such dataset is available, we synthetically design one using a template-based approach similar to MoVer [18] and CLEVR [13]. We provide the full details of our template-based construction process and visualize pairings in the appendix.

4.1.1. Prompts with ground truth shape families

Our benchmark contains 224 prompts paired with ground-truth shape families. To construct this benchmark, we first identified a set of 35 geometric base shapes that people commonly use to describe the trajectories found in motion graphics animations, as documented by supporting literature [21]. We then manually paired natural language descriptions of these geometric base shapes with a ground truth shape family (Eq. 1) formed by a prototype trajectory and a corresponding transformation group from our hierarchy. We were then able to procedurally create variations of these base shapes by appending the natural language descriptions with additional specifications that induce a change to the corresponding shape family through the transformation group. As an example, a prompt *"Animate the blue circle to move along a path shaped like a circle"* corresponds with a shape family that uses ground truth warps from $W_{\text{sim-ref}}$ (similarity + reflections). Modifying this prompt with additional specifications, *"Animate the blue circle to move along a path shaped like a circle with a radius of 50 px"*, constrains the shape family so that it instead can only use warps from $W_{\text{rgd-ref}}$ (rigid + reflections).

4.1.2. Test set of trajectories for verification

To evaluate performance on the task of verifying motion trajectories, for each prompt we require query trajectories and associated labels for whether they match or mismatch the prompt. To this end, we curated a test set of 2240 trajectories (10 per prompt). For a given (prompt, query trajectory) pair, we used the ground-truth shape family corresponding with the prompt to provide an automatic label through membership checks. To produce this set of query trajectories, we iteratively sampled an LLM (GPT-4.1 & GPT-5) to produce motion trajectories conditioned on the input prompt. We employed a rejection sampling-based approach until we found five true trajectories and five false trajectories for each prompt.

4.2. Motion Trajectory Generation

We use our benchmark to measure the performance on the task of our self-consistency-based motion trajectory generation methods. We consider a baseline generation method that uses an LLM to sample a single motion trajectory for each prompt in our evaluation set (LLM direct). We compare this baseline against our approach that samples an LLM multiple times to identify the most self-consistent trajectories, setting the number of samples to $N = 19$. We evaluate the performance of these methods with an accuracy metric, by checking whether the generated trajectory matches the prompt or mismatches the prompt using a membership check against the corresponding ground-truth shape family.

We report the results of this experiment in Table 1,

Table 1. Motion trajectory generation experiment (see Sec. 4.2). We evaluate the accuracy of LLM-generated motion trajectories under different decoding strategies. Under different decision criteria for selecting a transformation group W, our approach improves upon the alternative of generating a single sample (LLM-Direct).

Method	Decision Criteria	GPT-4.1	GPT-5
LLM-Direct	—	62.1	79.1
Ours	Majority-Consensus	**68.0**	**83.3**
Ours	Hierarchical-Consistency	66.7	82.6
Ours	Oracle	68.5	83.5

where we split the analysis by the LLM used for sampling (GPT-4.1 vs GPT-5). Trajectories generated directly by an LLM without self-consistency (top-row) produce accuracy rates of 62.1% (GPT-4.1) and 79.1% (GPT-5). Our self-consistency methods improve the generation accuracy for a given sampling LLM. In the oracle setting, where self-consistency is computed under the ground-truth transformation group W, we observe improvements of 6.4 and 4.4 percentage points for GPT-4.1 and GPT-5 respectively. In the unsupervised setting, when we use the different decision criteria to estimate W we observe these criteria still provide substantial boosts. Majority-Consensus closely matches the oracle setting, achieving 5.9 and 4.2 percentage point improvements for GPT-4.1 and GPT-5 respectively. Hierarchical-Consistency performs slightly worse, but still significant, with 4.6 and 3.5 percentage point improvements for GPT-4.1 and GPT-5 respectively.

4.3. Motion Trajectory Verification

We can also use our benchmark to measure the performance on the task of motion trajectory verification. In this setting, a method is presented with a prompt and a query motion trajectory, and is tasked with determining whether the trajectory matches the prompt. We consider two VLMs (GPT-4.1 and GPT-5) as baseline verification methods. We can feed a VLM with a prompt and a static rendering of a motion trajectory generated from that prompt, and request that the VLM should indicate whether these two items match or not. We compare these baselines against our self-consistency-based method for verification. We use GPT-5 as the LLM sampling model, and once again set $N = 19$.

We compute precision, recall, and F1 scores by comparing labels predicted by each method against the ground truth labels in our benchmark, and report the results of this experiment in Table 2. We observe that using GPT-4.1 as a verifier is poorly calibrated: the model's predicted prevalence (the rate at which it predicts True) is 90%, far above the true base rate of the data at 50%. This results in a high recall (96.9), at the expense of a low precision (62.0). GPT-5 produces more balanced precision (74.0) and recall (84.7), achieving an F1 score of 79. With oracle access to the ground-truth W, our self-consistency approach for verification is able to achieve better performance (F1 of 85.6).

Table 2. Motion trajectory verification experiment (see Sec. 4.3). We compare our self-consistency-based approach for verifying whether a trajectory matches an input prompt against the alternative of using a VLM (GPT-4.1 or GPT-5).

Method	Decision Criteria	Precision	Recall	F1 Score
GPT-4.1	—	62.0	**96.9**	75.6
GPT-5	—	74.0	84.7	79.0
Ours	Majority-Consensus	**85.8**	66.1	74.6
Ours	Hierarchical-Consistency	80.5	89.0	**84.6**
Ours	Oracle	87.9	83.3	85.6

5. Conclusion

We presented an unsupervised, training-free approach for discovering structured visual concepts in the domain of motion trajectories. Our key idea is to represent trajectory concepts as *shape families*, defined by a prototype trajectory and a geometric transformation group that captures allowable variation. Building on this representation, we adapt self-consistency to the visual domain by clustering LLM-generated trajectories under warp-invariant distances and selecting the shape family best supported by the samples. This recovered representation supports both improved trajectory generation and concept-based verification, and outperforms direct LLM generation and VLM-based verification baselines on our benchmark.

Like prior work on self-consistency [27], our approach cannot recover from all mistakes made by the sampling LLM. Our basic approach is limited to prompts that correspond to a single geometric shape family, and it may fail when the sampled trajectories do not satisfy the assumptions needed for self-consistency to recover the correct family. We discuss an extension that adapts this algorithm for shape families that require multiple prototypes in the appendix.

More broadly, we view this work as a step toward extending self-consistency beyond natural-language reasoning tasks, where consistency is typically defined through exact identity matching. Generalizing this paradigm requires domain-relevant notions of consistency that are flexible with respect to concept-level semantics. We believe this perspective may extend beyond motion trajectories to broader visual and spatial domains, opening new opportunities for structured concept discovery and automatic, training-free verification.

References

[1] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, pages 586–606. Spie, 1992. 8, 15

[2] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992. 8, 15

[3] Jaemin Cho, Abhay Zala, and Mohit Bansal. Visual programming for text-to-image generation and evaluation. In *NeurIPS*, 2023. 7

[4] Jaemin Cho, Yushi Hu, Jason Baldridge, Roopal Garg, Peter Anderson, Ranjay Krishna, Mohit Bansal, Jordi Pont-Tuset, and Su Wang. Davidsonian scene graph: Improving reliability in fine-grained evaluation for text-to-image generation. In *ICLR*, 2024. 7

[5] Fernanda De La Torre, Cathy Mengying Fang, Han Huang, Andrzej Banburski-Fahey, Judith Amores Fernandez, and Jaron Lanier. Llmr: Real-time prompting of interactive worlds using large language models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2024. Association for Computing Machinery. 7

[6] Jack Doyle. Greensock animation platform v3, 2025. 17

[7] Shaoyi Du, Nanning Zheng, Shihui Ying, and Jianyi Liu. Affine iterative closest point algorithm for point set registration. *Pattern Recognition Letters*, 31(9):791–799, 2010. 8, 15

[8] Ethan Eade. Lie groups for computer vision. *Cambridge Univ., Cam-bridge, UK, Tech. Rep*, 2:1, 2014. 2

[9] Douglas Hofstadter and Gary McGraw. Letter spirit: An emergent model of the perception and creation of alphabetic style. In *Technical Report 68, Center for Research on Concepts and Cognition*. 1993. 11

[10] Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A. Smith. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20406–20417, 2023. 7

[11] Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A Ross, Cordelia Schmid, and Alireza Fathi. Scenecraft: an llm agent for synthesizing 3d scenes as blender code. In *Proceedings of the 41st International Conference on Machine Learning*. JMLR.org, 2025. 7

[12] Junqi Jiang, Tom Bewley, Salim I. Amoukou, Francesco Leofante, Antonio Rago, Saumitra Mishra, and Francesca Toni. Representation consistency for accurate and coherent llm answer aggregation, 2025. 6

[13] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017. 3, 10

[14] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976. 16

[15] Shizuo Kaji and Hiroyuki Ochiai. A concise parametrization of affine transformation. *SIAM Journal on Imaging Sciences*, 9(3):1355–1373, 2016. 2

[16] Vladimir V Kisil. Erlangen program at large: an overview. *Advances in applied analysis*, pages 1–94, 2012. 2

[17] Tim Knappe, Ryan Li, Ayush Chauhan, Kaylee Chhua, Kevin Zhu, and Sean O'Brien. Semantic self-consistency: Enhancing language model reasoning via semantic weighting, 2025. 6

[18] Jiaju Ma and Maneesh Agrawala. Mover: Motion verification for motion graphics animations. *ACM Trans. Graph.*, 44 (4), 2025. 3, 7, 10, 17

[19] Nicolas Mellado, Dror Aiger, and Niloy J Mitra. Super 4pcs fast global pointcloud registration via smart indexing. In *Computer graphics forum*, pages 205–215. Wiley Online Library, 2014. 16

[20] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001. 16

[21] Mathias Sablé-Meyer, Kevin Ellis, Josh Tenenbaum, and Stanislas Dehaene. A language of thought for the mental representation of geometric shapes. *Cognitive Psychology*, 139:101527, 2022. 2, 3, 11

[22] Jeong seok Oh and Jay yoon Lee. Latent self-consistency for reliable majority-set selection in short- and long-answer reasoning, 2025. 6

[23] Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. Design2code: Benchmarking multimodal code generation for automated front-end engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3956–3974, 2025. 7

[24] Carsten Steger. Least-squares estimation of anisotropic similarity transformations from corresponding 2d point sets. *Pattern Recognition Letters*, 33(3):349–355, 2012. 7, 11

[25] Jiao Sun, Yushi Hu, Deqing Fu, Royi Rassin, Sjoerd van Steenkiste, Dana Alon, Su Wang, Charles Herrmann, Ranjay Krishna, Da-Cheng Juan, and Cyrus Rashtchian. Dreamsync: Aligning text-to-image generation with image understanding models. In *Synthetic Data for Computer Vision Workshop @ CVPR 2024*, 2024. 7

[26] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04): 376–380, 1991. 16

[27] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022. 2, 4, 6, 15

[28] Jiayi Zhang, Simon Yu, Derek Chong, Anthony Sicilia, Michael R. Tomz, Christopher D. Manning, and Weiyan Shi. Verbalized sampling: How to mitigate mode collapse and unlock llm diversity, 2025. 18

Self-Consistency for LLM-Based Motion Trajectory Generation and Verification

Supplementary Material

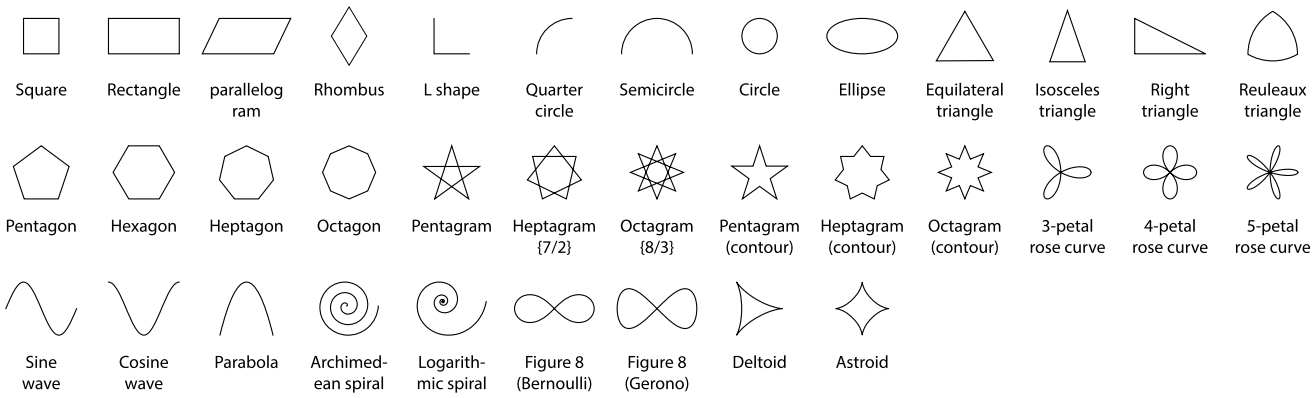


Figure S1. Our benchmark includes 224 prompts of motion trajectories derived from 35 different geometric base shapes. Variations are created by adding specifications to each shape to exercise different warps in our hierarchy of transformations.

Contents

A. Related Work

Self-consistency. Self-consistency [27] was originally introduced as a strategy for improving LLM decoding. In the vanilla set-up, this is done by sampling diverse chains of thought, and then selecting a final answer through a majority vote, where aggregation is performed through identity matching over discrete outputs (numerical values or strings). Subsequent work has expanded this idea by generalizing the unweighted majority vote to take into account consistencies across reasoning traces [12]. Semantic Self-Consistency [17] clusters embeddings of reasoning traces, filtering out outlier responses, and then performs a majority vote over the remaining answers. Latent Self-Consistency [22] similarly clusters embeddings of reasoning traces, does aggregation within this reasoning space, and then returns the answer from this reasoning chain. Collectively, these works demonstrate that self-consistency benefits from more comprehensive aggregation strategies, although their notions of clustering are still limited to bins defined by discrete identity matches. In order to extend self-consistency

to complex visual domains, like families of shapes commonly observed in the trajectories of elements within motion graphics, we propose an approach that determines sample-level consistency under a flexible distance metric.

Visual content verification. As generative AI systems have been used to create visual content across domains, recent work has developed various verification methods to ensure generated outputs match input prompts in terms of satisfying desired properties [5, 10, 11, 23, 25]. For image generation, many works employ a VQA-based verification paradigm. TIFA converts image generation prompts into question-answer pairs that a VQA model can verify [10]. DSG ensures the validity of these questions by structuring them as dependency graphs [4]. DreamSync applies verification in an iterative optimization loop, using VLM feedback to refine generations until they align with the prompt [25]. Verification strategies have also been proposed for other visual modalities. Design2Code proposes automatic metrics for evaluating LLM-generated webpages against design specifications [23]. SceneCraft uses LLM-based question answering to verify 3D Blender scene generation [11], while LLMR applies analogous techniques to VR scene synthesis [5]. VPGen demonstrates that synthesizing verification programs can provide more interpretable verification results than pure VLM evaluation [3].

The work most closely related to ours is MoVer [18], which introduces a first-order logic DSL for verifying motion graphics animations. While this approach enables verification of fine-grained spatio-temporal properties such as motion attributes and timing relationships, it requires animations to be explicitly specified in terms of low-level motion primitives and atomic predicates. More importantly, it cannot effectively express the geometric shape families that people commonly use to describe motion trajectories. In our work, we propose a set of self-consistency-based methods to enable the verification of motion trajectories.

B. Preliminary Details

B.1. Transformation Groups

We describe the full set of Lie transformation groups used in this work.

Rigid $[W_{\text{rgd}} \coloneqq \text{SE}(2)]$: The special Euclidean group that includes all combinations of translations and rotations. This group preserves all distances, angles, and orientation. Note that while this group is commonly denoted $\text{SE}(2)$ it is sometimes also denoted $\text{E}^+(2)$.

Rigid + reflections $[W_{\text{rgd-ref}} \coloneqq \text{E}(2)]$: The full Euclidean group is the set of all rigid transformations in $\text{SE}(2)$ plus reflections. This group allows orientation flips while preserving distances and angles.

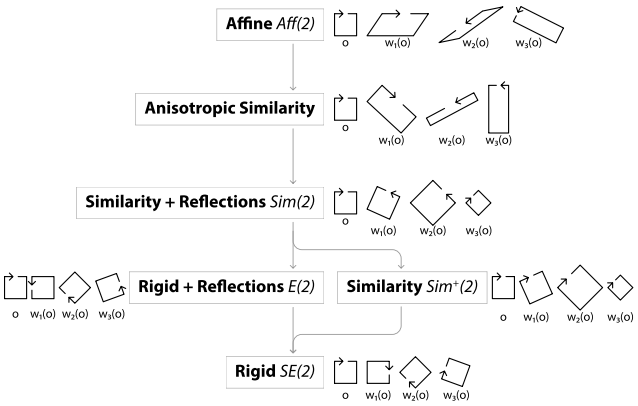


Figure S2. Hierarchy of Lie transformation groups and shape families they induce. Each node represents a transformation group and depicts a prototype square-shaped trajectory o as well as other trajectories $w(o)$ within the corresponding shape family.

Similarity $[W_{\text{sim}} \coloneqq \text{Sim}^+(2)]$: The similarity group includes all combinations of translations, rotations, and uniform scale. This group preserves angles and orientation, but allows uniform changes of size.

Similarity + reflections $[W_{\text{sim-ref}} \coloneqq \text{Sim}(2)]$: The similarity plus reflections group includes all similarity transformations $\text{Sim}^+(2)$ plus reflections. This group allows orientation flips while preserving angles.

Affine $[W_{\text{aff}} \coloneqq \text{Aff}(2)]$: The full affine group including all combinations of translation, rotation, non-uniform scaling, and shear. This group preserves parallel lines and ratios along parallel lines.

Anisotropic similarity $[W_{\text{sim-ani}}]$: The group of all combinations of translation, rotation and non-uniform scale [24]. This group is equivalent to the affine group but without shear. Note that trajectories are specified in the screen space coordinate frame. But many shapes (e.g. rectangles, right triangles, etc.) have canonical coordinate frames aligned with higher-level perceptual features such as right angles, symmetry axes, etc. The anisotropic similarity group allows non-uniform scale before or after rotation. However, since we do not have access to the canonical orientation of a prototype trajectory, such non-uniform scales can appear as a shear (Supplemental Figure S6).

B.2. Hierarchy of Geometric Warps

The Lie transformation groups form a partially ordered hierarchy (Figure S2), where the parent group of warps W_i contains its child group of warps W_j, such that $W_j \subseteq W_i$. Thus, each child group is more restrictive than its parent group. Additional Lie groups can be added to this hierarchy based on their subgroup relationships. For example, equi-affine transformations (area preserving warps

$\mathrm{SA}(2) \subseteq \mathrm{Aff}(2)$) or projective transformations (collinearity preserving warps, $\mathrm{PGL}(2) \supseteq \mathrm{Aff}(2)$) could be added to the hierarchy.

B.3. ICP Implementation Details

We implement our distance metrics using a generalized variant of the iterative closest point (ICP) algorithm [1, 2, 7]. We represent trajectories in a 400 px $\times$ 400 px SVG and compute d_W by resampling each trajectory to $n = 100$ by arc length. Computing d_W for one pair takes on average 67 milliseconds on a standard desktop CPU. In practice, due to discretization and alignment inaccuracies, we consider $t_2 \in \mathcal{F}(t_1, W)$ when $d_W(t_1, t_2)$ is less than a consistency threshold τ. See Section J for the full algorithm.

B.4. Anisotropic Orientation Assumption

As discussed in the transformation groups above, for shape families defined with $W_{\text{sim-ani}}$ (anisotropic similarity) the orientation of the prototype trajectories is important. More specifically, if the prototype trajectory is oriented so that its canonical orientation is not aligned with the screen space x- and y-axes, the transformation group $W_{\text{sim-ani}}$ may appear to shear the trajectory even though shears are not included in the transformation group. Therefore, for these families, we make an additional assumption that the LLM would generate at least some samples with their canonical orientations aligned with our screen-space coordinate frame.

C. Decision Criteria Details

Without any supervision or external knowledge, we do not have access to the correct warp function W for a given prompt. Under different assumptions about the distribution of motion trajectories produced by the LLM, we can design decision criteria that will use our hierarchy of geometric transformation groups to determine an appropriate W. We consider two such decision criteria: Majority-Consensus and Hierarchical-Consistency.

C.1. Majority-Consensus

This procedure identifies the most restrictive transformation group for which a majority cluster first appears. We order the transformation groups from most to least restrictive (ascending the hierarchy). Iteratively, for each group W, we cluster the LLM-sampled trajectories and record the size of the largest cluster. When we observe this clustering produces a dominant cluster that contains a strict majority of all samples (exceeding 50%), we select the corresponding transformation group. This criterion finds the correct transformation group when: (i) the LLM produces correct trajectories more frequently than incorrect ones; (ii) the collection of sampled trajectories diversely covers the modes of variation possible within the transformation group. Under these assumptions, a majority cluster can arise only once the warp aligns with the true geometric structure of the shape family.

C.2. Hierarchical-Consistency

In practice, assumption (ii) might not hold. So we offer an alternative criterion that better accounts for cases where the LLM samples do not cover the modes of variation within the true transformation group well. We order the warp functions from least to most restrictive (descending the hierarchy). Starting from the most permissive warp, we cluster the sampled trajectories, identify the largest cluster, and treat its members as example trajectories from the true shape family. Then, progressively considering more restrictive warps, we check whether clustering would remove any members from the previously observed largest cluster. We select the most restrictive warp for which the largest cluster remains intact. This procedure introduces a new assumption: incorrect LLM generated trajectories will not cluster with correct ones under any warp in our hierarchy. When this holds, the largest cluster begins to lose members precisely when the warp becomes overly restrictive for the true geometric structure of the underlying shape family.

D. Qualitative Results

Figure S3 shows qualitative results under the oracle geometric warps. In the pentagon case, as the false trajectories are distinct in a way that no geometric warps can map them to the correct ones and the largest cluster forms an absolute majority, both Majority-Consensus and Hierarchical-Consistency would correctly choose the ground truth warp of rigid with reflections. For the deltoids, although the samples in the bottom row have a triangular form, no geometric warps can further merge any clusters together. As a result, both Majority-Consensus and Hierarchical-Consistency would select the right warp, even when the largest cluster is less than half of the total number of trajectories. The parabola case is where Majority-Consensus would succeed, as the anisotropic similarity warp is the most restrictive one to produce a largest cluster containing at least half of the trajectories. However, the bottom left trajectory is a skewed parabola and can be merged into the largest group under affine, causing Hierarchical-Consistency to fail.

E. Results Discussion

We discuss how to choose a decision criterion and set the hyperparameters for our approach. See Sections I.1 and J.2 for more details.

Choosing decision criteria. Our extension of self-consistency works in an unsupervised setting, where the correct transformation group W is not known a

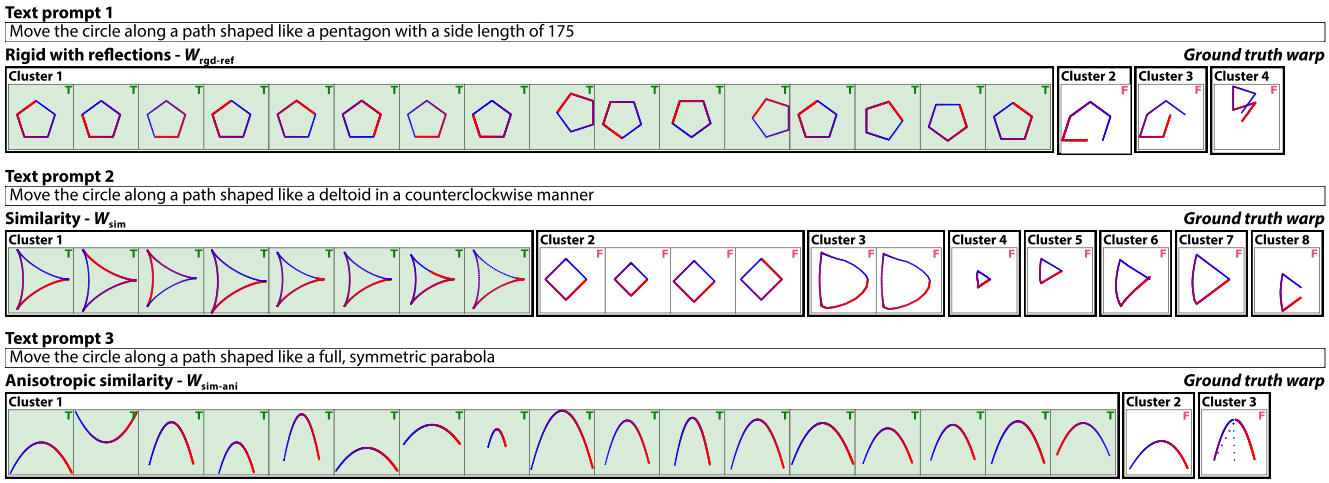


Figure S3. We present clustering results on LLM-generated samples from three example prompts in our dataset. Each trajectory has a ground truth label on the upper right. Clusters colored green are the chosen largest clusters, and we note that their sizes vary, sometimes less than half of the total number of trajectories (middle). In the pentagon case, failed samples (Cluster 2–4) appear visually distinct from the true ones, while some of the failed deltoids (Cluster 4–8) look closer to the correct one with their triangular forms. The rightmost parabola is a skewed one that would have been grouped into the largest cluster under the affine warp.

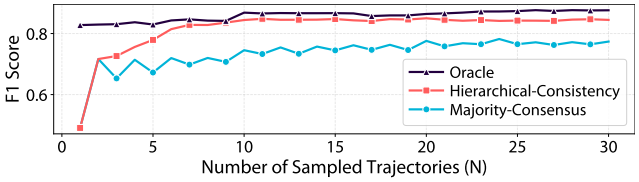


Figure S4. F1 scores across different numbers of sampled trajectories N. Decision criteria performances stabilize after $N = 10$.

priori. We allow users to flexibly choose between the two decision criteria for W estimation that would best support their downstream application needs. One should use Majority-Consensus to prioritize precision, and use Hierarchical-Consistency to balance improved recall while maintaining reasonable precision. The design of Majority-Consensus leads it to have a bias for selecting overly restrictive transformation groups, as it traverses our hierarchy of warps in an ascending order. For verification, we find that this form of conservatism leads to the best precision (85.8) at the expense of recall (66.1). Conversely, Hierarchical-Consistency traverses our hierarchy of warps in a descending order, which finds a nice trade-off between precision and recall, allowing it to achieve the best F1 score for the unsupervised verification task (84.6). These trends are further supported by an analysis of failure modes of these decision criteria: when they choose an incorrect W, Majority-Consensus selects an overly restrictive group 95.6% of the time while Hierarchical-Consistency selects an overly permissive one 80.6% of the time.

Choosing hyperparameters. Our approach has two main hyperparameters, the number of sampled trajectories N and the clustering threshold τ. For N, our algorithm works well with as few as $N = 10$ trajectories. We repeat the verification experiment while varying N from 1 to 30, using GPT-5 as the sampling LLM. As shown in Figure S4, at low N, F1 increases sharply as additional samples give the clustering more signal; performance then stabilizes around $N = 10$, with only marginal gains beyond. This trend holds across all ways of choosing a warp, showing that a modest sample budget of 10 is typically sufficient. While we perform DBSCAN clustering with a small distance threshold τ to account for discretization errors, our approach is not sensitive to the choice of its value. To show this, we repeat the verification experiment with Hierarchical-Consistency while sweeping τ from 0.25 to 8.0, a 32× range. We observe that F1 drops by only 7.2 percentage points across the full sweep.

F. Motion Trajectory Benchmark Details

Our benchmark consists of 224 motion trajectory prompts describing 35 different basic shapes, as shown in Figure S1. Each prompt is paired with a ground truth shape family following the definition of Equation 1 in the main paper – one prototype trajectory and a transformation group. Table S1 reports the number of prompts associated with each transformation group. The distribution is determined by the geometric properties of the base shapes (e.g., a circle cannot have anisotropic similarity or affine warps). We discuss and show examples of applying our approach to shape families that do not follow this definition in Section H.1. We illustrate all 224 prompts with their ground truth shape families in our benchmark in Figure S11–S17.

Table S1. Number of prompts per warp in our benchmark.

$W_{\text{rgd-ref}}$	W_{rgd}	W_{sim}	$W_{\text{sim-ref}}$	$W_{\text{sim-ani}}$	W_{aff}	Total
105	44	38	23	12	2	224

We use a template-based approach to generate the prompts, following MoVer [18] and CLEVR [13]. Each prompt starts with the main sentence `Animate the <SVG element> to move along a path shaped like <shape name>.` We manually pair a prompt to its ground truth shape family consisting of a Lie group of transformations and a prototype trajectory based on the shape properties. For example, we pair "circle" with a circular trajectory and the warp $W_{\text{sim-ref}}$ (similarity with reflections), "ellipse" with an elliptical path and $W_{\text{sim-ani}}$ (anisotropic similarity), and "parallelogram" to a parallelogram path with W_{aff} (affine). We note that, for shapes that are periodic (sine and cosine waves) or self-repeating (Archimedean and logarithmic spirals), the prompts need to specify the number of periods or turns, as changes in these values effectively create different shape families (e.g., a spiral with 2 turns cannot be geometrically warped into a spiral with 3 turns). We add a new sentence after the main sentence for this value: `Complete <periods/turns> of <shape name>.`

To introduce prompt variations that exercise all geometric warps in our hierarchy, we add modifiers to a prompt in the form of additional specifications on a shape. The first type of specifications is *shape orientation*, where we require a shape to be traversed in either a clockwise or counterclockwise manner. Adding this specification removes reflections from the warps. We use the following template sentence: `Traverse the path in a <orientation> manner.`

The second type of specifications is *shape size*. For some shapes, their sizes can be fully specified by one parameter (e.g., radius for circles and side length for polygons). Others require two parameters, such as width and height for rectangles and triangles, and horizontal extent and amplitude for sine and cosine waves. Fully specifying a shape's size reduces the transformation group to either W_{rgd} (rigid) or $W_{\text{rgd-ref}}$ (rigid with reflections). Alternatively, we can specify a shape's size in terms of ratios (e.g., a rectangle with a 4:3 aspect ratio), and doing so changes the transformation group to either W_{sim} or $W_{\text{sim-ref}}$. We append a prepositional phrase to the main sentence when adding size specifications: `a <shape name> with <size param name> of <size param value>` (e.g., *"a rectangle with a width of 50 px and a height of 80 px."*).

G. Decision Criteria Alternatives

In Section 4 of the main paper we describe how our method is able to recover a shape family from an input prompt. A critical component of this procedure is how to estimate which transformation group W, from our hierarchy of Lie transformation groups (Figure 3, main paper), is most appropriate. We introduce two decision criteria for selecting W, based on underlying assumptions made in the sampled motion trajectory distributions produced by an LLM: Majority-Consensus (Section 4.3.1) and Hierarchical-Consistency (Section 4.3.2). How do we know whether the decision criteria we propose select useful transformation groups? One way to justify our design of these procedures is by comparing their performance against very simple decision criteria alternatives. To this end, we consider two additional 'baseline' decision criteria, which we expect will recover worse shape families compared with Majority-Consensus and Hierarchical-Consistency. *Most-Restrictive* is a criterion that always selects the most restrictive transformation group in our hierarchy (rigid). *Least-Restrictive* is a criterion that always selects the least restrictive transformation group in our hierarchy (affine).

Motion trajectory generation With the same experimental set-up as described in Section 5.2 of the main paper, we compare how the different decision criteria aid in the task of motion trajectory synthesis. We report results of this experiment in Table S2. As a reminder, the metric we track is the "accuracy" of the generations, i.e., whether the production is a member of the ground-truth shape family from our benchmark dataset. For better numerical stability, we compute this accuracy as the ratio of true to false trajectories in the chosen cluster, averaged across tasks in the dataset. Note that for LLM-Direct, we say that the "chosen cluster" includes all trajectories produced from the same prompt.

We find that the two baseline decision criteria do strictly worse compared with the more sophisticated alternatives. Even these simple ways of choosing W do provide some benefit within the self-consistency paradigm in terms of improving the accuracy of the LLM's generations. For instance, Most-Restrictive is able to increase the accuracy over LLM-Direct by between 3 and 4 absolute percentage points. These results show that even with sub-optimal choices of W, self-consistency can benefit from the greater flexibility afforded in terms of checking whether generations match beyond identity comparisons.

Motion trajectory verification Following the experimental set-up as described in Section 5.3 of the main paper, we compare how these alternative decision criteria perform on the task of motion trajectory verification. We report results of this experiment in Table S3.

Table S2. Motion trajectory generation experiment with two additional baseline decision criteria of always selecting the most or least restrictive warps. We observe that they perform strictly worse than the two methods proposed in the main paper.

Method	Decision Criteria	GPT-4.1	GPT-5
LLM-Direct	—	62.1	79.1
Ours	Majority-Consensus	**68.0**	**83.3**
Ours	Most-Restrictive	66.5	82.5
Ours	Hierarchical-Consistency	66.7	82.6
Ours	Least-Restrictive	64.6	82.1
Ours	Oracle	68.5	83.5

Table S3. Motion trajectory verification experiment (Table 2 of the main paper) with additional decision criteria of always selecting the most or least restrictive warps.

Method	Decision Criteria	Precision	Recall	F1 Score
GPT-4.1	—	62.0	96.9	75.6
GPT-5	—	74.0	84.7	79.0
Ours	Majority-Consensus	85.8	66.1	74.6
Ours	Most-Restrictive	86.1	42.1	56.5
Ours	Hierarchical-Consistency	80.5	89.0	84.6
Ours	Least-Restrictive	65.0	91.8	76.1
Ours	Oracle	87.9	83.3	85.6

Hierarchical-Consistency continues to achieve the best F1 score, finding a nice balance between precision and recall. One way to view Least-Restrictive is as a partial implementation of Hierarchical-Consistency; this simplification to never descend the hierarchy results in a steep decrease in precision (15 points) with only moderate increases to recall (3 points). In a similar fashion, Most-Restrictive can also be considered a partial implementation of Majority-Consensus. Always choosing the most restrictive transformation group provides a marginal benefit for precision (.3 points), but it dramatically drops the associated recall (24 points).

H. Shape Family Limitations

H.1. Shape Families Requiring Multiple Prototypes

In our work, we define a shape family $\mathcal{F}(o, W)$ in Equation 1 of the main paper with one prototype and one geometric warp, and have demonstrated that a wide variety of common shape families (Figure S1) that people use to describe motion trajectories falls under this definition [21].

However, there are shape families that cannot be represented this way. One type of such families requires multiple but finitely many prototypes under one particular warp. For example, as described in the main paper, the shape family "heptagram" needs two distinct prototypes: heptagram 7/2 and heptagram 7/3 under the similarity warp (Figure S5). Another type of families either needs an infinite amount of prototypes under one warp, or one prototype with some sophisticated, non-geometric warp. An example of this is the family of any letter of the English alphabet, like the capital letter M (Figure S5). As demonstrated by Hofstadter and McGraw [9], since there are countless ways to produce a trajectory that a human would perceive as an 'M-shape', we either need an infinite number of prototypes or a warp beyond the geometric transformations we consider.

In Figure S5, we prompted for the heptagram and capital M shapes. In both cases, while the largest clusters do contain correct trajectories for generation, applying them for verification would result in limited recall, as correct shapes that cannot be mapped to by the single prototype and the warp would be labeled as false.

H.1.1. Extension to Multi-Prototype Shape Families

For shape families requiring a finite number of prototypes under one warp, we experimented with a simple modification to Hierarchical-Consistency that returns all largest clusters whose sizes are within 20% of one another, instead of only the single largest cluster. We tested this modification on four multi-prototype prompts under $W_{\text{sim-ref}}$: heptagram (2 prototypes: {7/2}, {7/3}; Figure S5), heptagram outline (2 prototypes: {7/2}, {7/3}), octagram outline (2 prototypes: {8/2}, {8/3}), and hendecagram (4 prototypes: {11/2}, {11/3}, {11/4}, {11/5}). With regular Hierarchical-Consistency, the verification F1 score on these multi-prototype prompts is 71.0. After applying the 20% tolerance modification, F1 improves to 88.9. Meanwhile, the modification has a negligible effect on single-prototype prompts, with F1 decreasing only slightly from 84.6 to 84.5, as the tolerance occasionally causes an additional outlier cluster to be selected. This small experiment suggests that we can potentially extend our approach to handle multi-prototype shape families by returning the largest clusters that satisfy certain statistical relationships with one another. Future work should more comprehensively evaluate this on a larger set of multi-prototype prompts and explore other decision criteria for selecting multiple clusters.

H.2. Shape Families with Anisotropic Similarity

For shape families with the anisotropic similarity warp, when a shape is not aligned with the screen coordinate frame (bottom row of Figure S6), a skew can be introduced to distort it out of its family if a non-uniform scaling is applied to it [24]. For symmetric shapes like the ones shown in Figure S6, their lines of symmetry need to align with the screen coordinate axes. For asymmetric shapes with right

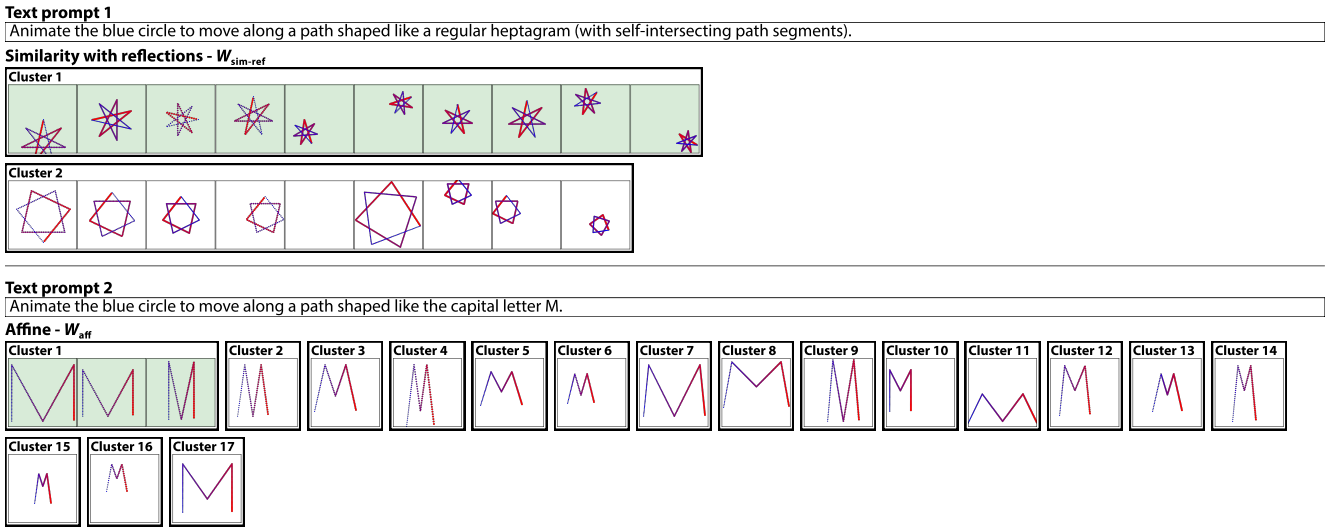


Figure S5. We demonstrate a limitation of our self-consistency approach in terms of the definition of a shape family. In text prompt 1, "heptagram" has two distinct prototypes: heptagram $\{7/2\}$ and heptagram $\{7/3\}$. Under the warp $W_{\text{sim-ref}}$, cluster 1 contains correct $\{7/3\}$ trajectories, while cluster 2 has $\{7/2\}$. While, for generation, our approach would produce a correct interpretation of heptagram, we would mark all heptagram $\{7/2\}$ as false for verification. Similarly, for the letter Ms in text prompt 2, our approach would generate a correct form of the letter, but verify all other letter Ms that are not an affine warp of the shapes in cluster 1 as false.

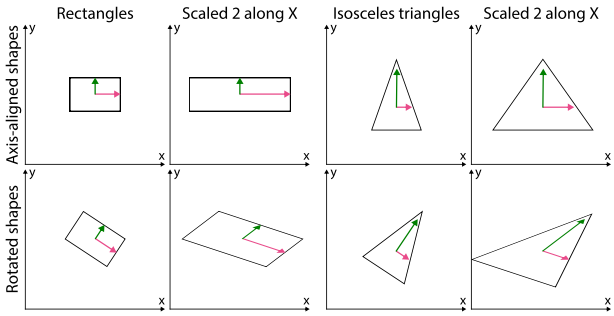


Figure S6. When a shape is aligned with the screen space's coordinate frame, applying an anisotropic scale of 2 along the x-axis does not introduce skews to the shapes (top). However, once a shape is rotated out of alignment, non-uniform scales effectively skew the shapes out of their respective shape families (bottom).

angles (e.g., right triangles), the lines along the right angles need to be in alignment so that they are preserved under scaling. This means that, in order to properly warp to all other members of the family, we need a prototype that is in alignment with the screen space's coordinate frame for non-uniform scaling to be applied without skew. We manually ensure this when we are constructing ground truth shape families for our dataset, and assume that the LLM will generate some trajectories in such canonical orientations.

Table S4. Our decision criteria has different failure modes when not selecting the ground truth warp. Majority-Consensus selects more restrictive distance metrics for 108 prompts (95.6% of the time when it is not matching with the ground truth), while Hierarchical-Consistency is more conservative for 50 prompts (80.6% of the time). Note that both decision criteria choose distance metrics that are on the same level of the hierarchy but different from the ground truth for 1 and 3 prompts respectively.

Decision Criteria	More Restrictive	Match	Less Restrictive
Majority-Consensus	48.2%	49.6%	1.8%
Hierarchical-Consistency	4.0%	72.3%	22.3%

I. Failure Modes

I.1. Decision Criteria Failure Modes

Our method can recover a shape family from an input prompt by using some decision criteria to estimate an appropriate transformation group W. In Section 4.3 of the main paper, we introduce two such decision criteria that analyze the distribution of motion trajectories generated by an LLM. When different sets of assumptions are met, in terms of how the LLM generates motion trajectories, these decision criteria will predict the correct transformation group. For Majority-Consensus the assumptions it makes are (i) the LLM produces correct trajectories more frequently than incorrect ones; (ii) the collection of sampled trajectories diversely covers the modes of variation possible within the transformation group. Hierarchical-Consistency relaxes this second assumption, but introduces a new assumption

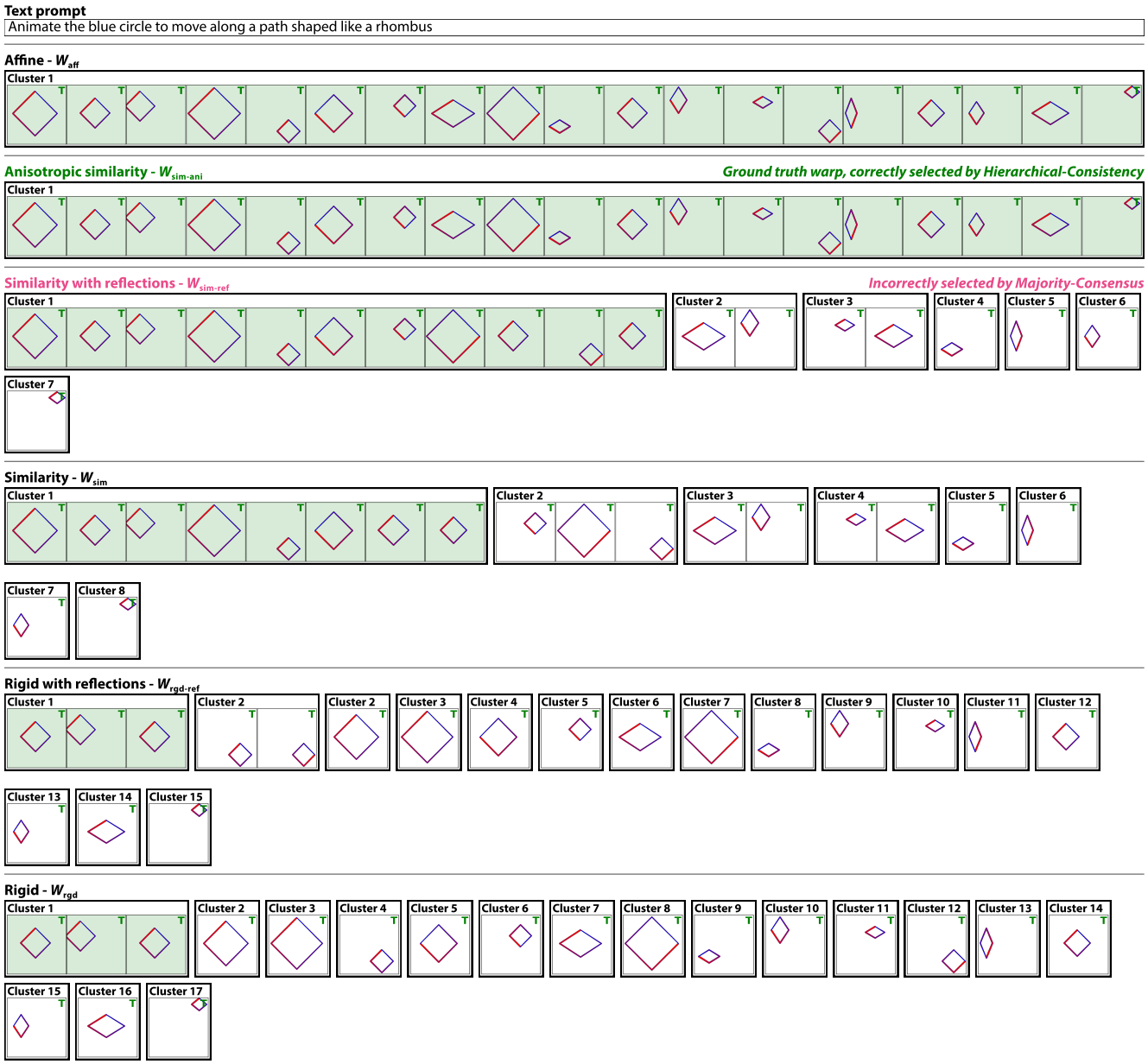


Figure S7. We demonstrate a case where Hierarchical-Consistency chooses the correct warp but Majority-Consensus fails by selecting a more restrictive warp. The prompt (top) asks for a rhombus-shaped path. We show clustering results of all warps in our hierarchy, with largest clusters under each warp highlighted in green. Each trajectory has a ground truth correctness label on the upper right.

in that incorrect trajectories produced by the LLM will not cluster with the correct trajectories produced by the LLM under any transformation group in our hierarchy. In cases where these assumptions are broken, it is possible for these decision criteria to fail (i.e. recover a different transformation group compared with the ground-truth annotation).

To demonstrate such cases, we include qualitative examples in Figure S7 and Figure S8. In each figure, we show LLM sampled motion trajectories from a text prompt (top of figure), and how these motion trajectories are clustered under different transformation groups from our hierarchy. The top transformation group we show is affine, the least restrictive, and transformation groups get progressively more restrictive, ending with rigid. For each transformation group, we highlight the largest cluster in green. Each trajectory is labeled with a T/F value in the top-right corner, indicating whether it is a member of our ground-truth shape family.

In Figure S7, we demonstrate a case where Hierarchical-Consistency identifies the correct W, while Majority-Consensus fails. For this prompt, all of

Figure S8. An example case where Majority-Consensus correctly chooses the ground truth warp but Hierarchical-Consistency fails by selecting a less restrictive warp. The prompt (top) asks for a path shaped like an equilateral triangle traversed in a counterclockwise manner. We show clustering results of all warps in our hierarchy.

the LLM generated motion trajectories were correct. For Hierarchical-Consistency, for the affine transformation group it identifies the largest cluster, and observes that this cluster does not lose any members when the transformation group is made more restrictive with anisotropic similarity, which is the correct transformation group for this shape family. Trying to make W more restrictive (similarity with reflections) would break this cluster, so Hierarchical-Consistency succeeds by stopping at the previous step. In contrast, Majority-Consensus identifies that under the clustering produced by similarity with reflections, over half of the generations would be included in the largest cluster, so it incorrectly would choose an overly restrictive transformation group. This sampling breaks an assumption made by Majority-Consensus, as the LLM produced trajectories do not provide a balanced enough coverage over the modes of variation under the ground-truth transformation group (anisotropic similarity).

In Figure S8, we demonstrate a case where Majority-Consensus identifies the correct W, while Hierarchical-Consistency fails. In this case, 10 LLM generations were correct and 9 LLM generations were

incorrect, as they produced paths that did not traverse in a clockwise manner (note time is indicated as a progression from blue to red in the renders). The correct transformation group in this case is similarity, as more permissive transformation groups would cause matches that violate the traversal orientation specification. Majority-Consensus starts with the clustering produced by the most restrictive transformation group (rigid), which has a largest cluster of size 3, and progressively considers less restrictive transformation groups until it finds a cluster with at least 10 members, correctly identifying similarity should be used as W. In contrast, Hierarchical-Consistency starts with affine, and once again identifies that all of the LLM productions are grouped into a single cluster. This cluster only loses members on the transition from anisotropic similarity to similarity with reflections, so Hierarchical-Consistency will incorrectly choose an overly permissive transformation group. In this case, some of the incorrect LLM samples would cluster with correct samples under the affine transformation group, breaking an assumption made by Hierarchical-Consistency.

So, depending on the distribution of the LLM produced motion trajectories, these decision criteria can fail in their task of recovering the correct transformation group. We quantitatively analyze the prevalence of these failure modes in Table S4. For our two proposed criteria, Hierarchical-Consistency and Majority-Consensus, we record for all 224 prompts how the chosen transformation group compares with the ground-truth transformation group in the benchmark dataset. *Match* indicates the criteria chose the correct transformation group, while *more restrictive* indicates the criteria chose a transformation group too low in the hierarchy, and *less restrictive* indicates the criteria chose a transformation group too high in the hierarchy.

In general, we find that Hierarchical-Consistency more reliably identifies the correct W compared with Majority-Consensus (72% vs 49%). This likely indicates that the assumptions made by Hierarchical-Consistency are more often met compared with the assumptions made by Majority-Consensus. From the qualitative examples, we can see the LLM struggles to produce sufficiently diverse samples that cover all of the possible modes of variation, so this is likely a major source of error for Majority-Consensus. Beyond checking whether or not these decision criteria recover the exact ground-truth W, we can additionally analyze how these methods tend to fail. Majority-Consensus starts at the bottom of the hierarchy, stopping when a majority clustering is found. As such, Majority-Consensus is overly prone to producing more restrictive transformation groups when it makes mistakes (25 times more likely to have a more restrictive error compared with a less restrictive error). Hierarchical-Consistency starts at the top of the hierarchy, with a stopping criterion when the majority cluster begins to break apart. As such, Hierarchical-Consistency is overly prone to producing transformation groups that are too loose when it makes mistakes (5 times more likely to have a less restrictive error compared with a more restrictive error). These quantitative trends match the qualitative examples.

I.2. LLM Failure Modes

The general self-consistency approach makes an assumption that the LLM is likely to produce the correct response more often than any other incorrect response [27]. We examine cases where this assumption does not hold and present quantitative results in Figure S9. Applying our self-consistency approach with the oracle warps to the trajectories generated by GPT-5 from the 224 prompts in our dataset (same experimental set-up as Section 5.2 of the main paper), we see 16.1% of the prompts with largest clusters that do not contain any correct trajectories. For GPT-4.1 generated trajectories, we have 31.3% (70) such clusters.

We show examples of these failure cases in the bottom half of Figure S9. Text prompt 3 asks for a trajectory in the shape of an outline of a pentagram, and the oracle similarity warp puts the generated trajectories into three clusters. While the LLM (GPT-5) was able to produce the correct trajectory 5 times (Cluster 2), it generates more instances of a regular pentagram shape with the internal path crossovers in both clockwise and counterclockwise fashion. Text prompt 4 asks for counterclockwise reuleaux triangle-shaped trajectories. Under the ground truth rigid transformation warp, the generated trajectories are grouped into three clusters. The LLM (GPT-4.1) similarly generates the correct trajectory 5 times (Cluster 3), but falsely traces the construction lines of a reuleaux triangle for the other 14 times. In both of these cases, the incorrect trajectories formed largest clusters in the form of relative majority with 9 trajectories. As the LLM produces more trajectories outside of the ground truth shape families than inside, the general assumption of self-consistency does not hold and the largest clusters do not contain any correct trajectories.

J. Method Implementation Details

J.1. Computing Warp-Invariant Distance

Algorithm 1 presents our ICP-based approach [1, 2, 7] for computing the warp-invariant distance between a source trajectory R and target trajectory T under a geometric warp W. We begin by resampling both trajectories to have n equally-spaced points along their arc length. This resampling establishes initial point correspondences, where we assume that points at the same index in R and T correspond to one another. The outer loop then randomly selects 3 non-repeated, non-collinear point indices $\mathcal{I}$ from these trajectories. These three points provide sufficient constraints to

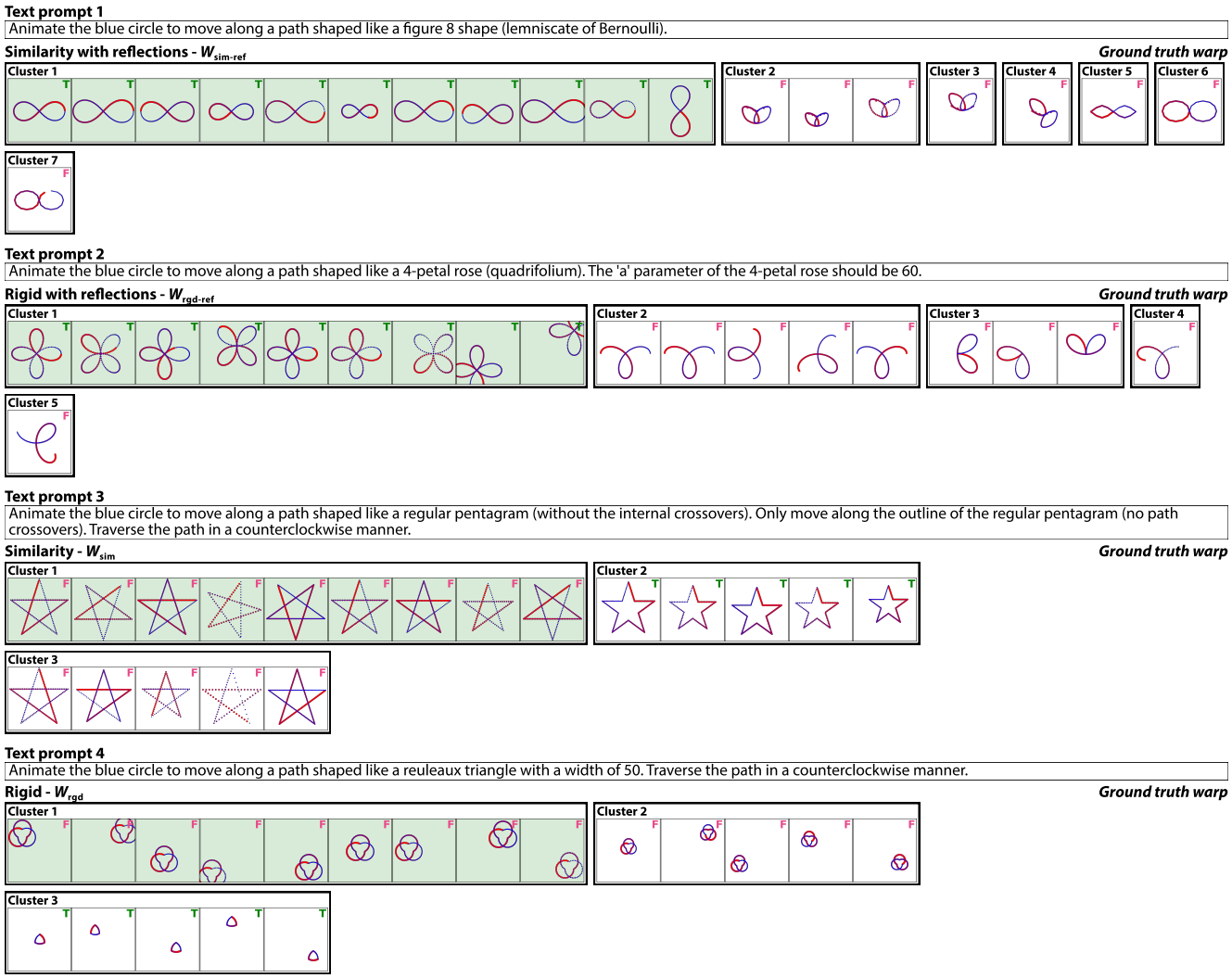


Figure S9. More results of applying our self-consistency approach with the oracle warps. The top half shows two success cases where the largest clusters do correspond to the correct shape families (figure 8 and 4-petal rose). The bottom half shows two failure cases where the LLM did not produce more members of the ground truth shape families than non-members (pentagram outline and reuleaux triangle).

estimate most geometric transformations in our hierarchy.

Given the sampled point indices, the inner loop iteratively refines the transformation A. At each iteration, we: (1) estimate a transformation A that aligns the sampled points from R_{trans} to the corresponding points in T, (2) apply this transformation to the source trajectory R, (3) resample the transformed trajectory for a new set of corresponding points, and (4) compute the average point-to-point distance. If the change in distance falls below a convergence threshold ϵ, the inner loop terminates. Otherwise, we map the newly sampled points back to the original R space by undoing the transformation and repeat. This iterative process effectively "nudges" points along trajectory R to better align with T under the transformation A.

After convergence, we check if the current alignment achieves a lower distance than previous iterations and update the best transformation $A_{\min}$ accordingly. The algorithm terminates when the distance falls below a threshold τ or after a maximum number of outer iterations. Our ICP-based implementation is a proof of concept to show the viability of our self-consistency approach for motion trajectories. It would likely be possible to improve its runtime with efficient variants in the future [19, 20].

Transformation estimation. `EstimateTransform` in line 11 of Algorithm 1 computes A based on the geometric warp type $\mathcal{W}$. For rigid and similarity transformations (with or without reflections), we use the Kabsch-Umeyama algorithm [14, 26], which computes optimal rotation, translation, and optionally uniform scaling and reflection via SVD. For affine warps, we directly solve the linear system relating source to target points. For anisotropic

Algorithm 1 ICP-based Trajectory Distance Metric

Input: Source trajectory R, target trajectory T, and geometric warp type $\mathcal{W}$
Output: Minimum distance $d_{\min}$ between R and T with corresponding transformation matrix $A_{\min}$

$T \leftarrow \text{ResampleByArcLength}(T, n)$
$d_{\min} \leftarrow \infty$, $A_{\min} \leftarrow \mathbf{I}$
for $i = 1$ to M_{outer} **do**
  $R_{\text{trans}} \leftarrow \text{ResampleByArcLength}(R, n)$
  $d_{\text{prev}} \leftarrow \infty$
  $\mathcal{I} \leftarrow$ Randomly sample 3 point indices
  for $j = 1$ to M_{inner} **do**
    $A \leftarrow \text{EstimateTransform}(R_{\text{trans}}[\mathcal{I}], T[\mathcal{I}], \mathcal{W})$
    $R_{\text{trans}} \leftarrow A \cdot R$ ▷ Apply transformation
    $R_{\text{trans}} \leftarrow \text{ResampleByArcLength}(R_{\text{trans}}, n)$
    $d_{\text{avg}} \leftarrow \frac{1}{n} \sum_{k=1}^{n} \|R_{\text{trans}}[k] - T[k]\|_2$
    if $|d_{\text{avg}} - d_{\text{prev}}| < \epsilon$ **then**
      break ▷ Inner loop converged
    end if
    $d_{\text{prev}} \leftarrow d_{\text{avg}}$
    $R_{\text{trans}} \leftarrow A^{-1} \cdot R_{\text{trans}}$ ▷ Undo transformation
  end for
  if $d_{\text{avg}} < d_{\min}$ **then**
    $A_{\min} \leftarrow A$, $d_{\min} \leftarrow d_{\text{avg}}$ ▷ Update alignment
  end if
  if $d_{\text{avg}} < \tau$ **then**
    break ▷ Outer loop converged
  end if
end for
return $d_{\min}, A_{\min}$

similarity, we test both $A = M_{\text{trs}} \cdot M_{\text{rot}} \cdot M_{\text{scl}}$ and $A = M_{\text{trs}} \cdot M_{\text{scl}} \cdot M_{\text{rot}}$ decomposition orders and select the transformation yielding the smaller distance.

Handling closed trajectories Our assumption that points with the same indices after arc-length resampling correspond to one another holds for open trajectories with distinct starting and end points. However, closed trajectories (e.g., circles, squares) may have arbitrary starting points. This means that, if T and R are the same closed trajectories but with distinct starting points, our algorithm will not be able to find an A that produces zero distance between them. To address this, we extend our algorithm with a coarse-to-fine search over different starting points along R. We compute the distance metric with each point in R taken as the starting point and report the minimum distance across all starting points.

Table S5. F1 scores of verification by Hierarchical-Consistency and the average number of DBSCAN clusters across the distance threshold τ.

	$\tau = 0.25$	$\tau = 0.5$	$\tau = 1.0$	$\tau = 2.0$	$\tau = 4.0$	$\tau = 8.0$
F1 score	84.0	84.6	83.8	83.1	80.3	76.8
Clusters (avg±std)	4.0±4.9	3.7±4.8	3.5±4.5	3.1±4.0	2.6±3.1	2.1±2.1

J.2. Sensitivity to DBSCAN Distance Threshold

Equation 2 of the main paper defines that two trajectories in the same shape family should have a distance of 0. We perform DBSCAN clustering with a small threshold τ (Section 4.2 of the main paper) only to account for discretization errors during resampling in our distance metric computation. Our approach is not sensitive to the value of τ. As shown in Table S5, increasing τ by a factor of 32 (from 0.25 to 8.0) results in only a 7.2 percentage point drop in F1 score.

J.3. Generating Motion Trajectories with an LLM

To use an LLM for motion trajectory animation generation, we follow the approach proposed by MoVer [18]; we prompt an LLM with a system message containing overall instructions and the complete documentation of an animation API (GSAP [6]), followed by a user prompt that contains the static SVG code to be animated, the description of a motion trajectory, and additional instructions.

We modify the MoVer approach in two places. First, we updated the animation API with an additional function `setProperty()` that allows changing an element's properties like positions without adding animation tweens.

```
/**
 * This function immediately sets properties on
    an SVG element without animation. Use this
    function to initialize or instantly update
    element properties such as position, scale,
    rotation, opacity, etc. This is useful for
    setting up initial states before animating.
 * @param {object} element - The SVG element
    whose properties should be set.
 * @param {object} properties - An object
    containing property-value pairs to set on the
     element. Common properties include x, y,
    scaleX, scaleY, rotation, opacity,
    transformOrigin, and svgOrigin.
 * @returns {void} - This function does not
    return anything.
 * @example
 * // Set the translate transform value of the
    element to [100, 200].
 * // Here, x, y values set the translation of
    the element (displacement from its original
    position at the start of the animation).
 * setProperty(circle, { x: 100, y: 200 });
 */
function setProperty(element, properties)
```

For the user prompt, we append additional instructions shown below after the motion trajectory prompt. At the end,

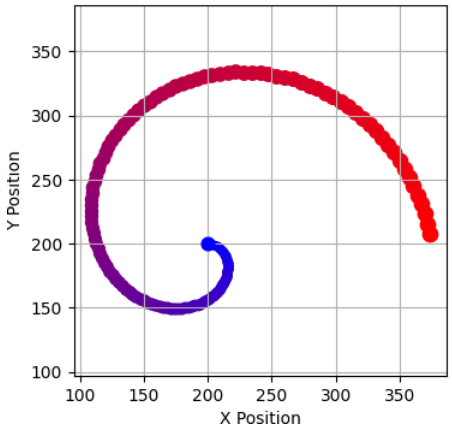


Figure S10. We prompt a VLM with a scatter plot representation of a motion trajectory. Each dot is the center position of the animated element and the color indicates time (transition from blue to red).

we instruct the LLM to produce five diverse animations following the diversity prompting strategy [28].

```
SVG Code:
{{ svg code }}

Prompt:
{{ motion trajectory prompt }}

## Additional Instructions
- Note that the negative y-axis is the upward direction in the SVG scene.
- To start a path animation, use setProperty() to position the element at the beginning of the path. Be mindful of the SVG viewbox and not move the element and the path out of view.
- For curved paths, use multiple consecutive very short linear translations to achieve smooth motions.
- The entire animation should last between 1 and 5 seconds unless otherwise specified.

Generate 5 animations to the prompt, each within a separate ```javascript ``` tag.
Each animation must include a numeric probability value as a comment within its code block.
Please sample at random from the tails of the distribution, such that the probability of each response is less than 0.10.
```

J.4. Verifying Motion Trajectories with a VLM

Our experiments use two VLMs (GPT-4.1 and GPT-5) as baseline verification methods, and we provide details here on our prompting approach. We represent the motion trajectory center point time series data of animation as a scatter plot where the color of each dot transitions from blue to red to indicate the passage of time (Figure S10).

We prompt a VLM with a system message asking it to output a true/false label indicating whether a trajectory animation follows a trajectory prompt or not, with instructions on how to read the scatter plot. We then add a user message containing the motion trajectory prompt and the plot.

```
You are an expert in evaluating how well motion trajectories follow given prompts. Given a plot of a motion trajectory, your task is to assess whether the trajectory follows the prompt (True or False). Blue indicates the beginning of the trajectory and red indicates the end of the trajectory. The color transitions from blue to red as the trajectory progresses across time.

Please output your response as a JSON object with the following format:
```json
{
    "<trajectory_id (the index of the trajectory file)>": 0 or 1, # 0 means False, 1 means True
}
```
```

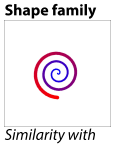
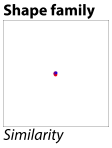
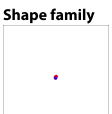
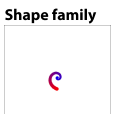
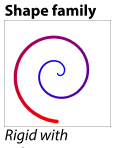
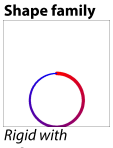
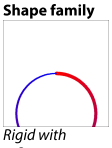
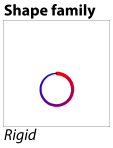
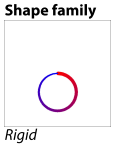
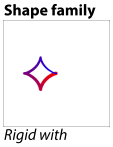
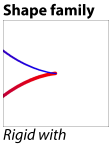
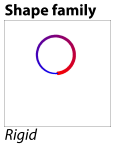
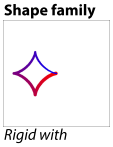
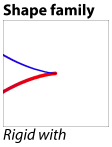
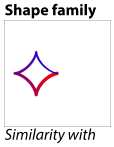
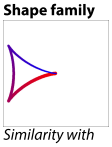
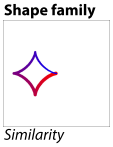
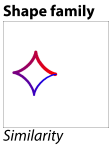
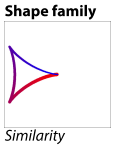
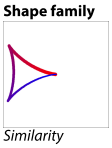
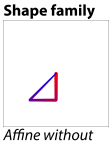
Shape family	Text prompt 1	Shape family	Text prompt 2	Shape family	Text prompt 3	Shape family	Text prompt 4
 *Similarity with reflections*	Animate the blue circle to move along a path shaped like an archimedean spiral (initial radius should be 0 and move outward from the scene center). Complete 1 turn of the archimedean spiral.	 *Similarity*	Animate the blue circle to move along a path shaped like an archimedean spiral (initial radius should be 0 and move outward from the scene center). Complete 1 turn of the archimedean spiral. Traverse the path in a counterclockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like an archimedean spiral (initial radius should be 0 and move outward from the scene center). Complete 1 turn of the archimedean spiral. Traverse the path in a clockwise manner.	 *Similarity with reflections*	Animate the blue circle to move along a path shaped like an archimedean spiral (initial radius should be 0 and move outward from the scene center). Complete 2 turns of the archimedean spiral.
Shape family	**Text prompt 6**	**Shape family**	**Text prompt 7**	**Shape family**	**Text prompt 8**	**Shape family**	**Text prompt 5**
 *Similarity*	Animate the blue circle to move along a path shaped like an archimedean spiral (initial radius should be 0 and move outward from the scene center). Complete 2 turns of the archimedean spiral. Traverse the path in a clockwise manner.	 *Similarity with reflections*	Animate the blue circle to move along a path shaped like an archimedean spiral (initial radius should be 0 and move outward from the scene center). Complete 3 turns of the archimedean spiral.	 *Similarity*	Animate the blue circle to move along a path shaped like a logarithmic spiral (initial radius should be 1 and growth rate should be 0.3. move outward from the scene center). Complete 1 turn of the logarithmic spiral. Traverse the path in a counterclockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like an archimedean spiral (initial radius should be 0 and move outward from the scene center). Complete 2 turns of the archimedean spiral. Traverse the path in a counterclockwise manner.
Shape family	**Text prompt 9**	**Shape family**	**Text prompt 10**	**Shape family**	**Text prompt 11**	**Shape family**	**Text prompt 12**
 *Similarity*	Animate the blue circle to move along a path shaped like a logarithmic spiral (initial radius should be 1 and growth rate should be 0.3. move outward from the scene center). Complete 1 turn of the logarithmic spiral. Traverse the path in a clockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like a logarithmic spiral (initial radius should be 1 and growth rate should be 0.3. move outward from the scene center). Complete 2 turns of the logarithmic spiral. Traverse the path in a counterclockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like a logarithmic spiral (initial radius should be 1 and growth rate should be 0.3. move outward from the scene center). Complete 2 turns of the logarithmic spiral. Traverse the path in a clockwise manner.	 *Rigid with reflections*	Animate the blue circle to move along a path shaped like an archimedean spiral (initial radius should be 0 and move outward from the scene center). Complete 2 turns of the archimedean spiral. The distance between each turn should be 30.
Shape family	**Text prompt 13**	**Shape family**	**Text prompt 14**	**Shape family**	**Text prompt 15**	**Shape family**	**Text prompt 16**
 *Rigid with reflections*	Animate the blue circle to move along a path shaped like an archimedean spiral (initial radius should be 0 and move outward from the scene center). Complete 2 turns of the archimedean spiral. The distance between each turn should be 60.	 *Rigid with reflections*	Animate the blue circle to move along a path shaped like an archimedean spiral (initial radius should be 0 and move outward from the scene center). Complete 2 turns of the archimedean spiral. The distance between each turn should be 90.	 *Similarity with reflections*	Animate the blue circle to move along a path shaped like a circle.	 *Rigid with reflections*	Animate the blue circle to move along a path shaped like a circle with a radius of 100.
Shape family	**Text prompt 17**	**Shape family**	**Text prompt 18**	**Shape family**	**Text prompt 19**	**Shape family**	**Text prompt 20**
 *Rigid with reflections*	Animate the blue circle to move along a path shaped like a circle with a radius of 150.	 *Rigid with reflections*	Animate the blue circle to move along a path shaped like a circle with a radius of 200.	 *Rigid with reflections*	Animate the blue circle to move along a path shaped like a circle with a radius of 50.	 *Rigid*	Animate the blue circle to move along a path shaped like a circle with a radius of 60. Traverse the path in a counterclockwise manner.
Shape family	**Text prompt 21**	**Shape family**	**Text prompt 22**	**Shape family**	**Text prompt 24**	**Shape family**	**Text prompt 26**
 *Rigid*	Animate the blue circle to move along a path shaped like a circle with a radius of 60. Traverse the path in a clockwise manner.	 *Rigid*	Animate the blue circle to move along a path shaped like a circle with a radius of 70. Traverse the path in a counterclockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like a circle. Traverse the path in a counterclockwise manner.	 *Rigid with reflections*	Animate the blue circle to move along a path shaped like an astroid (four-cusped hypocycloid). The 'a' parameter of the astroid should be 60.
Shape family	**Text prompt 28**	**Shape family**	**Text prompt 23**	**Shape family**	**Text prompt 25**	**Shape family**	**Text prompt 27**
 *Rigid with reflections*	Animate the blue circle to move along a path shaped like a deltoid (three-cusped hypocycloid). The 'a' parameter of the deltoid should be 60.	 *Rigid*	Animate the blue circle to move along a path shaped like a circle with a radius of 70. Traverse the path in a clockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like a circle. Traverse the path in a clockwise manner.	 *Rigid with reflections*	Animate the blue circle to move along a path shaped like an astroid (four-cusped hypocycloid). The 'a' parameter of the astroid should be 80.
Shape family	**Text prompt 29**	**Shape family**	**Text prompt 30**	**Shape family**	**Text prompt 31**	**Shape family**	**Text prompt 32**
 *Rigid with reflections*	Animate the blue circle to move along a path shaped like a deltoid (three-cusped hypocycloid). The 'a' parameter of the deltoid should be 80.	 *Similarity with reflections*	Animate the blue circle to move along a path shaped like an astroid (four-cusped hypocycloid).	 *Similarity with reflections*	Animate the blue circle to move along a path shaped like a deltoid (three-cusped hypocycloid).	 *Similarity*	Animate the blue circle to move along a path shaped like an astroid (four-cusped hypocycloid). Traverse the path in a counterclockwise manner.
Shape family	**Text prompt 33**	**Shape family**	**Text prompt 34**	**Shape family**	**Text prompt 35**	**Shape family**	**Text prompt 36**
 *Similarity*	Animate the blue circle to move along a path shaped like an astroid (four-cusped hypocycloid). Traverse the path in a clockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like a deltoid (three-cusped hypocycloid). Traverse the path in a counterclockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like a deltoid (three-cusped hypocycloid). Traverse the path in a clockwise manner.	 *Affine without skew*	Animate the blue circle to move along a path shaped like an isosceles triangle.

Figure S11. Motion trajectory benchmark: prompt 1–36 with ground truth shape families.

Shape family

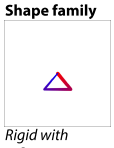


Affine without skew

Text prompt 37

Animate the blue circle to move along a path shaped like a right triangle.

Shape family



Rigid with reflections

Text prompt 38

Animate the blue circle to move along a path shaped like an isosceles triangle with a width of 100 and a height of 60.

Shape family

Rigid with reflections

Text prompt 39

Animate the blue circle to move along a path shaped like an isosceles triangle with a width of 100 and a height of 80.

Shape family

Rigid with reflections

Text prompt 40

Animate the blue circle to move along a path shaped like an isosceles triangle with a width of 50 and a height of 60.

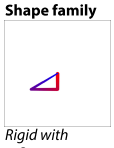
Shape family

Rigid with reflections

Text prompt 41

Animate the blue circle to move along a path shaped like an isosceles triangle with a width of 50 and a height of 80.

Shape family

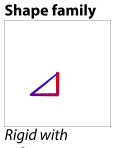


Rigid with reflections

Text prompt 42

Animate the blue circle to move along a path shaped like a right triangle with a width of 100 and a height of 60.

Shape family

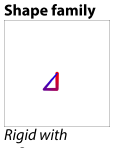


Rigid with reflections

Text prompt 43

Animate the blue circle to move along a path shaped like a right triangle with a width of 100 and a height of 80.

Shape family



Rigid with reflections

Text prompt 44

Animate the blue circle to move along a path shaped like a right triangle with a width of 50 and a height of 60.

Shape family

Rigid with reflections

Text prompt 45

Animate the blue circle to move along a path shaped like a right triangle with a width of 50 and a height of 80.

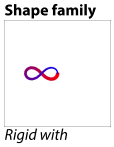
Shape family

Affine

Text prompt 46

Animate the blue circle to move along a path shaped like a L shape (two sides of a triangle).

Shape family

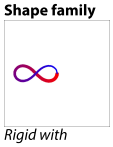


Rigid with reflections

Text prompt 47

Animate the blue circle to move along a path shaped like a figure 8 shape (lemniscate of Bernoulli). The 'a' parameter of the figure 8 shape should be 60.

Shape family

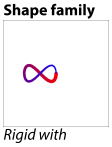


Rigid with reflections

Text prompt 48

Animate the blue circle to move along a path shaped like a figure 8 shape (lemniscate of Bernoulli). The 'a' parameter of the figure 8 shape should be 80.

Shape family

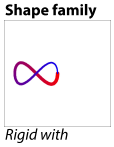


Rigid with reflections

Text prompt 49

Animate the blue circle to move along a path shaped like a figure 8 shape (lemniscate of Gerono). The 'a' parameter of the figure 8 shape should be 60.

Shape family

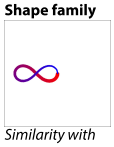


Rigid with reflections

Text prompt 50

Animate the blue circle to move along a path shaped like a figure 8 shape (lemniscate of Gerono). The 'a' parameter of the figure 8 shape should be 80.

Shape family

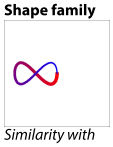


Similarity with reflections

Text prompt 51

Animate the blue circle to move along a path shaped like a figure 8 shape (lemniscate of Bernoulli).

Shape family

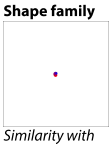


Similarity with reflections

Text prompt 52

Animate the blue circle to move along a path shaped like a figure 8 shape (lemniscate of Gerono).

Shape family

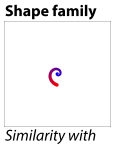


Similarity with reflections

Text prompt 53

Animate the blue circle to move along a path shaped like a logarithmic spiral (initial radius should be 1 and growth rate should be 0.3. move outward from the scene center). Complete 1 turn of the logarithmic spiral.

Shape family

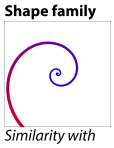


Similarity with reflections

Text prompt 54

Animate the blue circle to move along a path shaped like a logarithmic spiral (initial radius should be 1 and growth rate should be 0.3. move outward from the scene center). Complete 2 turms of the logarithmic spiral.

Shape family

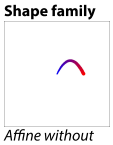


Similarity with reflections

Text prompt 55

Animate the blue circle to move along a path shaped like a logarithmic spiral (initial radius should be 1 and growth rate should be 0.3. move outward from the scene center). Complete 3 turms of the logarithmic spiral.

Shape family



Affine without skew

Text prompt 56

Animate the blue circle to move along a path shaped like a full, symmetric parabola.

Shape family

Rigid with reflections

Text prompt 57

Animate the blue circle to move along a path shaped like a full, symmetric parabola with a height of 60. The horizontal extent of the full, symmetric parabola should be 100.

Shape family

Rigid with reflections

Text prompt 58

Animate the blue circle to move along a path shaped like a full, symmetric parabola with a height of 80. The horizontal extent of the full, symmetric parabola should be 100.

Shape family

Rigid with reflections

Text prompt 59

Animate the blue circle to move along a path shaped like a full, symmetric parabola with a height of 60. The horizontal extent of the full, symmetric parabola should be 150.

Shape family

Rigid with reflections

Text prompt 60

Animate the blue circle to move along a path shaped like a full, symmetric parabola with a height of 80. The horizontal extent of the full, symmetric parabola should be 150.

Shape family

Affine

Text prompt 61

Animate the blue circle to move along a path shaped like a parallelogram.

Shape family

Rigid with reflections

Text prompt 62

Animate the blue circle to move along a path shaped like an equilateral triangle with a side length of 100.

Shape family

Rigid with reflections

Text prompt 63

Animate the blue circle to move along a path shaped like an equilateral triangle with a side length of 150.

Shape family

Rigid with reflections

Text prompt 64

Animate the blue circle to move along a path shaped like an equilateral triangle with a side length of 175.

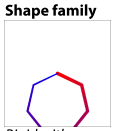
Shape family

Rigid with reflections

Text prompt 65

Animate the blue circle to move along a path shaped like an equilateral triangle with a side length of 50.

Shape family

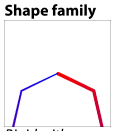


Rigid with reflections

Text prompt 66

Animate the blue circle to move along a path shaped like a regular heptagon with a side length of 100.

Shape family

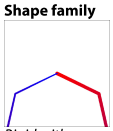


Rigid with reflections

Text prompt 67

Animate the blue circle to move along a path shaped like a regular heptagon with a side length of 150.

Shape family



Rigid with reflections

Text prompt 68

Animate the blue circle to move along a path shaped like a regular heptagon with a side length of 175.

Shape family

Rigid with reflections

Text prompt 69

Animate the blue circle to move along a path shaped like a regular heptagon with a side length of 50.

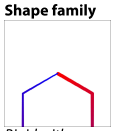
Shape family

Rigid with reflections

Text prompt 70

Animate the blue circle to move along a path shaped like a regular hexagon with a side length of 100.

Shape family

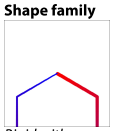


Rigid with reflections

Text prompt 71

Animate the blue circle to move along a path shaped like a regular hexagon with a side length of 150.

Shape family



Rigid with reflections

Text prompt 72

Animate the blue circle to move along a path shaped like a regular hexagon with a side length of 175.

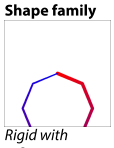
Figure S12. Motion trajectory benchmark: prompt 37–72 with ground truth shape families.

Shape family

Rigid with reflections

Text prompt 73
Animate the blue circle to move along a path shaped like a regular hexagon with a side length of 50.

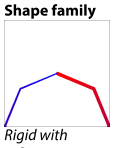
Shape family



Rigid with reflections

Text prompt 74
Animate the blue circle to move along a path shaped like a regular octagon with a side length of 100.

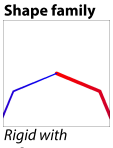
Shape family



Rigid with reflections

Text prompt 75
Animate the blue circle to move along a path shaped like a regular octagon with a side length of 150.

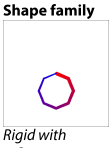
Shape family



Rigid with reflections

Text prompt 76
Animate the blue circle to move along a path shaped like a regular octagon with a side length of 175.

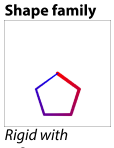
Shape family



Rigid with reflections

Text prompt 77
Animate the blue circle to move along a path shaped like a regular octagon with a side length of 50.

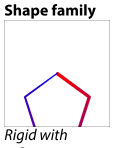
Shape family



Rigid with reflections

Text prompt 78
Animate the blue circle to move along a path shaped like a regular pentagon with a side length of 100.

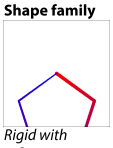
Shape family



Rigid with reflections

Text prompt 79
Animate the blue circle to move along a path shaped like a regular pentagon with a side length of 150.

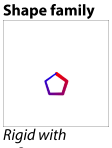
Shape family



Rigid with reflections

Text prompt 80
Animate the blue circle to move along a path shaped like a regular pentagon with a side length of 175.

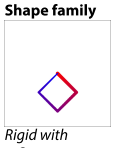
Shape family



Rigid with reflections

Text prompt 81
Animate the blue circle to move along a path shaped like a regular pentagon with a side length of 50.

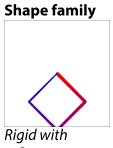
Shape family



Rigid with reflections

Text prompt 82
Animate the blue circle to move along a path shaped like a square with a side length of 100.

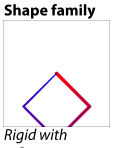
Shape family



Rigid with reflections

Text prompt 83
Animate the blue circle to move along a path shaped like a square with a side length of 150.

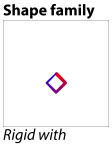
Shape family



Rigid with reflections

Text prompt 84
Animate the blue circle to move along a path shaped like a square with a side length of 175.

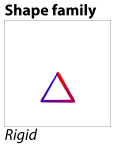
Shape family



Rigid with reflections

Text prompt 85
Animate the blue circle to move along a path shaped like a square with a side length of 50.

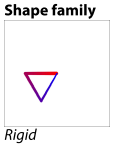
Shape family



Rigid

Text prompt 86
Animate the blue circle to move along a path shaped like an equilateral triangle with a side length of 120. Traverse the path in a counterclockwise manner.

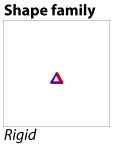
Shape family



Rigid

Text prompt 87
Animate the blue circle to move along a path shaped like an equilateral triangle with a side length of 120. Traverse the path in a clockwise manner.

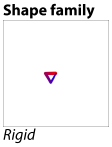
Shape family



Rigid

Text prompt 88
Animate the blue circle to move along a path shaped like an equilateral triangle with a side length of 40. Traverse the path in a counterclockwise manner.

Shape family



Rigid

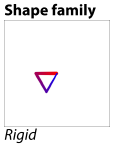
Text prompt 89
Animate the blue circle to move along a path shaped like an equilateral triangle with a side length of 40. Traverse the path in a clockwise manner.

Shape family

Rigid

Text prompt 90
Animate the blue circle to move along a path shaped like an equilateral triangle with a side length of 80. Traverse the path in a counterclockwise manner.

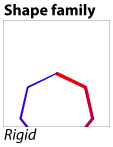
Shape family



Rigid

Text prompt 91
Animate the blue circle to move along a path shaped like an equilateral triangle with a side length of 80. Traverse the path in a clockwise manner.

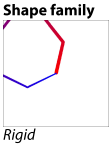
Shape family



Rigid

Text prompt 92
Animate the blue circle to move along a path shaped like a regular heptagon with a side length of 120. Traverse the path in a counterclockwise manner.

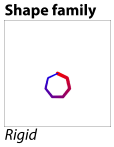
Shape family



Rigid

Text prompt 93
Animate the blue circle to move along a path shaped like a regular heptagon with a side length of 120. Traverse the path in a clockwise manner.

Shape family



Rigid

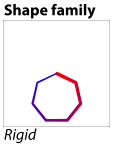
Text prompt 94
Animate the blue circle to move along a path shaped like a regular heptagon with a side length of 40. Traverse the path in a counterclockwise manner.

Shape family

Rigid

Text prompt 95
Animate the blue circle to move along a path shaped like a regular heptagon with a side length of 40. Traverse the path in a clockwise manner.

Shape family



Rigid

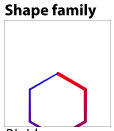
Text prompt 96
Animate the blue circle to move along a path shaped like a regular heptagon with a side length of 80. Traverse the path in a counterclockwise manner.

Shape family

Rigid

Text prompt 97
Animate the blue circle to move along a path shaped like a regular heptagon with a side length of 80. Traverse the path in a clockwise manner.

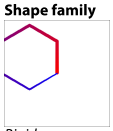
Shape family



Rigid

Text prompt 98
Animate the blue circle to move along a path shaped like a regular hexagon with a side length of 120. Traverse the path in a counterclockwise manner.

Shape family



Rigid

Text prompt 99
Animate the blue circle to move along a path shaped like a regular hexagon with a side length of 120. Traverse the path in a clockwise manner.

Shape family

Rigid

Text prompt 100
Animate the blue circle to move along a path shaped like a regular hexagon with a side length of 40. Traverse the path in a counterclockwise manner.

Shape family

Rigid

Text prompt 101
Animate the blue circle to move along a path shaped like a regular hexagon with a side length of 40. Traverse the path in a clockwise manner.

Shape family

Rigid

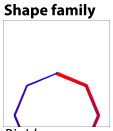
Text prompt 102
Animate the blue circle to move along a path shaped like a regular hexagon with a side length of 80. Traverse the path in a counterclockwise manner.

Shape family

Rigid

Text prompt 103
Animate the blue circle to move along a path shaped like a regular hexagon with a side length of 80. Traverse the path in a clockwise manner.

Shape family



Rigid

Text prompt 104
Animate the blue circle to move along a path shaped like a regular octagon with a side length of 120. Traverse the path in a counterclockwise manner.

Shape family

Rigid

Text prompt 105
Animate the blue circle to move along a path shaped like a regular octagon with a side length of 120. Traverse the path in a clockwise manner.

Shape family

Rigid

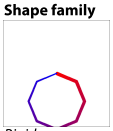
Text prompt 106
Animate the blue circle to move along a path shaped like a regular octagon with a side length of 40. Traverse the path in a counterclockwise manner.

Shape family

Rigid

Text prompt 107
Animate the blue circle to move along a path shaped like a regular octagon with a side length of 40. Traverse the path in a clockwise manner.

Shape family



Rigid

Text prompt 108
Animate the blue circle to move along a path shaped like a regular octagon with a side length of 80. Traverse the path in a counterclockwise manner.

Figure S13. Motion trajectory benchmark: prompt 73–108 with ground truth shape families.

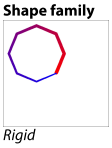
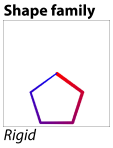
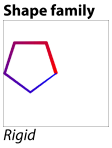
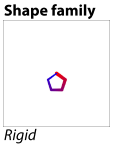
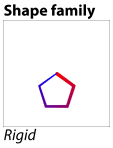
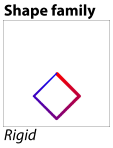
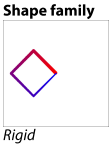
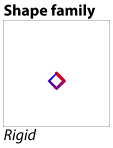
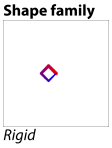
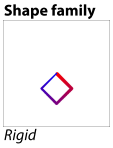
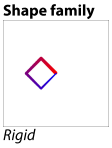
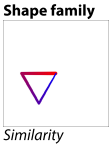
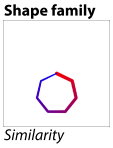
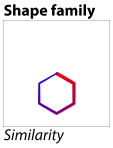
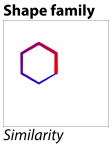
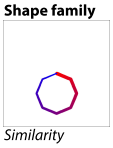
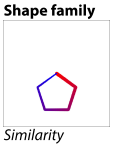
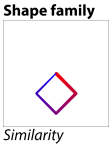
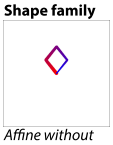
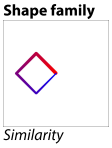
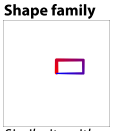
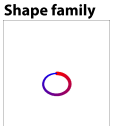
Shape family	Text prompt 109	Shape family	Text prompt 110	Shape family	Text prompt 111	Shape family	Text prompt 112
 *Rigid*	Animate the blue circle to move along a path shaped like a regular octagon with a side length of 80. Traverse the path in a clockwise manner.	 *Rigid*	Animate the blue circle to move along a path shaped like a regular pentagon with a side length of 120. Traverse the path in a counterclockwise manner.	 *Rigid*	Animate the blue circle to move along a path shaped like a regular pentagon with a side length of 120. Traverse the path in a clockwise manner.	 *Rigid*	Animate the blue circle to move along a path shaped like a regular pentagon with a side length of 40. Traverse the path in a counterclockwise manner.
Shape family	**Text prompt 113**	**Shape family**	**Text prompt 114**	**Shape family**	**Text prompt 115**	**Shape family**	**Text prompt 116**
 *Rigid*	Animate the blue circle to move along a path shaped like a regular pentagon with a side length of 40. Traverse the path in a clockwise manner.	 *Rigid*	Animate the blue circle to move along a path shaped like a regular pentagon with a side length of 80. Traverse the path in a counterclockwise manner.	 *Rigid*	Animate the blue circle to move along a path shaped like a regular pentagon with a side length of 80. Traverse the path in a clockwise manner.	 *Rigid*	Animate the blue circle to move along a path shaped like a square with a side length of 120. Traverse the path in a counterclockwise manner.
Shape family	**Text prompt 117**	**Shape family**	**Text prompt 118**	**Shape family**	**Text prompt 119**	**Shape family**	**Text prompt 120**
 *Rigid*	Animate the blue circle to move along a path shaped like a square with a side length of 120. Traverse the path in a clockwise manner.	 *Rigid*	Animate the blue circle to move along a path shaped like a square with a side length of 40. Traverse the path in a counterclockwise manner.	 *Rigid*	Animate the blue circle to move along a path shaped like a square with a side length of 40. Traverse the path in a clockwise manner.	 *Rigid*	Animate the blue circle to move along a path shaped like a square with a side length of 80. Traverse the path in a counterclockwise manner.
Shape family	**Text prompt 121**	**Shape family**	**Text prompt 122**	**Shape family**	**Text prompt 123**	**Shape family**	**Text prompt 124**
 *Rigid*	Animate the blue circle to move along a path shaped like a square with a side length of 80. Traverse the path in a clockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like an equilateral triangle. Traverse the path in a counterclockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like an equilateral triangle. Traverse the path in a clockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like a regular heptagon. Traverse the path in a counterclockwise manner.
Shape family	**Text prompt 125**	**Shape family**	**Text prompt 126**	**Shape family**	**Text prompt 127**	**Shape family**	**Text prompt 128**
 *Similarity*	Animate the blue circle to move along a path shaped like a regular heptagon. Traverse the path in a clockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like a regular hexagon. Traverse the path in a counterclockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like a regular hexagon. Traverse the path in a clockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like a regular octagon. Traverse the path in a counterclockwise manner.
Shape family	**Text prompt 129**	**Shape family**	**Text prompt 130**	**Shape family**	**Text prompt 132**	**Shape family**	**Text prompt 134**
 *Similarity*	Animate the blue circle to move along a path shaped like a regular octagon. Traverse the path in a clockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like a regular pentagon. Traverse the path in a counterclockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like a square. Traverse the path in a counterclockwise manner.	 *Affine without skew*	Animate the blue circle to move along a path shaped like a rhombus.
Shape family	**Text prompt 136**	**Shape family**	**Text prompt 131**	**Shape family**	**Text prompt 133**	**Shape family**	**Text prompt 135**
 *Similarity with reflections*	Animate the blue circle to move along a path shaped like an ellipse with a 16:9 width to height ratio.	 *Similarity*	Animate the blue circle to move along a path shaped like a regular pentagon. Traverse the path in a clockwise manner.	 *Similarity*	Animate the blue circle to move along a path shaped like a square. Traverse the path in a clockwise manner.	 *Affine without skew*	Animate the blue circle to move along a path shaped like an ellipse.
Shape family	**Text prompt 137**	**Shape family**	**Text prompt 138**	**Shape family**	**Text prompt 139**	**Shape family**	**Text prompt 140**
 *Similarity with reflections*	Animate the blue circle to move along a path shaped like an ellipse with a 2:1 width to height ratio.	 *Similarity with reflections*	Animate the blue circle to move along a path shaped like an ellipse with a 4:3 width to height ratio.	 *Similarity with reflections*	Animate the blue circle to move along a path shaped like a rectangle with a 16:9 width to height ratio.	 *Similarity with reflections*	Animate the blue circle to move along a path shaped like a rectangle with a 2:1 width to height ratio.
Shape family	**Text prompt 141**	**Shape family**	**Text prompt 142**	**Shape family**	**Text prompt 143**	**Shape family**	**Text prompt 144**
 *Similarity with reflections*	Animate the blue circle to move along a path shaped like a rectangle with a 4:3 width to height ratio.	 *Affine without skew*	Animate the blue circle to move along a path shaped like a rectangle.	 *Rigid with reflections*	Animate the blue circle to move along a path shaped like an ellipse with a width of 100 and a height of 60.	 *Rigid with reflections*	Animate the blue circle to move along a path shaped like an ellipse with a width of 100 and a height of 80.

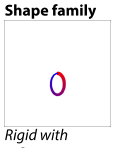
Figure S14. Motion trajectory benchmark: prompt 109–144 with ground truth shape families.

Shape family

Rigid with reflections

Text prompt 145
Animate the blue circle to move along a path shaped like an ellipse with a width of 50 and a height of 60.

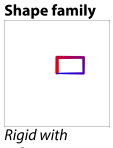
Shape family



Rigid with reflections

Text prompt 146
Animate the blue circle to move along a path shaped like an ellipse with a width of 50 and a height of 80.

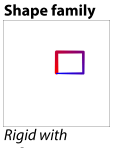
Shape family



Rigid with reflections

Text prompt 147
Animate the blue circle to move along a path shaped like a rectangle with a width of 100 and a height of 60.

Shape family



Rigid with reflections

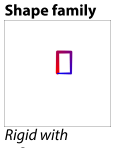
Text prompt 148
Animate the blue circle to move along a path shaped like a rectangle with a width of 100 and a height of 80.

Shape family

Rigid with reflections

Text prompt 149
Animate the blue circle to move along a path shaped like a rectangle with a width of 50 and a height of 60.

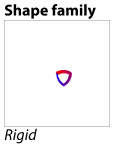
Shape family



Rigid with reflections

Text prompt 150
Animate the blue circle to move along a path shaped like a rectangle with a width of 50 and a height of 80.

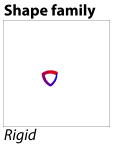
Shape family



Rigid

Text prompt 151
Animate the blue circle to move along a path shaped like a reuleaux triangle with a width of 50. Traverse the path in a counterclockwise manner.

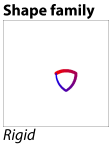
Shape family



Rigid

Text prompt 152
Animate the blue circle to move along a path shaped like a reuleaux triangle with a width of 50. Traverse the path in a clockwise manner.

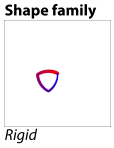
Shape family



Rigid

Text prompt 153
Animate the blue circle to move along a path shaped like a reuleaux triangle with a width of 75. Traverse the path in a counterclockwise manner.

Shape family



Rigid

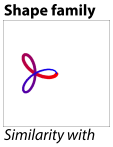
Text prompt 154
Animate the blue circle to move along a path shaped like a reuleaux triangle with a width of 75. Traverse the path in a clockwise manner.

Shape family

Similarity with reflections

Text prompt 155
Animate the blue circle to move along a path shaped like a reuleaux triangle.

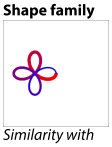
Shape family



Similarity with reflections

Text prompt 156
Animate the blue circle to move along a path shaped like a 3-petal rose (trifolium).

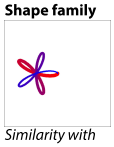
Shape family



Similarity with reflections

Text prompt 157
Animate the blue circle to move along a path shaped like a 4-petal rose (quadrifolium).

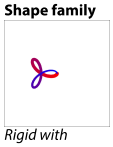
Shape family



Similarity with reflections

Text prompt 158
Animate the blue circle to move along a path shaped like a 5-petal rose (pentafolium).

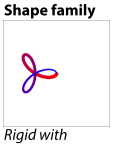
Shape family



Rigid with reflections

Text prompt 159
Animate the blue circle to move along a path shaped like a 3-petal rose (trifolium). The 'a' parameter of the 3-petal rose should be 60.

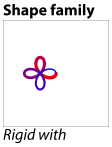
Shape family



Rigid with reflections

Text prompt 160
Animate the blue circle to move along a path shaped like a 3-petal rose (trifolium). The 'a' parameter of the 3-petal rose should be 80.

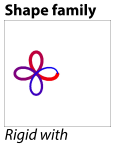
Shape family



Rigid with reflections

Text prompt 161
Animate the blue circle to move along a path shaped like a 4-petal rose (quadrifolium). The 'a' parameter of the 4-petal rose should be 60.

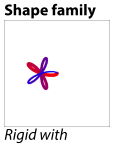
Shape family



Rigid with reflections

Text prompt 162
Animate the blue circle to move along a path shaped like a 4-petal rose (quadrifolium). The 'a' parameter of the 4-petal rose should be 80.

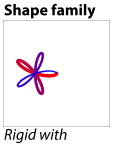
Shape family



Rigid with reflections

Text prompt 163
Animate the blue circle to move along a path shaped like a 5-petal rose (pentafolium). The 'a' parameter of the 5-petal rose should be 60.

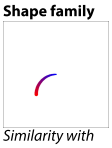
Shape family



Rigid with reflections

Text prompt 164
Animate the blue circle to move along a path shaped like a 5-petal rose (pentafolium). The 'a' parameter of the 5-petal rose should be 80.

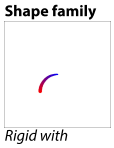
Shape family



Similarity with reflections

Text prompt 165
Animate the blue circle to move along a path shaped like a quarter-circle (a quarter of a circle's circumference).

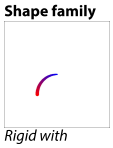
Shape family



Rigid with reflections

Text prompt 166
Animate the blue circle to move along a path shaped like a quarter-circle (a quarter of a circle's circumference) with a radius of 65.

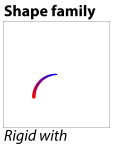
Shape family



Rigid with reflections

Text prompt 167
Animate the blue circle to move along a path shaped like a quarter-circle (a quarter of a circle's circumference) with a radius of 75.

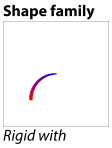
Shape family



Rigid with reflections

Text prompt 168
Animate the blue circle to move along a path shaped like a quarter-circle (a quarter of a circle's circumference) with a radius of 85.

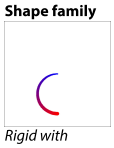
Shape family



Rigid with reflections

Text prompt 169
Animate the blue circle to move along a path shaped like a quarter-circle (a quarter of a circle's circumference) with a radius of 95.

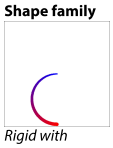
Shape family



Rigid with reflections

Text prompt 171
Animate the blue circle to move along a path shaped like a semi-circle (half of a circle's circumference) with a radius of 75.

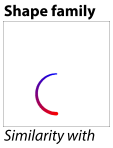
Shape family



Rigid with reflections

Text prompt 173
Animate the blue circle to move along a path shaped like a semi-circle (half of a circle's circumference) with a radius of 95.

Shape family



Similarity with reflections

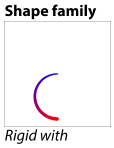
Text prompt 174
Animate the blue circle to move along a path shaped like a semi-circle (half of a circle's circumference).

Shape family

Rigid with reflections

Text prompt 170
Animate the blue circle to move along a path shaped like a semi-circle (half of a circle's circumference) with a radius of 65.

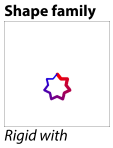
Shape family



Rigid with reflections

Text prompt 172
Animate the blue circle to move along a path shaped like a semi-circle (half of a circle's circumference) with a radius of 85.

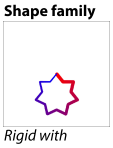
Shape family



Rigid with reflections

Text prompt 175
Animate the blue circle to move along a path shaped like a regular heptagram {7/2} (without the internal crossovers) with an outer radius of 50. Only move along the outline of the regular heptagram {7/2} (no path crossovers).

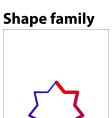
Shape family



Rigid with reflections

Text prompt 176
Animate the blue circle to move along a path shaped like a regular heptagram {7/2} (without the internal crossovers) with an outer radius of 80. Only move along the outline of the regular heptagram {7/2} (no path crossovers).

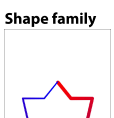
Shape family



Rigid with reflections

Text prompt 177
Animate the blue circle to move along a path shaped like a regular heptagram {7/2} (without the internal crossovers) with each line segment being 50 px long. Only move along the outline of the regular heptagram {7/2} (no path crossovers).

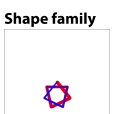
Shape family



Rigid with reflections

Text prompt 178
Animate the blue circle to move along a path shaped like a regular heptagram {7/2} (without the internal crossovers) with each line segment being 80 px long. Only move along the outline of the regular heptagram {7/2} (no path crossovers).

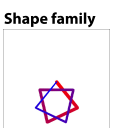
Shape family



Rigid with reflections

Text prompt 179
Animate the blue circle to move along a path shaped like a regular heptagram {7/2} (with self-intersecting path segments) with an outer radius of 50.

Shape family

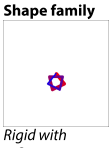


Rigid with reflections

Text prompt 180
Animate the blue circle to move along a path shaped like a regular heptagram {7/2} (with self-intersecting path segments) with an outer radius of 80.

Figure S15. Motion trajectory benchmark: prompt 145–180 with ground truth shape families.

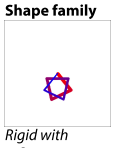
Shape family



Rigid with reflections

Text prompt 181
Animate the blue circle to move along a path shaped like a regular heptagram {7/2} (with self-intersecting path segments) with each line segment being 50 px long.

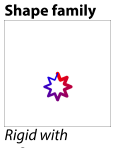
Shape family



Rigid with reflections

Text prompt 182
Animate the blue circle to move along a path shaped like a regular heptagram {7/2} (with self-intersecting path segments) with each line segment being 80 px long.

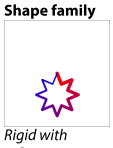
Shape family



Rigid with reflections

Text prompt 183
Animate the blue circle to move along a path shaped like a regular octagram {8/3} (without the internal crossovers) with an outer radius of 50. Only move along the outline of the regular octagram {8/3} (no path crossovers).

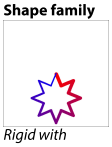
Shape family



Rigid with reflections

Text prompt 184
Animate the blue circle to move along a path shaped like a regular octagram {8/3} (without the internal crossovers) with an outer radius of 80. Only move along the outline of the regular octagram {8/3} (no path crossovers).

Shape family



Rigid with reflections

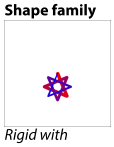
Text prompt 185
Animate the blue circle to move along a path shaped like a regular octagram {8/3} (without the internal crossovers) with each line segment being 50 px long. Only move along the outline of the regular octagram {8/3} (no path crossovers).

Shape family

Rigid with reflections

Text prompt 186
Animate the blue circle to move along a path shaped like a regular octagram {8/3} (without the internal crossovers) with each line segment being 80 px long. Only move along the outline of the regular octagram {8/3} (no path crossovers).

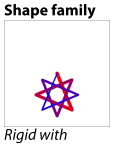
Shape family



Rigid with reflections

Text prompt 187
Animate the blue circle to move along a path shaped like a regular octagram {8/3} (with self-intersecting path segments) with an outer radius of 50.

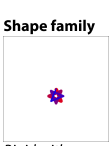
Shape family



Rigid with reflections

Text prompt 188
Animate the blue circle to move along a path shaped like a regular octagram {8/3} (with self-intersecting path segments) with an outer radius of 80.

Shape family



Rigid with reflections

Text prompt 189
Animate the blue circle to move along a path shaped like a regular octagram {8/3} (with self-intersecting path segments) with each line segment being 50 px long.

Shape family

Rigid with reflections

Text prompt 191
Animate the blue circle to move along a path shaped like a regular pentagram (without the internal crossovers) with an outer radius of 50. Only move along the outline of the regular pentagram (no path crossovers).

Shape family

Rigid with reflections

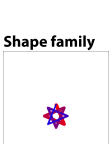
Text prompt 193
Animate the blue circle to move along a path shaped like a regular pentagram (without the internal crossovers) with each line segment being 50 px long. Only move along the outline of the regular pentagram (no path crossovers).

Shape family

Rigid with reflections

Text prompt 194
Animate the blue circle to move along a path shaped like a regular pentagram (without the internal crossovers) with each line segment being 80 px long. Only move along the outline of the regular pentagram (no path crossovers).

Shape family



Rigid with reflections

Text prompt 190
Animate the blue circle to move along a path shaped like a regular octagram {8/3} (with self-intersecting path segments) with each line segment being 80 px long.

Shape family

Rigid with reflections

Text prompt 192
Animate the blue circle to move along a path shaped like a regular pentagram (without the internal crossovers) with an outer radius of 80. Only move along the outline of the regular pentagram (no path crossovers).

Shape family

Rigid with reflections

Text prompt 195
Animate the blue circle to move along a path shaped like a regular pentagram (with self-intersecting path segments) with an outer radius of 50.

Shape family

Rigid with reflections

Text prompt 196
Animate the blue circle to move along a path shaped like a regular pentagram (with self-intersecting path segments) with an outer radius of 80.

Shape family

Rigid with reflections

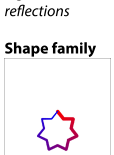
Text prompt 197
Animate the blue circle to move along a path shaped like a regular pentagram (with self-intersecting path segments) with each line segment being 50 px long.

Shape family

Rigid with reflections

Text prompt 198
Animate the blue circle to move along a path shaped like a regular pentagram (with self-intersecting path segments) with each line segment being 80 px long.

Shape family



Similarity

Text prompt 199
Animate the blue circle to move along a path shaped like a regular heptagram {7/2} (without the internal crossovers). Only move along the outline of the regular heptagram {7/2} (no path crossovers). Traverse the path in a counterclockwise manner.

Shape family

Similarity

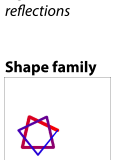
Text prompt 200
Animate the blue circle to move along a path shaped like a regular heptagram {7/2} (without the internal crossovers). Only move along the outline of the regular heptagram {7/2} (no path crossovers). Traverse the path in a clockwise manner.

Shape family

Similarity

Text prompt 201
Animate the blue circle to move along a path shaped like a regular heptagram {7/2} (with self-intersecting path segments). Traverse the path in a counterclockwise manner.

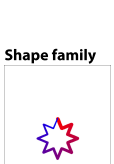
Shape family



Similarity

Text prompt 202
Animate the blue circle to move along a path shaped like a regular heptagram {7/2} (with self-intersecting path segments). Traverse the path in a clockwise manner.

Shape family



Similarity

Text prompt 203
Animate the blue circle to move along a path shaped like a regular octagram {8/3} (without the internal crossovers). Only move along the outline of the regular octagram {8/3} (no path crossovers). Traverse the path in a counterclockwise manner.

Shape family

Similarity

Text prompt 204
Animate the blue circle to move along a path shaped like a regular octagram {8/3} (without the internal crossovers). Only move along the outline of the regular octagram {8/3} (no path crossovers). Traverse the path in a clockwise manner.

Shape family

Similarity

Text prompt 205
Animate the blue circle to move along a path shaped like a regular octagram {8/3} (with self-intersecting path segments). Traverse the path in a counterclockwise manner.

Shape family

Similarity

Text prompt 206
Animate the blue circle to move along a path shaped like a regular octagram {8/3} (with self-intersecting path segments). Traverse the path in a clockwise manner.

Shape family

Similarity

Text prompt 207
Animate the blue circle to move along a path shaped like a regular pentagram (without the internal crossovers). Only move along the outline of the regular pentagram (no path crossovers). Traverse the path in a counterclockwise manner.

Shape family

Similarity

Text prompt 208
Animate the blue circle to move along a path shaped like a regular pentagram (without the internal crossovers). Only move along the outline of the regular pentagram (no path crossovers). Traverse the path in a clockwise manner.

Shape family

Similarity

Text prompt 209
Animate the blue circle to move along a path shaped like a regular pentagram (with self-intersecting path segments). Traverse the path in a counterclockwise manner.

Shape family

Similarity

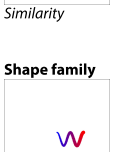
Text prompt 210
Animate the blue circle to move along a path shaped like a regular pentagram (with self-intersecting path segments). Traverse the path in a clockwise manner.

Shape family

Affine without skew

Text prompt 211
Animate the blue circle to move along a path shaped like a cosine wave. Complete 1 period of the cosine wave.

Shape family



Affine without skew

Text prompt 212
Animate the blue circle to move along a path shaped like a cosine wave. Complete 2 periods of the cosine wave.

Shape family

Affine without skew

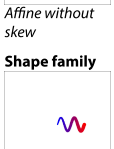
Text prompt 213
Animate the blue circle to move along a path shaped like a cosine wave. Complete 3 periods of the cosine wave.

Shape family

Affine without skew

Text prompt 214
Animate the blue circle to move along a path shaped like a sine wave. Complete 1 period of the sine wave.

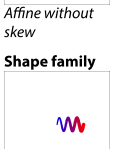
Shape family



Affine without skew

Text prompt 215
Animate the blue circle to move along a path shaped like a sine wave. Complete 2 periods of the sine wave.

Shape family



Affine without skew

Text prompt 216
Animate the blue circle to move along a path shaped like a sine wave. Complete 3 periods of the sine wave.

Figure S16. Motion trajectory benchmark: prompt 181–216 with ground truth shape families.

Shape family

Rigid with reflections

Text prompt 217
Animate the blue circle to move along a path shaped like a cosine wave with an amplitude of 50. Complete 2 periods of the cosine wave. The horizontal extent of the cosine wave should be 150.

Shape family

Rigid with reflections

Text prompt 218
Animate the blue circle to move along a path shaped like a cosine wave with an amplitude of 75. Complete 2 periods of the cosine wave. The horizontal extent of the cosine wave should be 150.

Shape family

Rigid with reflections

Text prompt 219
Animate the blue circle to move along a path shaped like a cosine wave with an amplitude of 50. Complete 2 periods of the cosine wave. The horizontal extent of the cosine wave should be 75.

Shape family

Rigid with reflections

Text prompt 220
Animate the blue circle to move along a path shaped like a cosine wave with an amplitude of 75. Complete 2 periods of the cosine wave. The horizontal extent of the cosine wave should be 75.

Shape family

Rigid with reflections

Text prompt 221
Animate the blue circle to move along a path shaped like a sine wave with an amplitude of 50. Complete 2 periods of the sine wave. The horizontal extent of the sine wave should be 150.

Shape family

Rigid with reflections

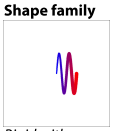
Text prompt 222
Animate the blue circle to move along a path shaped like a sine wave with an amplitude of 75. Complete 2 periods of the sine wave. The horizontal extent of the sine wave should be 150.

Shape family

Rigid with reflections

Text prompt 223
Animate the blue circle to move along a path shaped like a sine wave with an amplitude of 50. Complete 2 periods of the sine wave. The horizontal extent of the sine wave should be 75.

Shape family



Rigid with reflections

Text prompt 224
Animate the blue circle to move along a path shaped like a sine wave with an amplitude of 75. Complete 2 periods of the sine wave. The horizontal extent of the sine wave should be 75.

Figure S17. Motion trajectory benchmark: prompt 217–224 with ground truth shape families.