DRAGPHYSSVG: STRUCTURED PARTICLE DYNAMICS FOR SVG ANIMATION

Anonymous authors
Paper under double-blind review

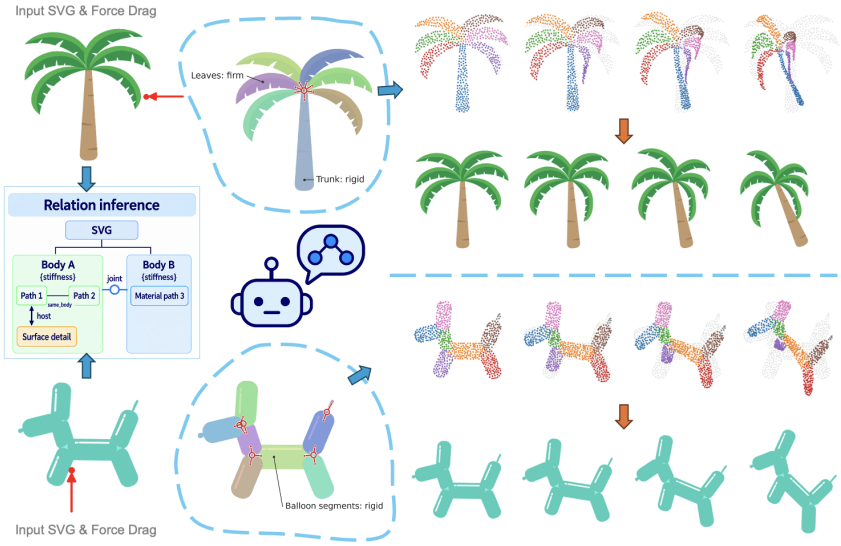


Figure 1: We propose *DragPhysSVG*, a structure-aware framework that infers physical relations from SVGs, performs drag-driven particle dynamics, and maps it back to editable vector paths.

ABSTRACT

Scalable vector graphics (SVGs) represent visual content as editable vector paths, but their path structure reflects how an image is drawn rather than how the depicted object is physically organized. This mismatch makes physically grounded animation of unrigged SVGs challenging: parts belonging to the same body should move coherently, articulated parts should remain connected, and embedded details should follow their supporting bodies. Existing SVG animation methods often rely on predefined structure or direct path deformation, while particle-based physical models do not preserve the path organization required for editable vector output. We introduce *DragPhysSVG*, a structured particle representation that bridges vector graphics and particle-based dynamics. Specifically, SVG paths are sampled into particles with particle-to-path correspondence, while a vision-language model infers their latent physical organization, including body groupings, embedded details, and articulated connections. These relations form a structured graph augmenting the particle representation with the original SVG organization. Given a user drag, a pretrained physical diffusion model predicts particle motion, and our articulated-flow guidance preserves attachments and constrains relative motion across connected parts. Resulting trajectories are transferred back to the original paths through path-local deformation, producing editable animated SVGs. Experiments on articulated SVGs demonstrate improved structural coherence, joint stability, and drag controllability while preserving vector organization. Video animations are available on the project website.

1 INTRODUCTION

Scalable vector graphics (SVGs) are widely used for digital illustration, interface design, and visual communication because they provide compact, editable, and resolution-independent representations. Extending these advantages to animation would enable dynamic assets that remain easy to inspect, modify, and reuse. Yet physically grounded animation requires more than deforming vector geometry: an object should respond coherently to interaction, connected parts should remain attached

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

while articulating, and visual details should follow the physical components to which they belong. For a static, unrigged SVG, none of this physical organization is explicitly available.

The fundamental challenge is that SVG paths encode how an image is drawn, rather than how the depicted object is physically organized. A single physical component may consist of several paths describing its silhouette, shading, and decorative details. Conversely, nearby or overlapping paths may represent either connected components, such as an arm attached to a torso, or independent objects that should move separately. Even when connectivity can be inferred geometrically, path layout alone does not specify whether two components should move rigidly together, articulate around a joint, or move independently. As a result, neither extreme provides a satisfactory representation: treating every path independently breaks coherent parts, while deforming the entire drawing as a single object suppresses articulation and introduces unwanted distortion. Physically grounded SVG animation therefore requires recovering a latent physical organization that is distinct from the graphical decomposition of the SVG.

Existing SVG animation methods address parts of this problem through learned motion priors, geometric deformation, or semantic grouping. AniClipart (Wu et al., 2025) combines video diffusion priors with skeleton-constrained deformation, LiveSVG (Levy et al., 2026) fits vector geometry to generated videos, and Vector Prism (Yun & Choo, 2026) recovers semantic groups to improve animation coherence. These approaches provide increasingly expressive motion, but do not explicitly model how user interactions should propagate through physical connections between vector components. In parallel, physics-guided generation methods such as PhysCtrl (Wang et al., 2026) and Force Prompting (Gillman et al., 2026) generate motion conditioned on physical interactions, typically in particle or raster domains. Directly applying such models to SVGs, however, loses the path identities, attachments, and structural relations needed to recover an editable vector animation. This exposes a representation gap: vector graphics preserve graphical structure but lack physical organization, whereas particle dynamics support physical motion but do not preserve vector structure.

We address this gap with *DragPhysSVG*, a structured particle representation that connects editable vector graphics with particle-based dynamics. Our key idea is to augment particles with the latent physical structure of the SVG while retaining explicit correspondence to its original paths. As shown in Fig. 1, given an unrigged SVG, a vision-language model infers physical bodies, embedded visual details, and articulated connections, which are compiled into a structured relation graph. We then sample particles for each body while preserving particle-to-path correspondence and encode inter-body relations as constraints on their motion. The resulting representation jointly captures three complementary forms of information: particles provide a domain for physical dynamics, structural relations specify how motion can propagate across components, and vector correspondences preserve how the resulting motion maps back to the original SVG.

This representation provides a general interface between SVGs and particle-based physical models. Given a user-specified drag, we instantiate this interface with a pretrained physical diffusion model that predicts particle motion. However, a generic physical prior alone does not guarantee that articulated SVG components remain consistently attached or respect their inferred relations. We therefore introduce *articulated-flow guidance*, which uses the relation graph to propagate motion across connected bodies while maintaining attachment continuity and constraining relative motion at joints. Finally, the predicted particle trajectories are converted into path-local deformation fields and transferred back to their corresponding vector paths. Physical components can therefore articulate coherently, while embedded details follow their supporting bodies and the original vector organization remains editable. Experiments on diverse articulated SVGs demonstrate improved structural coherence, joint stability, and drag controllability while preserving vector structure.

Our contributions are threefold:

- We introduce a structured SVG-to-particle representation that encodes physical bodies, inter-body relations, and path correspondences, bridging vector graphics and particle dynamics.
- We develop a VLM-guided structure inference procedure that recovers latent physical organization from unrigged SVGs and compiles it into explicit body, attachment, and articulation relations.
- We propose articulated-flow guidance for drag-controlled physical diffusion, together with path-local motion transfer that maps particle trajectories back to coherent and editable SVG animations.

2 Related Work

SVG Representation and Structural Understanding. Recent work has explored increasingly structured representations of vector graphics. DiffVG (Li et al., 2020) introduced differentiable vector rasterization for optimizing vector primitives through raster-space objectives, while LIVE (Ma et al., 2022) incorporated layer-wise topology for editable SVG reconstruction. More recent approaches model SVGs in structured sequence or code spaces. StarVector (Rodriguez et al., 2025) generates SVG code with a multimodal language model, LLM4SVG (Xing et al., 2025) introduces semantic SVG tokens, and OmniSVG (Yang et al., 2026b) leverages pretrained vision-language models for complex SVG generation. These methods primarily focus on vector synthesis and semantic organization rather than physical structure for motion. Most closely related, Vector Prism (Yun & Choo, 2026) groups low-level SVG elements semantically for VLM-based animation. In contrast, we recover a physical decomposition that groups paths into deformation bodies, separates visual details from simulated geometry, and explicitly models articulated inter-body relations.

Vector Graphics Animation. Existing methods animate static vector graphics using learned motion priors, geometric deformation, or direct vector sequence modeling. AniClipart (Wu et al., 2025) optimizes keypoint trajectories with text-to-video priors and propagates them through ARAP deformation, while LiveSVG (Levy et al., 2026) fits SVG geometry to generated videos using group-level transformations and local Bézier deformation. Vector Prism (Yun & Choo, 2026) uses semantic SVG structure for VLM-based animation planning, and VAnim (Liang et al., 2026) models animation as sparse state updates over persistent SVG structure. LottieGPT (Chen et al., 2026) and OmniLottie (Yang et al., 2026a) instead generate vector animations as parameterized token sequences. While these methods improve motion quality, editability, or structural consistency, they do not explicitly model physical bodies and articulated relations for propagating user interactions.

Physics-Grounded and Controllable Motion Generation. Recent video generation methods incorporate physical interactions as explicit motion controls (Liu et al., 2024; Zhang et al., 2024; Wang et al., 2026; Gillman et al., 2026). PhysCtrl (Wang et al., 2026) represents dynamics through 3D point trajectories and conditions a generative physics model on physical parameters and applied forces. Force Prompting (Gillman et al., 2026) conditions video generation on local and global force signals to control physical interactions. Such methods operate mainly in particle or raster domains and do not preserve the path structure or editability of vector graphics. Conversely, vector animation methods (Gal et al., 2024; Wu et al., 2025; Levy et al., 2026) retain vector representations but lack explicit physical interactions. Our representation bridges these domains by exposing SVGs as structured particle bodies with articulated relations while preserving correspondence.

3 DragPhysSVG

Given a static SVG and a user-specified drag interaction, our goal is to construct a physically meaningful representation of the SVG that supports structure-aware motion generation while preserving the editability and organization of the original vector paths. Our method first infers the physical organization underlying the SVG and compiles it into a body-aware particle representation. This representation makes explicit which visual elements belong to the same physical body, how stiff each body is, how different bodies are connected, and how particle motion corresponds back to the source vector paths. A pretrained particle-based motion generator then produces dynamics under user control. Since the generator itself is unaware of the inferred structure, we guide its sampling toward the inferred articulation and project the sampled trajectories onto the structure, and the projected motion is transferred back to the original SVG through body-local deformation.

Let $\mathcal{S} = (P_1, \ldots, P_M)$ denote the ordered vector paths of an input SVG. We organize these visual primitives into K simulation bodies, each defining a continuous deformation domain. Our body-aware representation is defined as

$$\mathcal{R} = \left(\{\mathcal{X}_b^0\}_{b=1}^K, \{s_b\}_{b=1}^K, \mathcal{J}, \Phi\right), \tag{1}$$

where $\mathcal{X}_b^0$ is the initial particle set of body b, s_b is its stiffness level, $\mathcal{J}$ encodes joints between distinct bodies, and Φ maps each original vector path to its associated simulation body. Surface details contribute no physical particles and instead inherit the deformation field of their host body. This representation preserves the original paths as editable visual elements while organizing their motion through shared body deformation and inter-body joint constraints.

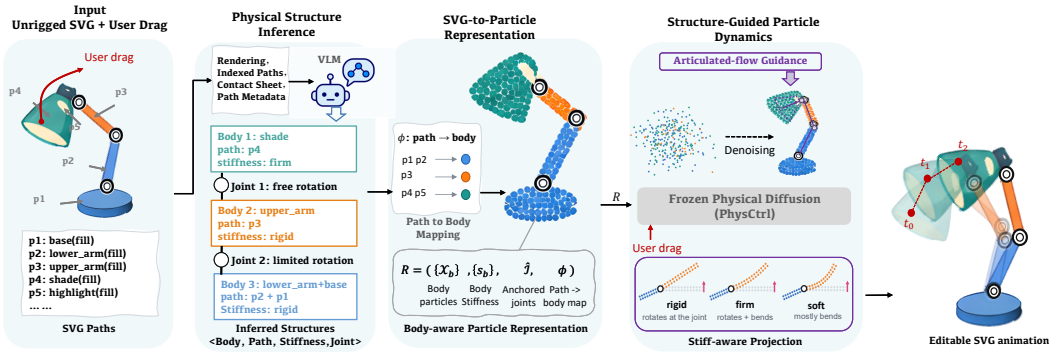


Figure 2: Overview of the proposed DragPhySVG. Given an unrigged SVG and a user drag, a VLM infers bodies, stiffness levels, and joints from a composite input, which are compiled into a body-aware particle representation $\mathcal{R}$; a frozen PhysCtrl then generates particle trajectories under articulated-flow guidance, and stiffness-aware projection transfers the motion back to the editable SVG paths.

Given a user interaction, a downstream motion model generates trajectories for the particles of $\mathcal{R}$, with its sampling guided toward the articulation encoded in $\mathcal{R}$. We decompose each body's trajectory into a rigid motion and a local deformation, constrain the relative rotations at joints by the joint types and body stiffness, and scale each body's local deformation by its stiffness. We then use Φ to assign each vector path to the projected motion of its body. Applying this motion to the path's control points produces an editable animated SVG, with surface details following their host body and the original path organization preserved. The overview is illustrated in Figure 2.

3.1 Physical Structure Inference

Composite Inference Input. SVGs are often highly abstracted, so visual composition and geometry alone may not reveal their underlying physical organization. Multiple paths may belong to one continuous body, while overlapping paths may represent independently moving parts or decorative elements. We therefore use a vision-language model (VLM) to infer physical organization from both visual and path-level information. The input comprises an unannotated SVG rendering, an indexed-path rendering, a per-path contact sheet, and structured path metadata, providing global appearance, path identity, and vector information for grounding predictions to specific SVG primitives.

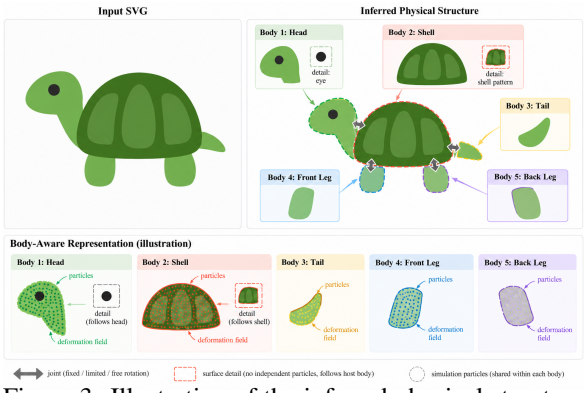


Figure 3: Illustration of the inferred physical structure and body-aware representation.

Body and Relation Representation. The VLM infers how vector paths form physical bodies and how these bodies are connected, as illustrated in Figure 3. Paths linked by `same_body` relations form a shared deformation domain. Surface details, such as highlights or decorative marks, contribute no physical particles and instead inherit the deformation field of their host body. Between distinct bodies, `joint` relations specify attachments with fixed, limited, or free relative rotation. Each physical path is assigned a stiffness level in $\{rigid, firm, soft\}$, and the body stiffness s_b is determined by majority vote, with ties resolved toward the stiffer level. These associations, stiffness levels, and inter-body connections define the physical structure used below.

We represent the inferred structure as

$$\mathcal{H} = (\mathcal{B}, \mathcal{J}), \qquad \mathcal{B} = \{B_b\}_{b=1}^{K}, \tag{2}$$

where each body

$$B_b = (\mathcal{P}_b, \mathcal{Q}_b, s_b) \tag{3}$$

contains physical paths $\mathcal{P}_b$ forming a shared deformation domain, surface-detail paths $\mathcal{Q}_b$ that follow the body without contributing particles, and body stiffness s_b. The set $\mathcal{J}$ contains inter-body joints and their relative-motion types.

3.2 Body-Aware SVG-to-Particles Representation

We convert $\mathcal{H}$ into a particle representation by discretizing each body's deformation domain, preserving path associations, and grounding inter-body joints in SVG geometry.

Body-Level Sampling Domains. For each body B_b, we construct its deformation domain by taking the union of the spatial regions occupied by its physical paths:

$$\Omega_b = \bigcup_{P_i \in \mathcal{P}_b} \Omega(P_i), \tag{4}$$

where $\Omega(P_i)$ denotes the region occupied by path P_i. We rasterize these regions on a shared canvas and merge their masks before sampling, removing internal boundaries introduced by the visual decomposition and avoiding duplicate samples in overlapping regions.

Particle Discretization. We distribute a fixed budget of N particles across bodies according to their union areas, subject to a minimum allocation per body. Within body b, candidate points are sampled uniformly from Ω_b and N_b points are selected by farthest point sampling:

$$\mathcal{X}_b^0 = \left\{\mathbf{x}_{b,n}^0\right\}_{n=1}^{N_b}, \qquad \sum_{b=1}^{K} N_b = N. \tag{5}$$

Farthest point sampling provides even coverage of the material region. All particles are normalized using a shared coordinate transform while retaining their body identities.

Path-to-Body Associations. To preserve SVG editability, we retain a mapping Φ from each original path to its associated body:

$$\Phi(P_i) = b, \qquad P_i \in \mathcal{P}_b \cup \mathcal{Q}_b. \tag{6}$$

Physical paths follow their `same_body` component, while surface details inherit the body of their host path. Thus, multiple paths may share one particle-driven deformation field while their original vector geometry is retained for direct deformation.

Geometric Joint Grounding. Inferred joints specify which bodies are connected and the allowed relative motion, but not the attachment location. For each joint, we locate the closest contact region between the corresponding source-path boundaries and estimate an attachment anchor for each body. We then organize the joints into a spanning forest, where each connected component is rooted at the body with the largest area and highest joint degree, and every other body has one parent joint.

After this step, the body-aware SVG-to-particle representation is

$$\mathcal{R} = \left(\left\{\mathcal{X}_b^0\right\}_{b=1}^{K}, \{s_b\}_{b=1}^{K}, \widehat{\mathcal{J}}, \Phi\right), \tag{7}$$

where $\widehat{\mathcal{J}}$ denotes the joint forest augmented with geometric attachment anchors. This representation provides body-level particles, explicit inter-body constraints, and the correspondence required to transfer motion back to the original SVG paths.

3.3 Structure-Guided Motion Generation and Transfer

Given a drag, a frozen particle-based generator predicts trajectories $\mathbf{p}_{b,n}^t$ for the particles of $\mathcal{R}$, with $\mathbf{p}_{b,n}^0 = \mathbf{x}_{b,n}^0$. Since the generator is unaware of the inferred structure, we impose it in three stages: guidance steers sampling toward articulated motion, a stiffness-aware projection then enforces $\mathcal{R}$ exactly, and the result is transferred to the SVG.

Stage 1: Articulated-Flow Guidance. At each denoising step, the clean estimate $\hat{\mathcal{X}} = \{\hat{\mathbf{p}}_{b,n}^t\}$ yields per-frame velocities $\hat{\mathbf{v}}_{b,n}^t = \hat{\mathbf{p}}_{b,n}^t - \hat{\mathbf{p}}_{b,n}^{t-1}$, which we explain by the generalized velocities $\boldsymbol{\xi}^t$ of the joint forest (root twists and joint angular velocities):

$$\boldsymbol{\xi}^t = \arg\min_{\boldsymbol{\xi}} \sum_{b,n} \omega_{b,n} \left\| \mathbf{J}_{b,n}\boldsymbol{\xi} - \hat{\mathbf{v}}_{b,n}^t \right\|^2, \tag{8}$$

where $\mathbf{J}_{b,n}$ is the kinematic Jacobian at the previous articulated pose and $\omega_{b,n}$ emphasizes the dragged path and decays with structural distance from it. Integrating $\boldsymbol{\xi}^t$ within the joint ranges gives an articulated rollout $\mathcal{A}(\hat{\mathcal{X}})$; temporal integration allows weak per-frame rotational cues to accumulate into finite articulation. We move the estimate toward this rollout, $\hat{\mathcal{X}} \leftarrow \hat{\mathcal{X}} + \lambda\big(\mathcal{A}(\hat{\mathcal{X}}) - \hat{\mathcal{X}}\big)$. The guidance strength $\lambda \in (0, 1]$ is adjustable: $\lambda = 1$ replaces the estimate by the rollout, while smaller values retain more of the generator's own dynamics. Guidance uses only the joint structure, so the sample is close to, but not exactly on, the permitted articulation.

Stage 2: Stiffness-Aware Projection. We fit a rigid transform with angle θ_b^t to each body of the sample and take its rotation $\Delta\theta_b^t$ relative to its parent. A stiff body cannot bend, so its motion must be carried by its joint, whereas a soft body can absorb it by deforming. With a deformation penalty k_b increasing with s_b and $\kappa_b = k_b/(1 + k_b)$, the applied joint rotation is

$$\psi_b^t = \underset{\psi \in \mathcal{C}_b}{\arg\min}\ \kappa_b \left(\psi - \Delta\theta_b^t\right)^2 + \beta\, \psi^2, \tag{9}$$

where $\mathcal{C}_b$ is $\{0\}$, $[-\alpha, \alpha]$, or $\mathbb{R}$ for fixed, limited, and free joints; the minimizer is $\frac{\kappa_b}{\kappa_b + \beta}\Delta\theta_b^t$ projected onto $\mathcal{C}_b$. Roots keep their fitted motion, and forward kinematics places every other body so that its anchor coincides with its parent's. The motion left unexplained by these transforms $(\mathbf{R}_b^t, \boldsymbol{\tau}_b^t)$ forms a body-frame residual $\mathbf{r}_{b,n}^t$, of which each body keeps $\mathbf{d}_{b,n}^t = \arg\min_{\mathbf{d}} \|\mathbf{d} - \mathbf{r}_{b,n}^t\|^2 + k_b\|\mathbf{d}\|^2 = \mathbf{r}_{b,n}^t/(1 + k_b)$, constrained to vanish at every joint. Rigid bodies thus move as rigid pieces, soft bodies reproduce the sampled deformation, and joints stay attached by construction.

Stage 3: Vector Path Deformation. Each control point $\mathbf{c}_{i,\ell}^0$ of a path P_i with $\Phi(P_i) = b$, including surface details, follows its body:

$$\mathbf{c}_{i,\ell}^t = \mathbf{R}_b^t\Big(\mathbf{c}_{i,\ell}^0 + \sum_{n \in \mathcal{N}_b(\mathbf{c}_{i,\ell}^0)} w_{b,n}\, \mathbf{d}_{b,n}^t\Big) + \boldsymbol{\tau}_b^t, \tag{10}$$

where $\mathcal{N}_b(\cdot)$ contains the nearest reference particles of the same body and $w_{b,n}$ are normalized inverse-distance weights, so motion never blends across bodies. The deformed points are written as animation keyframes that preserve the original path structure and appearance.

4 Evaluation

Settings. We evaluate our method on single-object SVGs collected from two sources. The first set contains 42 SVGs from AniClipart (Wu et al., 2025). We additionally select 100 SVGs from the SVG-Emoji subset of SVG-Bench (Rodriguez et al., 2025). We retain examples containing at least three vector paths ($n_{\text{paths}} \geq 3$), while excluding text-only graphics, logos, repetitive geometric patterns, and overly complex SVGs with more than 100 paths. We further restrict the evaluation to single-object illustrations whose paths jointly depict a coherent object with body-level structure. Detailed implementation settings, including structure inference, particle sampling, motion generation, and projection parameters, are provided in Appendix A.1.

4.1 Comparison with Existing Representations and Animation Methods

We compare against both physical-video generation methods and SVG animation baselines. For the physical-video baselines, we include PhysCtrl (Wang et al., 2026) and Force Prompting Gillman et al. (2026) using their official implementations and released checkpoints. We follow a controlled setup with the same rasterized SVG inputs, four inward force directions per object, and matched text prompts; for PhysCtrl, we use the released elastic model with gravity disabled, while Force Prompting uses its point-force model with the official sampling settings. We additionally compare with AniClipart (Wu et al., 2025) as an SVG animation baseline, using the same input prompt as previous methods and its default inference settings.

Fig. 4 shows two representative cases. Force Prompting's motion is inaccurate and weakly conditioned on the applied force. The kite barely responds to the downward push, and its colors drift from frame to frame. AniClipart is driven by the text prompt alone, so it is difficult to follow the applied force accurately. Moreover, the physical conditions cannot be applied to the animated ARAP-based deformation merely with prompts. PhysCtrl does respond along the force direction. However, it

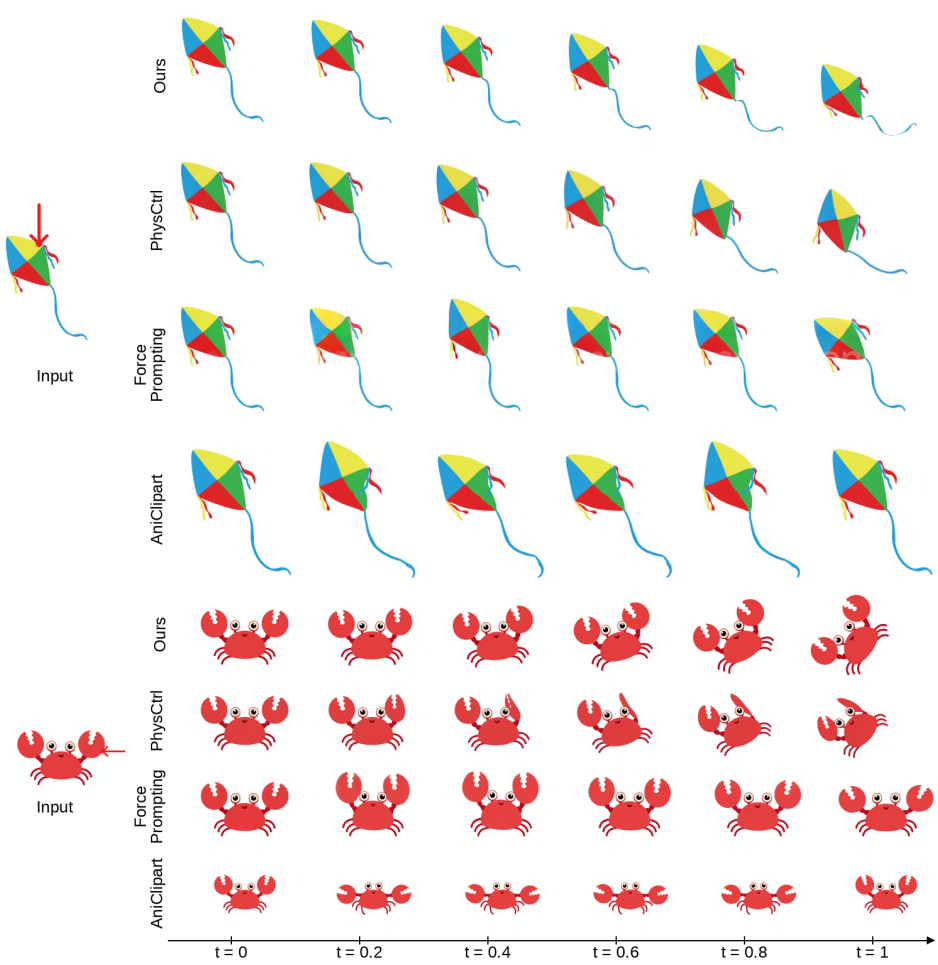


Figure 4: Qualitative comparison of our method with PhysCtrl, Force Prompting, and AniClipart on SVG animation examples. Given an input SVG and a user-specified force direction, we visualize the generated motion at normalized timestamps. More cases are available on the project website.

treats the object as an unstructured point set, so the response spreads through the shape as a continuum deformation, making it difficult to preserve the structure of the SVG input. The crab's right claw is sheared into a thin sliver, and the kite's outline distorts as it rotates. In contrast, our method keeps every part consistent with its assigned stiffness. The kite frame and the crab carapace stay rigid, the tail and streamers bend, and the claws rotate about their joints, while the overall motion follows the applied force.

4.2 Qualitative Results

Metrics. Because SVG inputs lack ground-truth motion, we evaluate rendered results using CoTracker3 (Karaev et al., 2025) to track densely sampled points from the first frame. We group metrics into four categories. *Controllability* is measured by force alignment A_{force}, the mean cosine similarity between the applied force $\mathbf{f}$ and the mean displacement $\bar{\mathbf{u}}_h^t$ of points near the interaction location. We compute it over the first third of the clip, excluding frames with displacement below $0.01L$, where L is the object bounding-box diagonal. *Structure preservation* uses three metrics. Body distortion E_{body} measures the normalized change in pairwise distances between points on the same body, reported for rigid bodies and all bodies. Joint drift $E_{\text{joint}} = \|T_a^t(\mathbf{p}) - T_b^t(\mathbf{p})\|/L$ measures disagreement between the two local body motions at a shared joint anchor $\mathbf{p}$, remaining zero for an attached joint even under rotation. BNE_0 (Stucki et al., 2023) measures changes in the number of connected foreground components, providing a structure-independent topology metric. *Video quality* is evaluated using VBench Subject Consistency (Huang et al., 2024). Finally, *perceived physical plausibility* is evaluated with Qwen3-VL-32B (Bai et al., 2025) as a pairwise judge.

Table 1: Quantitative comparison. (a) The AniClipart dataset. AniClipart is text-driven, so A_{force} does not apply. (b) The SVG-Bench-100 dataset. Best results per block are in bold. SC: subject consistency.

	Control	Structure				Video quality
Method	$A_{\text{force}} \uparrow$	$E_{\text{body}}^{\text{rigid}} \downarrow$	$E_{\text{body}} \downarrow$	$E_{\text{joint}} \downarrow$	$\text{BNE}_0 \downarrow$	SC↑
(a) 42 inputs covered by AniClipart (168 push configurations)						
PhysCtrl	0.925	0.0248	0.0247	0.0096	0.356	0.973
Force Prompting	0.260	0.0207	0.0244	0.0129	0.419	0.922
AniClipart	–	0.0287	0.0232	0.0089	**0.024**	0.975
Ours	**0.959**	**0.0102**	**0.0194**	**0.0064**	0.362	**0.978**
(b) Remaining 100 inputs (400 push configurations)						
PhysCtrl	0.859	0.0238	0.0233	0.0111	0.258	0.978
Force Prompting	0.222	0.0151	0.0162	0.0079	0.255	0.935
Ours	**0.975**	**0.0057**	**0.0100**	**0.0050**	**0.220**	**0.987**

Table 2: Pairwise VLM judgment with Qwen3-VL-32B: win rate (%) of our method against each baseline. Each pair is judged in both presentation orders; ties count as half.

Ours vs.	Force response	Material behavior	Structural–semantic integrity	Plausibility
PhysCtrl	54.1	51.2	54.0	58.6
Force Prompting	63.9	63.1	58.2	63.5
AniClipart	60.7	57.3	47.6	58.9

Given uniformly sampled frames, the object name, and push direction, it compares two results on force response, material behavior, structural–semantic integrity, and overall plausibility. Each pair is evaluated in both presentation orders, with ties counted as half. The judge receives only rendered frames and no inferred structural annotations.

Table 1 summarizes the results. Our method achieves the highest force alignment in both blocks (0.959 and 0.975), so the structural guidance does not come at the cost of controllability: the motion still follows the applied force, whereas Force Prompting responds only weakly to the push direction ($A_{\text{force}} \leq 0.26$). The main gains lie in structure preservation. Rigid bodies deform 2–4$\times$ less than with any baseline ($E_{\text{body}}^{\text{rigid}} = 0.0102$ vs. 0.0207–0.0287 in (a), and 0.0057 vs. 0.0151–0.0238 in (b)), and joint drift is the lowest in both blocks, indicating that connected parts stay attached while still moving relative to each other. Our method also attains the highest subject consistency. For BNE_0, our method performs best in (b) and is on par with PhysCtrl in (a), while AniClipart obtains a much lower error: it deforms the input through a fixed triangulated mesh regularized by ARAP, which keeps the connected components nearly unchanged by construction.

As shown in Table 2, our method achieves higher overall plausibility win rates than all compared baselines, with particularly clear gains over Force Prompting and AniClipart. The improvements are also reflected in force response and material behavior. For structural-semantic integrity, our method outperforms both physical video baselines and is only slightly below AniClipart, indicating that the improved physical plausibility is achieved while largely preserving the structural and semantic consistency of the original SVG.

4.3 Ablation Studies

The ablations highlight the complementary roles of the three components. Removing articulated-flow guidance leads to excessive local rotations and drift, showing that guidance is important for steering the generator toward structurally coherent motion. Without stiffness awareness, nominally rigid parts inherit unconstrained deformation and may collapse or tear, while removing the relation graph causes motion to blend across semantic parts, producing severe cross-body stretching and thin-strip artifacts. Together, these results show that guidance, stiffness modeling, and explicit inter-part relations are all necessary for physically plausible and structure-preserving SVG animation.

Articulated-flow guidance. One could instead sample the generator freely and fit the inferred structure to the resulting point cloud. However, the generator has no notion of joints, so its trajectory often contains motion that no articulation can explain. A post-hoc fit can only clip this motion, not

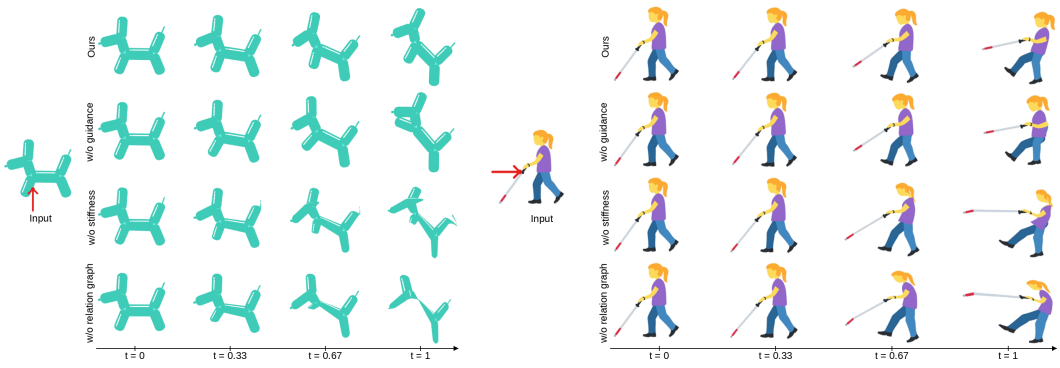


Figure 5: Ablation study. Removing articulated-flow guidance causes excessive rotation and drift, removing stiffness awareness lets rigid parts collapse or tear, and removing the relation graph blends motion across parts, causing cross-body stretching.

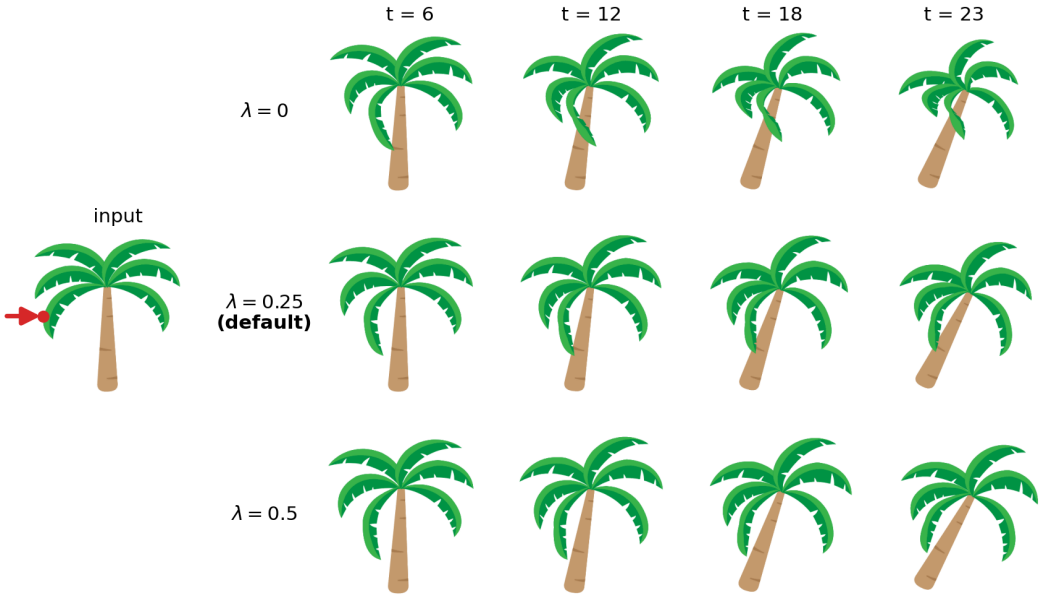


Figure 6: Effect of the guidance strength λ. Without guidance, the branch becomes overly twisted. Guidance keeps the motion within the permitted articulation. Demos can be found here.

redistribute it. As shown in Figure 6, without guidance ($\lambda = 0$) the branch becomes overly twisted even after projection. Guidance instead acts during sampling, letting later denoising steps adapt the rest of the motion to the articulated estimate, and thus keeps the motion within the permitted articulation. Increasing λ further yields increasingly regular but less varied motion, and results at $\lambda = 0.5$ are visually close to full replacement ($\lambda = 1$). We therefore use $\lambda = 0.25$, which removes the folding while retaining most of the generator's own dynamics.

5 Conclusion

We presented *DragPhysSVG*, a structure-aware framework for physically grounded, drag-controlled SVG animation. By bridging vector paths and particle dynamics through explicit body relations and path correspondences, our method enables coherent motion while preserving SVG editability. Articulated-flow guidance further stabilizes connected parts during interaction, allowing particle trajectories to be transferred back to vector paths with improved structural consistency. We hope this representation provides a useful foundation for combining editable vector graphics with general physical motion models.

AI USE STATEMENT

In this work, we used generative AI tools to assist with parts of the code implementation. We have not used generative AI tools to generate or clean data sets, develop the conceptual framework, propose hypotheses, design the methodology or experiments, or interpret results, and formulating mathematical claims, writing proofs, translation, and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools to assist with code debugging and to improve the readability of the paper. We have reviewed all AI-assisted work: all AI-assisted code was reviewed and tested by the authors, all reported experiments were run and verified by the authors, and all AI-edited text was checked against our results and cited references. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work develops a method for physically grounded, drag-controlled animation of static SVG illustrations. It does not involve human-subject studies, and it does not collect or process personal data. All experiments are conducted on publicly available SVGs from AniClipart and the SVG-Emoji subset of SVG-Bench, used in accordance with their original licenses, and we do not release any new data set. Our method relies on publicly released pretrained models (Qwen3-VL and PhysCtrl), which we use without modification in accordance with their licenses. Since the physical structure is inferred by a vision-language model, it may inherit errors or biases of that model, so the generated animations should be regarded as editable drafts that users can inspect and correct. We have read and adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We have made efforts to ensure the reproducibility of our results. Section 3 describes each component of our method, including physical structure inference, the body-aware SVG-to-particle representation, and structure-guided motion generation and transfer. Section 4.1 describes the data sources and the selection criteria for the evaluation SVGs, and Section 4.2 provides the implementation details, including the pretrained models, hardware, and all hyperparameters. Section 4.4 defines the evaluation metrics, and Appendix B provides the settings used to reproduce the baseline methods.

REFERENCES

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

Junhao Chen, Kejun Gao, Yuehan Cui, Mingze Sun, Mingjin Chen, Shaohui Wang, Xiaoxiao Long, Fei Ma, Qi Tian, Hao Zhao, et al. Lottiegpt: Tokenizing vector animation for autoregressive generation. In *CVPR*, pp. 31639–31651, 2026.

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 *CVPR*, pp. 4325–4336, 2024.

Nate Gillman, Charles Herrmann, Michael Freeman, Daksh Aggarwal, Evan Luo, Deqing Sun, and Chen Sun. Force prompting: Video generation models can learn and generalize physics-based control signals. *NeurIPS*, 38:103174–103201, 2026.

Zekai Gu, Rui Yan, Jiahao Lu, Peng Li, Zhiyang Dou, Chenyang Si, Zhen Dong, Qifeng Liu, Cheng Lin, Ziwei Liu, Wenping Wang, and Yuan Liu. Diffusion as shader: 3d-aware video diffusion for versatile video generation control. *arXiv preprint arXiv:2501.03847*, 2025.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818. IEEE, 2024.

Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labeling real videos. In *ICCV*, pp. 1–10. IEEE, 2025.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *2023 IEEE/CVF international conference on computer vision (ICCV)*, pp. 3992–4003. IEEE, 2023.

Matan Levy, Ran Margolin, Bar Cavia, Dvir Samuel, Yael Pritch, Shmuel Peleg, Alex Rav Acha, Ariel Shamir, and Dani Lischinski. Livesvg: Zero-shot svg animation via video generation. In *ACM SIGGRAPH Asia Conf.*, 2026.

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM TOG*, 39(6):1–15, 2020.

Guotao Liang, Zhangcheng Wang, Chuang Wang, Juncheng Hu, Haitao Zhou, Junhua Liu, Jing Zhang, Dong Xu, and Qian Yu. VAnim: Rendering-aware sparse state modeling for structure-preserving vector animation. In *ICML*, 2026.

Shaowei Liu, Zhongzheng Ren, Saurabh Gupta, and Shenlong Wang. Physgen: Rigid-body physics-grounded image-to-video generation. In *ECCV*, 2024.

Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *CVPR*, pp. 16293–16302, 2022.

Juan A Rodriguez, Abhay Puri, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. Starvector: Generating scalable vector graphics code from images and text. In *CVPR*, pp. 16175–16186, 2025.

Nico Stucki, Johannes C Paetzold, Suprosanna Shit, Bjoern Menze, and Ulrich Bauer. Topologically faithful image segmentation via induced matching of persistence barcodes. In *ICML*, pp. 32698–32727, 2023.

Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *European Conference on Computer Vision*, pp. 1–18. Springer, 2024.

Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitry Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. Sv3d: Novel multi-view synthesis and 3d generation from a single image using latent video diffusion. In *European Conference on Computer Vision*, pp. 439–457. Springer, 2024.

Chen Wang, Chuhao Chen, Yiming Huang, Zhiyang Dou, Yuan Liu, Jiatao Gu, and Lingjie Liu. Physctrl: Generative physics for controllable and physics-grounded video generation. *NeurIPS*, 38:167907–167932, 2026.

Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. Aniclipart: Clipart animation with text-to-video priors. *IJCV*, 133(6):3149–3165, 2025.

Ximing Xing, Juncheng Hu, Guotao Liang, Jing Zhang, Dong Xu, and Qian Yu. Empowering llms to understand and generate complex vector graphics. In *CVPR*, pp. 19487–19497. IEEE, 2025.

Yiying Yang, Wei Cheng, Sijin Chen, Honghao Fu, Xianfang Zeng, Yujun Cai, Gang Yu, and Xingjun Ma. Omnilottie: Generating vector animations via parameterized lottie tokens. In *CVPR*, pp. 39293–39303, June 2026a.

Yiying Yang, Wei Cheng, Sijin Chen, Xianfang Zeng, Fukun Yin, Jiaxu Zhang, Liao Wang, Gang Yu, Xingjun Ma, and Yu-Gang Jiang. Omnisvg: A unified scalable vector graphics generation model. *NeurIPS*, 38:113670–113696, 2026b.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, Da Yin, Yuxuan Zhang, Weihan Wang, Yean Cheng, Bin Xu, Xiaotao Gu, Yuxiao Dong, and Jie Tang. Cogvideox: Text-to-video diffusion models with an expert transformer, 2025. URL https://arxiv.org/abs/2408.06072.

Jooyeol Yun and Jaegul Choo. Vector prism: Animating vector graphics by stratifying semantic structure. In *CVPR*, pp. 17316–17325, 2026.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3813–3824. IEEE, 2023.

Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y. Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T. Freeman. Physdreamer: Physics-based interaction with 3d objects via video generation. In *ECCV*, 2024.

A Appendix

A.1 Implementation Details

Structure inference. We use Qwen3-VL-32B-Instruct (Bai et al., 2025) as the VLM, served with vLLM on two NVIDIA L40S GPUs. The composite input consists of the clean rendering, the indexed rendering, and the per-path contact sheet, all rendered at 1024×1024, together with the structured path table. The answer is decoded under a JSON schema that fixes the per-path roles, stiffness levels, and relation types, with greedy decoding (temperature 0). Inference takes about 1.5–2.5 minutes per SVG.

Particle representation. Each SVG is represented by $N = 2048$ particles, matching the input size of the motion generator. Body domains are rasterized at 1024×1024. Within each body, we draw $20N_b$ candidate points uniformly from its domain and select N_b of them by farthest point sampling.

Motion generation. We use the elastic PhysCtrl model (Wang et al., 2026) without any fine-tuning as the default model. It generates 24 frames with 25 DDIM steps. Particles are scaled to the generator's training scale before inference, and the output is mapped back. For all drag directions, we use $\log E = 4$, $\nu = 0.4$, and a force magnitude of 0.04 applied at the drag point, without gravity. Articulated-flow guidance uses $\lambda = 0.25$ and is active from the first 10% of the denoising steps onward. The path-level drag weight is 12, decaying by a factor of 0.65 per joint away from the dragged body. Sampling one drag takes about 15 seconds on a single NVIDIA L40S GPU.

Projection and transfer. The stiffness levels map to $k = \infty$ (rigid), 1 (firm), and 0 (soft). We set $\beta = 0.1$ and $\alpha = 40^\circ$ for all SVGs, and no parameter is tuned per case. Because the generator is trained on free-floating objects, its output contains a whole-body translation. For better visualization, we remove it by keeping each root's rest centroid fixed while retaining its rotation. Control points interpolate the residuals of the 8 nearest particles of their body with inverse-distance weights ($p = 2$).

A.2 Baseline Reproduction Details

PhysCtrl. PhysCtrl Wang et al. (2026) generates physics-grounded 3D point trajectories with a diffusion model and renders them into video. We run the official code and released checkpoints with the pipeline itself unchanged. Each SVG is rasterized with CairoSVG, cropped to its non-transparent pixels, scaled so that the object spans at most 50% of the frame width or height, and centered on a 720×480 white background. PhysCtrl segments the object with SAM Kirillov et al. (2023), synthesizes novel views with SV3D Voleti et al. (2024), reconstructs 3D Gaussians with LGM Tang et al. (2024), and samples 2,048 points. It then predicts 24 frames of point trajectories, about 2 s of motion. We use the elastic base model, the only released model trained with externally applied drag forces. Gravity is disabled, $\log_{10} E = 4$, $\nu = 0.4$, and the force magnitude is 0.04 in PhysCtrl's

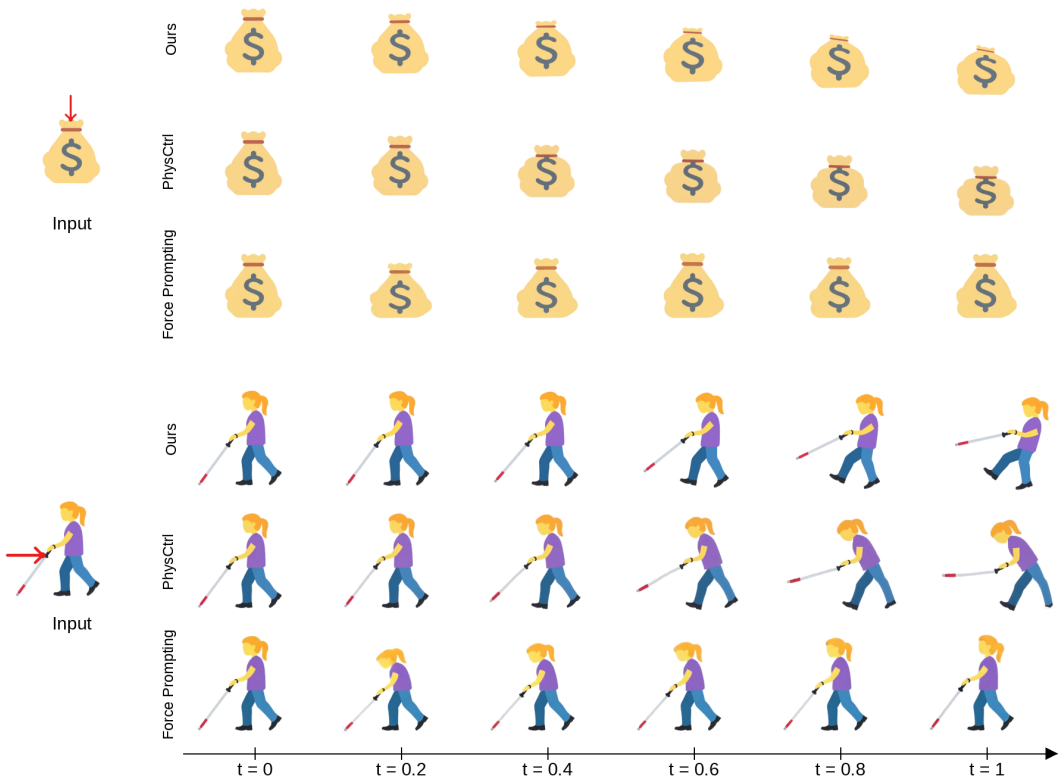


Figure 7: Qualitative comparison of our method with PhysCtrl, Force Prompting on representative SVG-Bench examples. More animation results are on our project website.

normalized units. Every SVG is animated four times, once with an inward force from each side (top, bottom, left, right), which gives $142 \times 4 = 568$ videos. The drag point is the middle of the loaded side, and the force acts on all points within a cube of half-width 0.4 around it. The trajectories are projected into the image, interpolated to 49 frames, and rendered by Diffusion as Shader Gu et al. (2025), a CogVideoX-5B Yang et al. (2025) model, with 50 denoising steps, guidance scale 6.0, and seed 42, at 24 fps. The prompt follows the template "$\langle$*subject*$\rangle$ *on a plain white background is* $\langle$*action*$\rangle$ *and deforms softly like rubber and springs back.*", where $\langle$action$\rangle$ names the loaded side, e.g., "pushed up from the bottom".

Force Prompting. Force Prompting Gillman et al. (2026) conditions CogVideoX-5B-I2V Yang et al. (2025) on a physical force through a six-block ControlNet Zhang et al. (2023). We use the official code and the released point-force checkpoint, which was trained on Blender videos of rolling balls and poked carnations. We do not use the wind-force model: it takes a spatially uniform wind field, whereas our task applies a force at one location of the object. To keep the comparison controlled, we reuse PhysCtrl's settings: the same input images, the same four inward force directions per SVG, and the same prompts. A point force is rendered as a Gaussian blob of radius 20 px that starts at the poke point and moves along the force direction over the 49 frames. The travel distance grows linearly with the normalized magnitude, from 90 to 360 px. We use the maximum training magnitude, 1.0. We place the poke point on the row (or column) through the object's mask centroid, 10% of the object's extent inside its edge on the loaded side. Force Prompting has no material input, so we append a fixed sentence to each PhysCtrl prompt to describe a soft, elastic material and its deformation: "*The object is made of a soft, squishy and highly elastic rubbery material, like a jelly toy, so it squashes, stretches and wobbles under the push before bouncing back to its original shape.*" The prompt reaches both the ControlNet and the base transformer, but not the force signal. Sampling uses the released settings: 50 DPM steps, guidance scale 6.0, and seed 42, producing 49 frames at 720×480 and 8 fps (about 6 s).