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

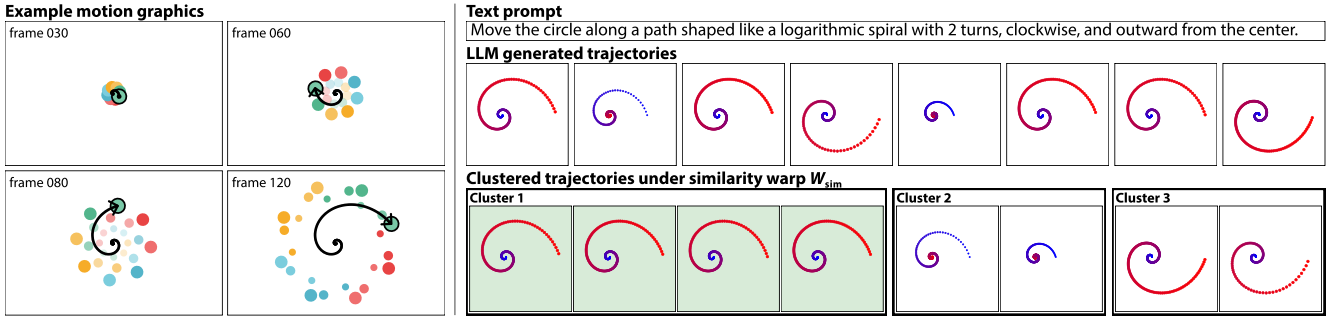

Figure 1. Complex motion graphics animations are often composed of trajectories in the form of geometric shapes (left). While LLMs can generate motion graphics animations from a prompt describing the shape of an object's trajectory, the resulting animation does not always follow the prompt specification (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

Self-consistency has proven to be an effective technique for improving LLM performance on natural language reasoning tasks in a lightweight, unsupervised manner. In this work, we study how to adapt self-consistency to visual domains. Specifically, we consider the generation and verification of LLM-produced motion graphics trajectories. Given a prompt (e.g., "Move the circle in a spiral path"), we first sample diverse motion trajectories from an LLM, and then identify groups of consistent trajectories via clustering. Our key insight is to model the family of shapes associated with a prompt as a prototype trajectory paired with a group of geometric transformations (e.g., rigid, similarity, and affine). Two trajectories can then be considered consistent if one can be transformed into the other under the warps allowable by the transformation group. We propose an algorithm that automatically recovers a shape family, using hierarchical relationships between a set of candidate transformation groups. Our approach improves the accuracy of LLM-based trajectory generation by 4–6%. We further extend our method to support verification, observing 11% precision gains over VLM baselines. Our code and dataset are available at https://majiaju.io/trajectory-self-consistency.

1. Introduction

Self-consistency has recently emerged as an effective method for improving the performance of LLMs on tasks with natural language outputs [28]. At a high-level, the approach samples an LLM to produce diverse responses, and then groups responses to find the most consistent answer. Consider a math problem like "A train travels 45 miles per hour. How far does it travel in 3 hours?" Self-consistency would use an LLM to sample diverse chain-of-thought reasoning traces, identify the final numerical value produced by each response, and return the most common number as the answer. This paradigm makes two base assumptions: (i) that the LLM is likely to produce the correct response more often than any other incorrect response; (ii) that it is possible to identify when answers are consistent with one another. Self-consistency is a useful paradigm, because it offers an unsupervised, training-free method for producing better generations from generative AI systems.

Beyond text domains, LLMs have also proven effective as generative systems for visual outputs, often via program synthesis. These models take a text prompt as input and generate a program (often in a domain specific language) that is then executed to render a visual image [10], vector graphics [20, 29], 3D scenes [12, 31], or animation [9, 18, 19, 27]. This leads to a question: how can we

apply self-consistency approaches in the context of LLM-based visual generation? If we want to identify the most common answer from a set of LLM responses, we need a way of checking whether different visual outputs are consistent with one another.

Here we consider motion graphics animations as a motivating visual domain. In such animations, it is common to describe the trajectories of SVG elements as forming different shapes such as parabolas, figure 8's, or spirals (Figure 1). Given a prompt that describes a desired motion trajectory (e.g., "Move the circle in a logarithmic spiral path"), we can ask an LLM to produce an SVG motion program that renders to a corresponding animation. By incorporating prompting strategies that encourage diversity [30], we can use an LLM to produce representative samples of motion trajectories corresponding to the prompt.

To apply self-consistency to the resulting motion trajectories, we need to know which of the trajectories are consistent with one another – this is a clustering problem. For language-based reasoning tasks, self-consistency approaches often assume that this clustering step is trivially done through direct comparisons (e.g., identity matching of numerical values in math domains). For visual domains like motion trajectories, it is unlikely that any two trajectories will identically match (e.g., at a pixel-level). This is partly due to underspecification in the natural language prompt. A prompt like "move the circle in a logarithmic spiral path" does not describe a single motion trajectory, but rather a family of shapes. One way to model the family of trajectories induced by a natural language prompt is as a prototype trajectory together with a group of geometric transformations (e.g., rigid, similarity, affine, etc.): warps produced from this group specify the allowable transformations that preserve the semantic identity of the desired shape.

These transformation groups provide us with the machinery to perform the clustering required by self-consistency. Specifically, two trajectories are consistent if one can be transformed into the other under the warps allowable by the transformation group. This consistency property can be checked with a distance-based clustering algorithm, assuming we have access to a distance metric that is invariant under the transformation group for the shape family. Thus, to apply self-consistency over a set of LLM generated motion trajectory samples, if we know the correct transformation group, we can cluster the sampled trajectories under the corresponding distance metric, and choose any response from the largest cluster.

In our case, however, the *correct* family and transformation group associated with the shape described in the input prompt are not known a priori. Our approach is to first consider a hierarchy of geometric shape families based on Lie transformation groups that expose different degrees of freedom in how the prototype is allowed to vary. Our hierarchy is based on shape families commonly used to describe trajectories [21]. We cluster the LLM-produced motion trajectory samples using each transformation group in the hierarchy and we propose two decision criteria that analyze properties of these clusterings to identify the geometric shape family (with an associated prototype and transformation group) that is most consistent with the LLM samples.

Once we know the shape family, we can choose any LLM-generated sample from the largest cluster as a self-consistent generation. We also show how to extend this self-consistency approach to verify that any new query trajectory matches the prompt. More specifically, we check whether the query trajectory is a member of the shape family identified using the self-consistency approach, by testing whether it can be added to the largest cluster for the family (e.g., its distance to the prototype is below a threshold).

To evaluate our approach, we design a benchmark set of over 200 prompts describing shapes common to motion graphics animations. For generation, we find that our proposed extension of self-consistency over motion trajectories is able to improve the accuracy of LLM generations on this benchmark by 4-6% for the unsupervised setting when the transformation group needs to be heuristically estimated. Additionally, we adapt this benchmark to evaluate verification abilities by collecting over 2000 pairs of (trajectory, prompt) with labels indicating whether these match or mismatch with one another. We find that our proposed unsupervised verification strategy, which uses self-consistency to automatically recover shape families, outperforms the alternative of performing verification with a VLM with precision increases of 11.8% and F1 increases of 5.6%.

2. Related Work

Self-consistency. Self-consistency [28] 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 [13]. 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 [23] 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 com-

monly 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, 11, 12, 24, 26]. 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 [11]. 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 [26]. Verification strategies have also been proposed for other visual modalities. Design2Code proposes automatic metrics for evaluating LLM-generated webpages against design specifications [24]. SceneCraft uses LLM-based question answering to verify 3D Blender scene generation [12], 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 [19], 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.

3. Preliminaries

Motion trajectory. 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,

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

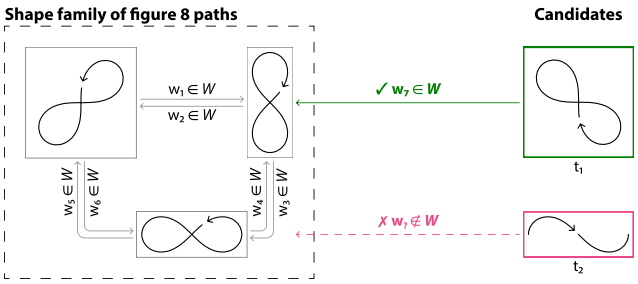

Figure 2. We define a shape family $\mathcal{F}(o, W)$ with a prototype trajectory o and a Lie transformation group W. The shape family is then the set of all trajectories that can be warped into one another $w(o)$ by all warps $w \in W$ (left). In this example, any of the figure 8's in the family can 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).

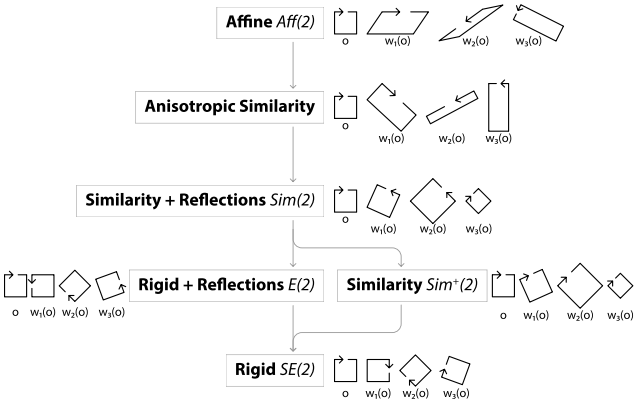

Figure 3. Hierarchy of Lie transformation groups and shape families they induce. Each node represents a transformation group and depicts a prototype square-shaped trajectory o as well as other trajectories $w(o)$ within the corresponding shape family.

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 (Figure 2).

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].

Rigid $[W_{\text{rgd}} := \mathrm{SE}(2)]$: 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 $\mathrm{SE}(2)$ it is sometimes also denoted $\mathrm{E}^+(2)$.

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

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

Affine $[W_{\text{aff}} \coloneqq \mathrm{Aff}(2)]$: 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 [25]. 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 S3).

Hierarchy of geometric warps. The Lie transformation groups form a partially ordered hierarchy (Figure 3). 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.

Warp-invariant distance metrics. 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:

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

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 $t_2 \in \mathcal{F}(t_1, W)$. 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 Supplemental Section E for details.

4. Method

With the definitions from the previous section, we have the machinery to apply self-consistency to the domain of motion trajectories. Our procedure takes an input prompt that specifies a desired motion trajectory described as shapes. In this work we assume that we can model the desired trajectory as a shape family (Equation 1) using a transformation group from our hierarchy (Figure 3). We extend ideas from self-consistency to automatically recover this shape family in an unsupervised fashion. Access to this shape family allows us to produce better LLM responses (by selecting a member from the most consistently generated shape family) and verify whether other trajectories match the specification of the prompt (by checking membership in the family).

In the rest of this section, we present the components of our approach. We first describe how we use an LLM to sample a collection of motion trajectories (Section 4.1). Next, we show that if we are given an oracle that can provide the *correct* transformation group for the desired shape family, we can use self-consistency to identify a prototype trajectory from the collection of LLM-generated samples (Section 4.2). Then we consider the case when the *correct* transformation group for the desired shape family is unknown, and show how we can estimate the transformation group using two different decision criteria (Section 4.3). Finally, in Section 4.4, we describe how to verify whether a query trajectory is a member of a given shape family.

4.1. Sampling Motion Trajectories with an LLM

Our first step uses an LLM to generate N samples of motion trajectories conditioned on the input prompt (Figure 1 right). We frame this step as a program synthesis problem: requesting the LLM to generate a motion graphics animation program, which uses functions from an API of high-level animation operations [6]. Given a static SVG scene (e.g. a single element, centered on the canvas), we can then execute this program to produce a motion trajectory.

To encourage this collection of trajectory samples to exhibit representative modes of variations for the specified tra-

jectory, we employ a prompting strategy that encourages diversity [30]. Instead of repeatedly asking the LLM to produce *independent* animation programs, we ask the LLM to produce $k < N$ programs in each response, and additionally specify that these responses should well-cover the "tails" of the correct distribution. We repeat this sampling process in batches of size k until the LLM has produced N trajectories. See Supplemental Section E.3 for the full wording of the diversity sampling prompt.

As discussed in Section 3, 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.

4.2. Self-Consistent Trajectory Clustering

If we have access to the *correct* geometric transformation group for the desired shape family (i.e. W is provided by some oracle), we describe how to use the collection of LLM sampled motion trajectories to choose a prototype. Using the distance metric d_W (associated with the transformation group W), we form a distance matrix D by computing pairwise distances under d_W between each pair of trajectories. We send this distance matrix to the DBSCAN algorithm [22] to produce clusters of consistent trajectories under W, setting the distance threshold as τ. The cluster that has the most members corresponds to the set of trajectories that are most self-consistent with one another under W; producing a self-consistent generation is then equivalent to returning any member from this cluster. To create a shape family that best aligns with the trajectories in this cluster under W, we choose the centroid sample trajectory (minimum average distance to all other cluster members under d_W) as the prototype trajectory o.

4.3. Decision Criteria for Selecting W

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.

4.3.1. 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.

4.3.2. 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.

4.4. Shape Family Membership Verification

We can use the machinery described in the previous sections to solve verification problems. For verification, we are given an input prompt that specifies a desired trajectory and shape family and a query motion trajectory t. The goal of verification is to determine if t matches the prompt. We start by applying our self-consistency approach as described above, to recover a shape family $\mathcal{F}$ from the input prompt, identifying both a transformation group W and a prototype trajectory o from a collection of LLM-generated samples. Then, given a query trajectory t, we solve the verification task by checking whether t is a member of the shape family. Formally, we say that t is a valid response with respect to the input prompt if and only if $t \in \mathcal{F}(o, W)$. Checking membership is done using the associated distance metric d_W: we compare the distance between o and t, and say that t is consistent with $\mathcal{F}$ if this distance is less than τ.

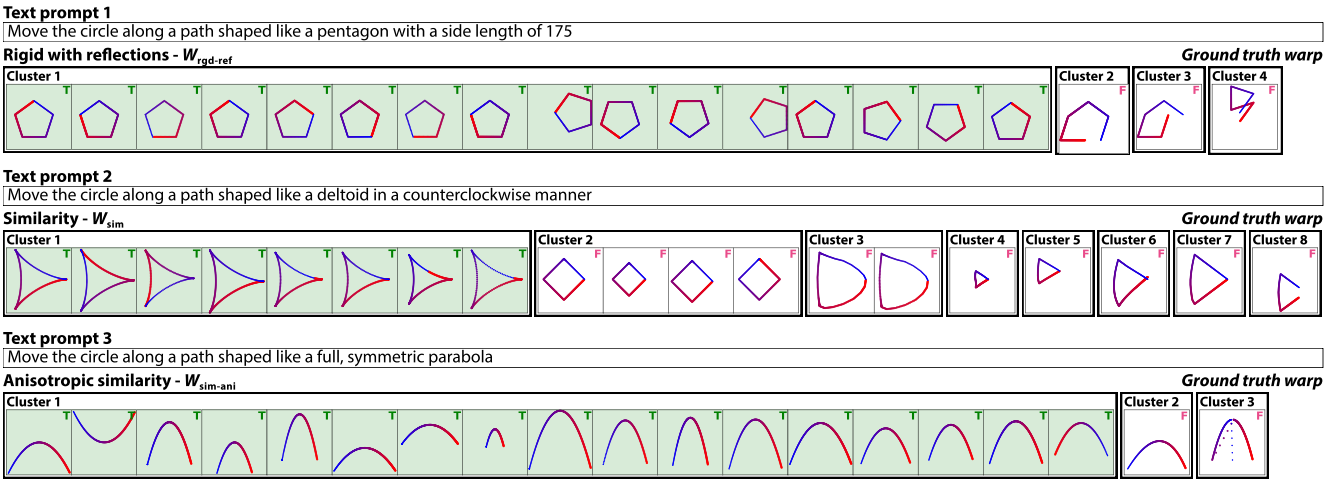

Figure 4. We present clustering results on LLM-generated samples from three example prompts in our dataset. Each trajectory has a ground truth label on the upper right. Clusters colored green are the chosen largest clusters, and we note that their sizes vary, sometimes less than half of the total number 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.

5. 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 5.1). We use this benchmark to validate that our approach provides performance boosts over baseline alternatives for both generation (Section 5.2) and verification (Section 5.3) settings.

5.1. 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 [19] and CLEVR [14]. We provide the full details of our template-based construction process and visualize all 224 pairings in the supplemental material (Section A).

5.1.1. 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).

5.1.2. 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 (Section 4.4). 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 (Section 4.1). We employed a rejection sampling-based approach until we found five true trajectories and five false trajectories for each prompt.

5.2. 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 com-

Table 1. Motion trajectory generation experiment (see Sec. 5.2). We evaluate the accuracy of LLM-generated motion trajectories (GPT-4.1 and GPT-5) under different decoding strategies. Under different decision criteria for selecting a transformation group W, our self-consistency approach improves upon the baseline alternative of directly 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

pare this baseline against our approach that samples an LLM multiple times (Section 4.1) to identify the most self-consistent trajectories (Section 4.2), 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, 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 (Section 4.3) 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.

Figure 4 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.

Table 2. Motion trajectory verification experiment (see Sec. 5.3). 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

5.3. 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 (see Supplemental Section E.4 for details). We compare these baselines against our self-consistency-based method for verification, as described in Section 4.4. 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).

5.4. Discussion

We discuss how to choose a decision criterion and set the hyperparameters for our approach. See the supplemental material (Sections D.1 and E.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 priori. We

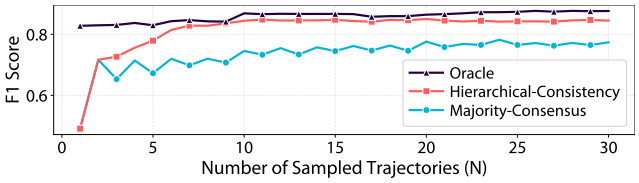

Figure 5. F1 scores across different numbers of sampled trajectories N. Decision criteria performances stabilize after $N = 10$.

allow users to flexibly choose between the two decision criteria for W estimation introduced in Section 4.3 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 (Table 2), 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 of Section 5.3 while varying N from 1 to 30, using GPT-5 as the sampling LLM. As shown in Figure 5, 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$\times$ range. We observe that F1 drops by only 7.2 percentage points across the full sweep.

6. Conclusion

We introduced an approach that extends self-consistency to the visual domain of motion graphics animations. The core challenge is determining when motion trajectories should be considered consistent with one another. We propose a shape-specific notion of consistency by identifying a hierarchy of Lie transformation groups that can be used to define common geometric shape families. Each group is paired with a distance metric that is invariant under its warps, allowing us to cluster LLM-produced motion trajectories in a self-consistent fashion. We further proposed multiple decision criteria that automatically select an appropriate transformation group for a given prompt by analyzing the behavior of these clusters. We empirically observed that our training-free, unsupervised extension of self-consistency improves performance on both motion trajectory generation and verification tasks over LLM and VLM baselines.

Limitations. Like prior work on self-consistency [28], our approach cannot recover from all mistakes made by the sampling LLM. As discussed in Section 4.3, our procedure may fail when certain assumptions on the distribution of LLM-generated trajectories are not met. We analyze these failure modes in the supplemental material (Section D).

Our approach handles prompts describing shape families characterized by a single prototype and warp, such as "move in a circular orbit" (circle under similarity warp) and "trace an Archimedean spiral" (Archimedean spiral under similarity warp with reflections). Ambiguous descriptions (e.g., "move in a curved path") or prompts with multiple disjoint families are outside our scope. For example, the prompt "move in a heptagram," "heptagram" maps to both heptagram {7/2} and heptagram {7/3} and no geometric warps can bring one to the other. When multiple prototypes exist, each prototype's cluster is smaller. This makes it harder for our decision criteria to distinguish within-family clusters from outlier ones. We experimented with a simple modification to Hierarchical-Consistency that returns all largest clusters whose sizes are within 20% of one another, improving F1 from 71.0 to 88.9 on multi-prototype prompts. See the full analysis in Supplemental Section C.1.1.

Future work. Looking ahead, we view this work as a template for extending self-consistency beyond natural–language reasoning tasks, where consistency is typically defined narrowly through identity matching. Generalizing this paradigm requires identifying a domain-relevant notion of consistency, which likely needs to be flexible with respect to prompt-level semantics. Our proposed solution for the domain of motion graphics leverages a hierarchy of common geometric transformation groups, which may be directly extendable for related spatial and shape-based domains. Exploring how this general framework might adapt over a broader set of application areas is a promising direction. Such efforts may prove especially impactful, as beyond better generation, the ability to extend consistency over distributions of related samples opens a path toward automatic, training-free verification.

Acknowledgments

Jiaju Ma was supported by the Stanford Graduate Fellowship. This work was partially supported by the Brown Institute for Media Innovation at Stanford University, NSF Awards #2219864 and #2211258, ONR YIP N00014-24-1-2117, AFOSR YIP FA9550-23-1-0127, the Stanford HAI Hoffman-Yee Research Grant (Integrating Intelligence), and the HAI-HPI Program on Artificial Intelligence (AI) and Human-Computer Interaction (HCI).

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. 4

[2] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992. 4

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

[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. 3

[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. 3

[6] Jack Doyle. Greensock animation platform v3, 2025. 4

[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. 4

[8] Ethan Eade. Lie groups for computer vision. *Cambridge Univ., Cam-bridge, UK, Tech. Rep*, 2:1, 2014. 3

[9] Rinon Gal, Yael Vinker, Yuval Alaluf, Amit Bermano, Daniel Cohen-Or, Ariel Shamir, and Gal Chechik. Breathing life into sketches using text-to-video priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4325–4336, 2024. 1

[10] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14953–14962, 2023. 1

[11] 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. 3

[12] 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. 1, 3

[13] 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. 2

[14] 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. 6

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

[16] Vladimir V Kisil. Erlangen program at large: an overview. *Advances in applied analysis*, pages 1–94, 2012. 3

[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. 2

[18] Vivian Liu, Rubaiat Habib Kazi, Li-Yi Wei, Matthew Fisher, Timothy Langlois, Seth Walker, and Lydia Chilton. Logomotion: Visually grounded code generation for content-aware animation, 2024. 1

[19] Jiaju Ma and Maneesh Agrawala. Mover: Motion verification for motion graphics animations. *ACM Trans. Graph.*, 44 (4), 2025. 1, 3, 6

[20] Sagi Polaczek, Yuval Alaluf, Elad Richardson, Yael Vinker, and Daniel Cohen-Or. Neuralsvg: An implicit representation for text-to-vector generation, 2025. 1

[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, 6

[22] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017. 5

[23] Jeong seok Oh and Jay yoon Lee. Latent self-consistency for reliable majority-set selection in short- and long-answer reasoning, 2025. 2

[24] 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. 3

[25] Carsten Steger. Least-squares estimation of anisotropic similarity transformations from corresponding 2d point sets. *Pattern Recognition Letters*, 33(3):349–355, 2012. 4

[26] Jiao Sun, Yushi Hu, Deqing Fu, Royi Rassin, Sjoerd van Steenkiste, Dana Alon, Su Wang, Charles Herrmann, Ran-

jay 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. 3

[27] Tiffany Tseng, Ruijia Cheng, Andrew M McNutt, and Jeffrey Nichols. Keyframer: A design probe for exploring llm assistance in 2d animation design. In *2025 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 33–45. IEEE, 2025. 1

[28] 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. 1, 2, 8

[29] Ximing Xing, Juncheng Hu, Guotao Liang, Jing Zhang, Dong Xu, and Qian Yu. Empowering llms to understand and generate complex vector graphics. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19487–19497, 2025. 1

[30] 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. 2, 5

[31] Yunzhi Zhang, Zizhang Li, Matt Zhou, Shangzhe Wu, and Jiajun Wu. The scene language: Representing scenes with programs, words, and embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24625–24634, 2025. 1