EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

Vector sketch animation generation with differentiable motion trajectories

X. Zhu[1], X. Yang[1], S. Zheng[1], Z. Zhang[2], F. Gao[1], J. Huang[3] and J. Chen[†1]

[1]Zhejiang University of Technology [2] Hangzhou Dianzi University, [3] Zhejiang Gongshang University

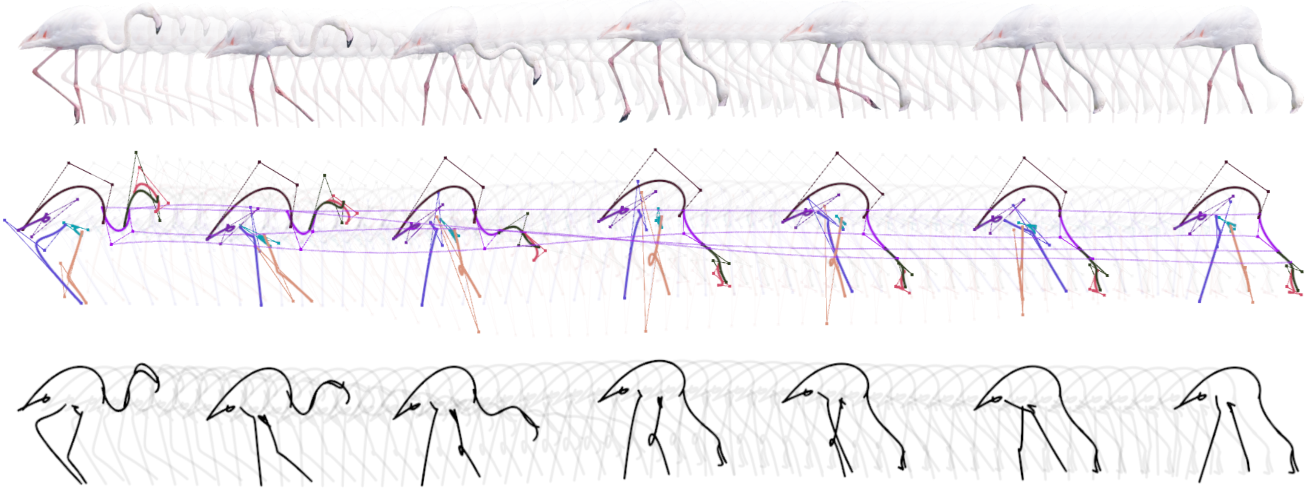


Figure 1: *Our method converts an object video (top) into a 2D vector sketch animation (bottom). We propose a differentiable motion trajectory with Bernstein basis to model cross-frame movement of stroke control points. The visualization in the middle displays their trajectories from a single stroke, and our approach boosts temporal/semantic consistency, performing robustly for sparse strokes and long videos.*

Abstract

Sketching is a direct and inexpensive means of visual expression. Though image-based sketching has been well studied, video-based sketch animation generation is still very challenging due to the temporal coherence requirement. In this paper, we propose a novel end-to-end automatic generation approach for vector sketch animation. To solve the flickering issue, we introduce a Differentiable Motion Trajectory (DMT) representation that describes the frame-wise movement of stroke control points using differentiable polynomial-based trajectories. DMT enables global semantic gradient propagation across multiple frames, significantly improving the semantic consistency and temporal coherence, and producing high-framerate output. DMT employs a Bernstein basis to balance the sensitivity of polynomial parameters, thus achieving more stable optimization. Instead of implicit fields, we introduce sparse track points for explicit spatial modeling, which improves efficiency and supports long-duration video processing. Evaluations on DAVIS and LVOS datasets demonstrate the superiority of our approach over SOTA methods. Cross-domain validation on 3D models and text-to-video data confirms the robustness and compatibility of our approach.

Keywords: Sketch Synthesis, Animation Generation, Differentiable Motion Trajectories, Temporal Coherence

1. Introduction

Vector animations, defined by geometric primitives (e.g., points, lines, curves, and polygons), offer significant advantages, including small file size, resolution-independent scalability, and ease of editing. Consequently, they are widely adopted in various appli-

[†] Corresponding author. Email: cjz@zjut.edu.cn.

cations, such as artistic design, industrial design, and data visualization [DRVDP15, BOH11, TG22, ZCZ*09]. As a special form of vector animation, vector sketch animation simulates the human hand-drawing process with stylistic stroke feel, line dynamics, and visual abstraction, thus providing enhanced artistic expression and emotional appeal. This makes vector sketch animation particularly valuable for educational demonstrations, artistic communication, and industrial design [FBCW23, HE17, ABR*14, LLGRK20]. Vector sketch generation for static images has been well studied recently. CLIPasso [VPB*22] employs sets of Bézier curves to represent sketches, optimizing control point coordinates using CLIP perceptual loss [RKH*21] and a differentiable rasterizer [LLGRK20]. And CLIPascene [VACOS23] further separates and optimizes foreground and background elements before blending them, achieving full-scene sketch generation. However, applying these methods for frame-by-frame sketch animation generation often results in unpleasant flickering [CZE*23], as it neglects the temporal coherence of stroke topology across frames. To this end, Zheng et al. [ZCXP24] introduced a consistency loss based on Neural Layered Atlas (NLA) [KOWD21] to enhance cross-frame stroke stability; and Fang et al. [FC25] built upon CLIPascene and Zheng et al.'s work, employing Content Deformation Fields (CoDeF) [OWX*24] to further improve temporal coherence in full-frame sketch animations. However, these methods have several limitations:

Temporal popping and jitter are still observed in their results, since the consistency losses employed by Zheng et al. [ZCXP24] and Fang et al. [FC25] can not strictly enforce continuity of stroke motion. Moreover, their ineffectiveness against minor temporal fluctuations makes jitter between adjacent frames an intractable issue.

Local semantics are presented in strokes, because only nearby frames are considered during the optimization process. The weak association of strokes across frames loses the full leverage of the semantic information from the entire video.

Video length is very limited, as their neural representations (e.g., deformation fields) fail to maintain spatial fidelity over long videos, resulting in matching errors and temporal incoherence in the generated sketch animation [FC25].

To tackle these issues, a novel Differentiable Motion Trajectories (DMT) representation is proposed to enforce the temporal coherence of strokes across frames. DMT represents vector sketch animations as a process driven by control points moving along continuous motion trajectories, thus **fully avoiding temporal popping and jitter**. Motion trajectories of control points are modeled as polynomials with Bernstein bases instead of the traditional power bases, enhancing learning stability. DMT enables the conversion of low-framerate input videos into high-framerate sketch animations, achieving video frame interpolation effects.

To support long videos, an explicit representation of video spatial information via sparse tracking points is adopted, instead of previous implicit representations using neural networks. These tracking points can not only be estimated from real videos using various computer vision techniques but also be precisely obtained from 3D model node projections when the input is a 3D animation, offering excellent editability and compatibility. The proposed approach was evaluated on two widely-used datasets: DAVIS [PTPC*17] has short videos with 50-100 frames, and LVOS [HLC*25] has longer videos with more complex motions.

The main contributions of this paper are as follows:

- A differentiable motion trajectories representation that describes the frame-wise movement of stroke control points using differentiable polynomial-based trajectories is proposed. It significantly improves the temporal coherence and semantic consistency, and produces stable and high-framerate vector sketch animation.
- An end-to-end framework for video-based vector sketch animation generation is presented. It incorporates a consistency loss based on sparse tracking, a stroke initialization strategy guided by video spatial information, and a differentiable rasterizer to produce sketch animations from long-duration videos.

2. Previous work

2.1. Vector sketch generation

Existing style transfer techniques have been developed to generate sketch-style outputs from bitmap images [IZZE17, LLM*19, GSH*19, BSM*13], along with pixel-level aligned approaches designed to directly generate vector sketches from bitmaps [CDAT20, DYH*20, RGLM21]. However, the aforementioned methods suffer from the issue of stroke branching or folding. To address the integration of vector graphics into deep learning pipelines, Li et al. [LLGRK20] proposed a differentiable rendering framework for vector graphics that supports multiple geometric primitives, enabling the seamless incorporation of vector representations into deep models via a differentiable paradigm. Building upon this foundation, CLIPasso [VPB*22] leveraged a pre-trained CLIP model to compute perceptual loss between an input image and a Bézier curve-based generated output, optimizing curve control points through backpropagation to transform real-world images into abstract sketches. Expanding on this line of work, CLIPascene [VACOS23] decomposed input images into foreground and background regions for independent optimization, thereby generating full-frame sketches with tunable levels of abstraction. In contrast to image-driven methods, CLIPDraw [FSW22] focused on text-to-vector-sketch generation, directly producing vector sketches from textual prompts. Recently, diffusion models [RBL*22] have increasingly surpassed CLIP-based approaches for text-to-sketch tasks [XWZ*23, JXA23, XZW*24], with optimization guided by the Score Distillation Sampling (SDS) loss [PJBM22] to enhance generation quality. Notably, SwiftSketch [AFCO*25] extended diffusion-based methods to image-conditioned vector sketch generation, replacing the time-consuming iterative optimization processes of prior works and enabling high-quality sketch production within seconds. Collectively, these advancements establish a solid foundation for exploring vector sketch animation generation—an area that remains an active and potential direction in computer graphics research.

2.2. Vector sketch animation generation

Extending image-based vector sketch generation to multi-frame animation poses the key challenge of maintaining temporal coherence of strokes across frames. One category of methods relies on the spatial structure information from an input video to guide stroke optimization. Zheng et al. [ZCXP24] employed a Neural Layered Atlas (NLA) [KOWD21] to represent video spatial information as multi-plane images, constraining stroke positions via inter-frame point mapping. Fang et al. [FC25] utilized the more advanced Content Deformation Fields (CoDeF) [OWX*24] to achieve stronger consistency. Liv3Stroke [LCKP25] extracted 3D point clouds from videos to guide the optimization of 3D curves. However, these methods struggle with long videos (hundreds of frames) due to limitations of NLA or CoDeF, leading to temporal incoherence.

Another category of methods starts with a first-frame sketch and generates subsequent frames by adapting it. These approaches typically require an initial sketch and text prompts as input. Live-Sketch [GVA*24] and MoSketch [LXF*25] first applied coarse-grained rigid transformations to the object in the first frame using an MLP, then refined the control points using SDS loss [PJBM22]. However, this two-stage optimization struggles with complex motions, thus limiting its flexibility and ability to express complex motion semantics. GroupSketch [LHX*25] involved users grouping sketch elements and setting keyframes; after generating motion trajectories via interpolation, it refined the motion using a Group-based Displacement Network (GDN). FlipSketch employed DDIM inversion [SME20] to extract visual features from an input sketch, which were then fed into a Text-to-Video (T2V) diffusion model [WYC*23] fine-tuned on data synthesized by Live-Sketch to generate raster animations. Rai et al. [RS24] introduced a length-area regularization to enhance temporal coherence by estimating smooth motion of control points. The dependence on an initial sketch input, as well as the difficulty in representing complex motions, limits these methods to generating short animations or those with a limited range of motion.

2.3. Animation driven by control points

Control point-driven animation is a widely used technique in computer graphics that efficiently animates complex models or images through smooth deformations and motion by manipulating sparse control points [DRVDP15]. It is extensively applied in visual effects, game animation, vector illustration, and medical and scientific visualization [CWX*15, SKC01, YYF*25]. Traditional methods often rely on artists manually editing control points or extracting motion trajectories from videos, after which the computer automatically generates the deformation results [DRVDP15, SBF*18, AHSS04, BLCD02, GRGC15, SCBS13] .

With the advancement of deep learning, predicting motion trajectories and generating vivid animations without user intervention has become feasible [HFW*22, LZT*19, JLJ*20]. Animation Drawing [SZL*23] utilized predefined character skeletal motions to animate characters drawn by children. AnaMoDiff [TWW*24] predicted optical flow fields from a reference video and applied the resulting deformations to an initial image. Siarohin et al. [SLT*19] generated image animations by applying local affine transformations to an initial image based on keypoints learned from videos. AniClipart [WSML25] and FlexiClip [Kha25] model control point trajectories via Bézier curves, leverage SDS for animated clipart generation, require the first frame as input, and optimize solely for motion trajectory smoothness without explicit semantic consistency constraints. These approaches inspire us to explore control point-driven animation, which can represent complex motions with a compact set of parameters.

3. Differential Motion Trajectories (DMT)

This section introduces the theory of differentiable motion trajectories and their representations suitable for deep learning.

3.1. The DMT definition

The mathematical definition of differential motion trajectories is first introduced in this section. Taking video-to-sketch animation as an example, which involves converting a video of continuous motion into a sketch animation composed of 2D vector curves in each frame. Due to the temporal coherence requirement, vector curves appearing in consecutive frames should transform continuously.

Each frame in a video has N_s strokes, one stroke can be represented as a Bézier curve with $m+1$ control points $\{P_0, P_1, ..., P_m\}$:

$$C(u) = \sum_{i=0}^{m} B_{m,i}(u) P_i \tag{1}$$

where $B_{m,i}(u) = \binom{m}{i} u^i (1-u)^{m-i}$ is the Bernstein basis function [Far12].

The key insight is that any continuously varying Bézier curve can be approximated by a dynamic Bézier curve whose trajectories of control points $\{P_i\}$ are polynomial functions of time t. We call trajectories of these control points as **Motion Trajectory**. As proven in Appendix A, if a Bézier curve changes continuously over time, then its control points must also vary continuously, i.e. its corresponding motion trajectories are continuous functions. By the Weierstrass approximation theorem [Sto48], these motion trajectories can be approximated well by polynomials with sufficient degrees. Thus, it is straightforward to represent a motion trajectory of a control point P as a polynomial:

$$P(t) = \sum_{i=0}^{n} t^i k_i \tag{2}$$

Integrating DMT into the automatic sketch animation generation pipeline allows the semantic loss for a single frame to backpropagate not only to the control point parameters of that frame's strokes but also to the global parameters of DMT. **It encodes global video semantic information into strokes of each frame,** allowing to represent complete semantics with a small number of strokes.

Since polynomial functions $P(t)$ are infinitely differentiable, we call trajectories of control points **Differential Motion Trajectories (DMT in short)**. The DMT continuity enforces the temporal coherence of strokes across frames. When DMT is used in the differentiable optimization process for 2D vector graphics, it allows gradients, which are originally passed to the coordinates of the control point, to be further back-propagated to the set of polynomial parameters $\{k_i\}$. This mechanism enables local loss optimization on a single frame to influence the global state of the entire sketch animation. **It encodes global video semantic information into strokes of each frame,** ensuring that the generated vector graphics exhibit semantic consistency across frames and allowing to represent complete semantics with a small number of strokes.

3.2. Bernstein basis representation

The parameter set $\{k_i\}$ in Equation 2 can be optimized by machine learning methods. However, the power basis representation suffers from uneven sensitivity across the temporal domain, which adversely affects gradient-based optimization [GBCB16].

The L1 norm of the sensitivity for the power basis form is:

$$\|S(t)\|_1 = \sum_{i=0}^{n} \left| \frac{\partial P(t)}{\partial k_i} \right| = \sum_{i=0}^{n} t^i \tag{3}$$

This sensitivity varies dramatically with t. For example, $\|S(0)\|_1 = 1$ when $t = 0$, while $\|S(1)\|_1 = n+1$ when$t = 1$. This variation causes two optimization challenges:

Gradient vanishing: When t is small, the sensitivity approaches 1, resulting in minimal gradient magnitudes that hinder parameter updates for early frames.

Gradient explosion: When t is large, the sensitivity grows exponentially, causing overlarge gradients and training instability.

To address these issues, a Bernstein basis representation is adopted [ZZLZ15, See04]:

$$P'(t) = \sum_{i=0}^{n} B_{n,i}(t) \cdot q_i \tag{4}$$

It achieves uniform sensitivity across the temporal domain:

$$\|S'(t)\|_1 = \sum_{i=0}^{n} \left| \frac{\partial P'(t)}{\partial q_i} \right| = \sum_{i=0}^{n} B_{n,i}(t) = 1 \tag{5}$$

This constant sensitivity ensures stable gradient flow throughout optimization, mitigating the risks of vanishing/exploding gradients and enabling more reliable convergence [HTF*09, DBDB78]. A corresponding ablation study can be found in Section 5.5.1.

4. Methodology

4.1. Overview

Figure 2 shows the framework of our vector sketch animation generation method based on Differentiable Motion Trajectories (DMT). It first extracts tracking information of sparsely sampled points from the input video using computer vision algorithms or from 3D animations using graphic rendering. Subsequently, parameters of DMT corresponding to the vector strokes and their control points are initialized.

During the iterative optimization process, a differentiable rasterizer [LLGRK20] is used to render vector graphics into images frame by frame, and the semantic loss and the geometric loss between the generated results and the original video frames are computed using the CLIP model [RKH*21]. And the temporal consistency loss is constructed to leverage the video's spatial information. By back-propagating losses to the motion trajectory parameters, trajectory functions are jointly optimized, ultimately producing a sketch animation with visual coherence and semantic consistency.

By representing the vector animation as the evolution of Bézier curve control points along continuous motion trajectories, the generated results are naturally continuous between frames, fundamentally avoiding jitter and flickering artifacts. During optimization, the semantic loss from each frame can be back-propagated to the global motion trajectory parameters, enabling each frame's strokes to encapsulate the semantic information of the entire video. Furthermore, the introduced consistency loss constrains the position of the same stroke across various frames of the video, aligning it with the original video content, thereby enhancing the temporal semantic consistency and spatial stability of the strokes.

4.2. Initialization

Motion-aware probability density map . CLIPasso [VPB*22] obtains initial coordinates of strokes by randomly sampling the attention map, which is suitable for static images. However, for dynamic videos, this approach often fails to allocate enough strokes to depict moving regions. Therefore, motion magnitude must be considered besides semantic attention and integrated into the probability density map. To this end, a motion weight for the j-th sample point is first computed:

$$V_m(j) = \left(\sum_{i=1}^{N_f-1} \|TRACK_{i,j} - TRACK_{i-1,j}\| \right)^{1/2} \tag{6}$$

where N_f is the number of video frames, and $TRACK_{i,j}$ denotes the 2D coordinates of the j-th tracked sample point in the i-th frame. These tracks can be obtained via estimation from video or accurately acquired from 3D animation (see Section 4.4 for details).

$V_m(j)$ is defined on sparse tracking positions, a radial basis function interpolation is then employed to obtain a per-pixel motion weight across the entire image, which is further normalized to the range [0, 1] to form the motion heatmap M_{motion}. It first blends with the CLIP attention map $M_{attention}$ linearly, and then multiplies with an XDoG edge map [WKO12] to encourage initial stroke placement near object contours. The final probability density map is:

$$M = M_{XDoG} \otimes \left((1-\beta) \cdot M_{attention} + \beta \cdot M_{motion} \right) \tag{7}$$

where $\otimes$ is the Hadamard product, and $\beta = 0.5$ in all experiments.

Initial stroke coordinates are sampled according to this probability density distribution, favoring regions with higher density [Fis22]. An ablation study in Section 5.5.2 compares results with and without the motion heatmap.

The DMT initialization. The degree of the time polynomial n is set to $N_f/4$ as the default. Users can adjust this parameter based on the smoothness of the target motion. To avoid local minima and accelerate convergence, we initialize strokes such that their trajectories roughly approximate the true object motion. Instead of randomly selecting pixel positions as in CLIPasso, a set of points (equal to the number of strokes) is randomly sampled from the sparsely tracked points with motion trajectories. These sampled trajectories serve as fitting targets for the initial strokes.

For each motion trajectory, a polynomial should be fitted with a Bernstein basis. Three fitting methods are evaluated: polynomial interpolation, least squares, and ridge regression. Considering the fitting error and numerical stability, the ridge regression is ultimately adopt for polynomial fitting. A comparison with these three methods can be found in Appendix B.

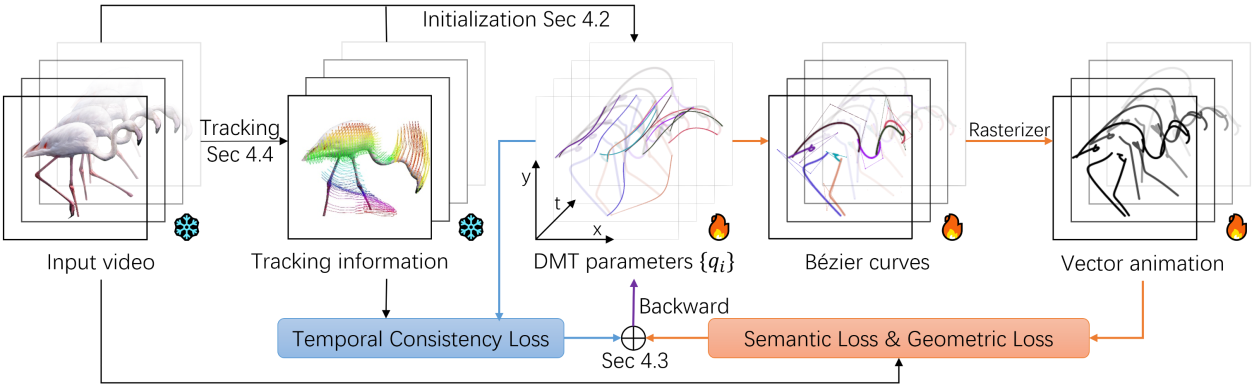


Figure 2: *The framework of our DMT-based vector sketch animation generation. First, we obtain the tracking information from the video and initialize the DMT parameters. Then, these parameters are iteratively optimized to make the rasterized sketch animation semantically and geometrically close to the input video, and the stroke movement trajectories are consistent with the tracking information.*

Stroke width calculation. Inspired by SketchVideo [ZCXP24], the stroke width per frame should be adapted to the relative size of the target object to prevent artifacts. When the object is small, thinner strokes are used to avoid unexpected overlap of thick strokes in dense areas; when the object is large, thicker strokes are employed to minimize blank areas.

The stroke width for the i-th frame, $width_i$, is adapted according to the area of the object mask (generated by a UNet2 network [QZH*20]) relative to the image area:

$$width_i = W_{max} \cdot \sqrt{Area_i/(W \cdot H)} \tag{8}$$

where $W_{max} = 3$ denotes the maximum stroke width, $Area_i$ is the pixel count of the target object in the mask for the i-th frame, and W and H are the image width and height respectively.

4.3. Loss functions

We propose a comprehensive loss function: the semantic and geometric losses are derived from CLIPasso [VPB*22], while a temporal constraint is newly added to our design. The total loss is defined as:

$$\mathcal{L}_{total} = w_s \cdot \mathcal{L}_{sem} + w_g \cdot \mathcal{L}_{geo} + w_c \cdot \mathcal{L}_{cons} \tag{9}$$

where w_s, w_g and w_c are weighting coefficients. In optimization, they are set to 0.1,1 and 2, respectively.

Semantic loss. The final layer of the CLIP encoder captures high-level semantic information. The semantic loss $\mathcal{L}_{sem}$ is defined as the cosine-based distance between the CLIP embeddings of the rasterized vector graphics and the input video frames:

$$\mathcal{L}_{sem} = \sum_{i=0}^{N_f-1} dist\left(CLIP\left(R(\mathbb{C}(i))\right), CLIP(I_i)\right) \tag{10}$$

where $\mathbb{C}(i)$ denotes the set of Bézier curves at frame i, $R(\mathbb{C}(i))$ is the image rendered by the differentiable rasterizer, I_i is the i-th input video frame, and $dist(x,y) = 1 - \frac{x \cdot y}{\|x\| \|y\|}$ computes the cosine-based distance.

Geometric Loss. Intermediate layers of the CLIP network contain more spatial and geometric information compared to the final layer. Thus, a geometric loss $\mathcal{L}_{geo}$ is defined as the L2 distance between intermediate feature activations:

$$\mathcal{L}_{geo} = \sum_{i=0}^{N_f-1} \sum_{l \in \{3,4\}} \left\| CLIP_l\left(R(\mathbb{C}(i))\right) - CLIP_l(I_i) \right\|^2 \tag{11}$$

where $CLIP_l$ is the CLIP encoder activation at layer l.

Both $\mathcal{L}_{sem}$ and $\mathcal{L}_{geo}$ are derived from the CLIP model and can be computed efficiently in a single forward pass (implementation details in Section 4.5.2).

Temporal consistency loss. To ensure stroke motion follows the underlying video motion, a temporal consistency loss is introduced, which is the consistency sum of all strokes in all frames:

$$\mathcal{L}_{cons} = \sum_{i=0}^{N_f-1} \sum_{j=1}^{N_s} \mathcal{L}_{cons}^{stroke}(i,j) \tag{12}$$

where N_s is the number of strokes, and $\mathcal{L}_{cons}^{stroke}(i,j)$ measures the consistency for the j-th stroke at the i-th frame.

For computational efficiency, stroke consistency is computed by uniformly sampling N_p points along each Bézier curve:

$$\mathcal{L}_{cons}^{stroke}(i,j) = \frac{1}{N_p} \sum_{k=0}^{N_p-1} \mathcal{L}_{cons}^{point}\left(i,j,u = k/(N_p-1)\right) \tag{13}$$

The point-wise consistency loss $\mathcal{L}_{cons}^{point}(i,j,u)$ measures the average L2 distance between the position of the sampled point across all frames and its corresponding position along the tracked trajectory:

$$\mathcal{L}_{cons}^{point}(i,j,u) = \frac{1}{N_f} \sum_{t=0}^{N_f-1} \|\mathbf{T}(\mathcal{C}(i,j,u),i,t) - \mathcal{C}(t,j,u)\|^2 \tag{14}$$

where $\mathcal{C}(i,j,u)$ represents the 2D coordinates of the j-th Bézier curve at parameter u in frame i, and $\mathbf{T}(\mathbf{p},i,t)$ predicts the position of pixel $\mathbf{p}$ from frame i to frame t using the video motion trajectory. Since we only have sparse trajectory data, an interpolation scheme is employed, whose details can be found in Section 4.5.3.

4.4. Tracking information

Existing approaches typically employ neural-based methods such as Neural Layered Atlas (NLA) [KOWD21] or Content Deformation Fields (CoDeF) [OWX*24] to maintain temporal consistency of strokes. However, these methods suffer from two main limitations: 1) These methods are difficult to processing long videos accurately and will increasing the inconsistency when the number of frames is over 100. [FC25]; 2) the encoded information is embedded within neural network weights, lacking interpretability and editability, which restricts practical application flexibility.

To address these issues, an explicit representation of video spatial information based on sparse sample point trajectories is proposed. Our method uniformly samples a moderate number of feature points (typically 2,000 to 10,000) from the video and tracks their positions across frames. Temporal consistency of sketch strokes is enforced by measuring the similarity between a stroke's motion trajectory and those of nearby sampled points.

Sparse tracking for natural videos. For natural videos, CoTracker [KMW*24] is employed as our base tracker and enhances its memory management to support long video sequences. Implementation details are discussed in Section 4.5.2.

Sparse tracking for 3D animation. For 3D animation input, vertices on the model surface from each sequence frame are uniformly sampled, and their screen-space coordinates are recorded. To ensure strict data alignment, a customized rendering script is used to synchronously capture the frame and record vertex coordinates after each frame is rendered, avoiding inter-frame deviations caused by GPU-CPU asynchronous execution in real-time rendering.

4.5. Implementation details

4.5.1. Bernstein polynomials with high degrees

Representing complex motion trajectories requires higher-degree polynomials. For Bernstein basis functions $B_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}$, when n is large ($n > 100$), the binomial coefficient $\binom{n}{i}$ can become extremely large while t^i or $(1-t)^{n-i}$ becomes extremely small, leading to numerical precision issues with PyTorch's default 32-bit floating-point arithmetic. While switching to 64-bit precision provides some relief, it becomes insufficient as n increases further.

Considering that the value of $P_x(t)$ should not exceed the image dimensions and does not require high decimal precision, a logarithmic approach is employed. $\log\binom{n}{i}$, $\log(t^i)$, and $\log((1-t)^{n-i})$ are computed first, then sum them to obtain $\log(\binom{n}{i} t^i (1-t)^{n-i})$, and finally convert back to the non-logarithmic value. This logarithmic strategy maintains sufficient precision with 32-bit floating-point arithmetic while supporting much larger values of n.

4.5.2. Memory optimization for long videos

Long video generation and analysis are often constrained by memory limitations [OWX*24, YZAS21, LNC*22]. To this end, a memory-efficient optimization strategy is designed for both video spatial information extraction and vector animation generation stages. The related experimental results are shown in Table 1.

CoTracker inference optimization. The original CoTracker [KMW*24] maintains the state of all frames and all tracked points in GPU memory, leading to linear memory growth with sequence length, and thus can only process videos with fewer than 100 frames. To overcome this limitation, the tracking pipeline is redesigned to store intermediate features and inactive point cloud data in host memory (CPU RAM), transfer data to GPU only during essential computations such as optical flow propagation and feature matching. This heterogeneous storage strategy significantly reduces peak GPU memory usage, enabling sparse tracking for videos with more than 800 frames.

Multi-frame gradient accumulation optimization. In gradient-based video optimization, the total loss depends on accumulated errors across all frames. Direct backpropagation leads to memory usage proportional to the number of frames. Profiling reveals that CLIP-based losses (e.g., $\mathcal{L}_{geo}$ and $\mathcal{L}_{sem}$) constitute the primary memory bottleneck. Theoretical analysis shows their gradients are decomposable into frame-independent terms. We therefore adopt an alternating computation-and-release gradient accumulation scheme: gradients are computed and accumulated per frame while immediately freeing intermediate variables before processing the next frame. This reduces memory complexity from $O(T)$ to $O(1)$, supporting stable optimization for long videos.

4.5.3. Sparse-to-dense tracking information conversion

To balance system compatibility, editability, and computational efficiency, our method explicitly represents video spatial information using sparse tracking points. To recover per-pixel motion from this sparse representation, a motion propagation mechanism based on spatial smoothness priors is employed. Specifically, for any pixel $\mathbf{p}$ in frame i, its corresponding position in frame t, is given by

$$\mathbf{T}(\mathbf{p},i,t) = \mathbf{p} - \mathbf{N}(\mathbf{p},i) + \mathbf{T}_R(\mathbf{N}(\mathbf{p},i),t) \tag{15}$$

where $\mathbf{N}(\mathbf{p},i)$ represents the nearest sample point to $\mathbf{p}$ in frame i, and $\mathbf{T}_R(\mathbf{s},t)$ denotes the coordinates of tracked sample point $\mathbf{s}$ in frame t. This method efficiently estimates full-frame motion while preserving the advantages of sparse representation.

5. Results

This section presents a comprehensive evaluation of our proposed method through multiple experiments, including: comparisons with SOTA methods (Sec 5.1), experiments on longer videos (Sec 5.2) and 3D animation-to-sketch conversion (Sec 5.3), vector animation generation and frame interpolation from text-to-video models (Sec 5.4), multiple ablation studies (Sec 5.5). Performance (Sec 5.6) and limitations (Sec 5.8) are discussed in the end. Since our method produces dynamic vector animation sequences whose motion effects are difficult to fully capture in static images, readers are referred to the supplementary materials for complete dynamic comparisons.

5.1. Comparisons with SOTA methods

Our approach is compared with SOTA methods, including detection-based methods (Canny [Can09] and HED [XT15]), image-based sketching (CLIPasso [VPB*22]), video-specific

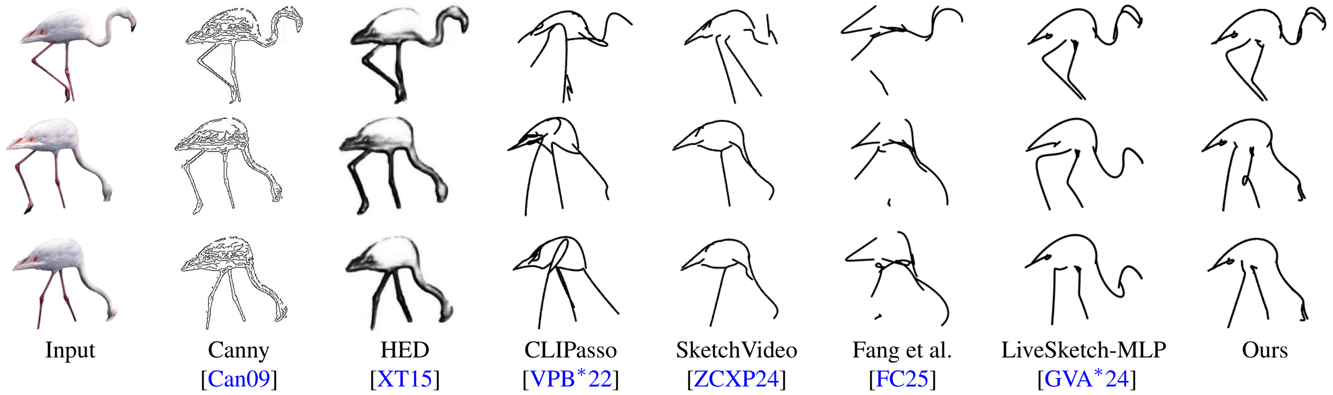


Figure 3: ***Comparisons with SOTA methods with 3 frames (the first, 25th, and 49th frames from top to bottom).*** *Canny and HED do not produce vector graphics. CLIPasso, SketchVideo, and Fang et al. produce competitive results, but suffer from the temporal incoherence issue, such as legs and the tail. LiveSketch-MLP fails to capture the bent posture of the flamingo's head. The results produced by our approach achieve superior temporal coherence and semantic consistency.*

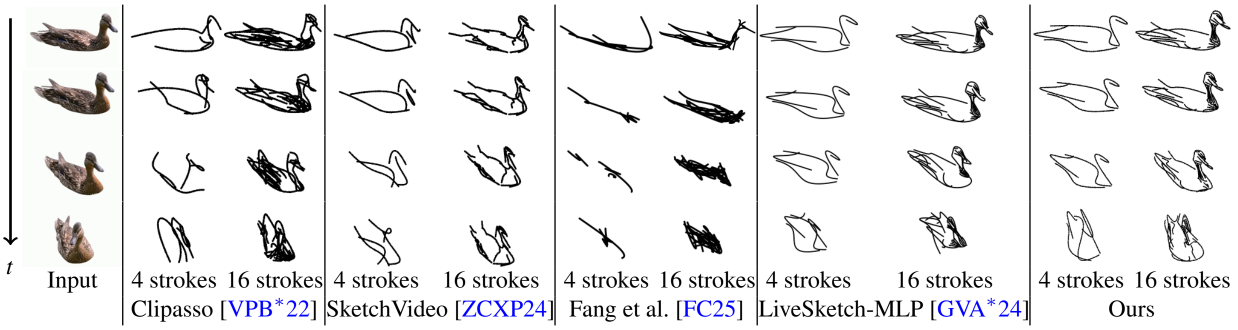


Figure 4: ***Comparisons of 5 SOTA methods with different numbers of strokes.*** *CLIPasso lacks temporal coherence; CLIPasso, SketchVideo and Fang et al. suffer from semantic loss with a small number of strokes. LiveSketch-MLP shows less cross-frame semantic consistency, particularly with increasing frame index t.*

sketch generation (SketchVideo [ZCXP24] and Fang et al. [FC25]), and a modified variant derived from LiveSketch's MLP architecture [GVA*24], namely LiveSketch-MLP.

The original LiveSketch takes the first frame of a pre-existing sketch animation and a text prompt as inputs [GVA*24], which deviates from the unified setting of our experiment (video input, sketch animation output). To resolve this discrepancy, we adapt the core MLP architecture of LiveSketch, initialize its first frame with the output of the first-frame generation module in our approach, and integrate it with the proposed loss function to construct LiveSketch-MLP. This modification ensures compliance with the input-output protocol of our experiment while preserving the architectural essence of the original model.

As shown in Figure 3, Canny and HED produce results that resemble edge maps rather than semantically abstract sketches. While CLIPasso generates strokes with reasonable abstraction, its results exhibit poor temporal consistency, leading to noticeable visual flickering across frames. SketchVideo achieves better temporal coherence, but still suffers from inter-frame stroke jittering, particularly in regions with complex dynamic structures. VideoSketch proposed by Fang et al. demonstrates improved flicker suppression, yet shows instability in detailed areas (e.g., the flamingo's tail in Figure 3). Note that VideoSketch was applied to the entire image; only the foreground is shown for comparison purposes. LiveSketch-MLP suffers from inter-frame jittering and exhibits semantic inconsistency, failing to capture the bent posture of the flamingo's head. In contrast, our method produces animation sequences without visible flickering across all frames, demonstrating superior temporal coherence and semantic consistency.

Figure 4 further compares results with varying numbers of strokes. Both CLIPasso and SketchVideo suffer from significant semantic loss with a small number of strokes (e.g., 4 strokes) and tend to fail to capture the original image content. LiveSketch-MLP exhibits semantic inconsistency, especially when the frame index t is large (the bottom line). Our method maintains clear semantic expression and stable visual content even with a small number of strokes, validating the advantage of our DMT representation in cross-frame semantic modeling.

We quantitatively evaluated semantic consistency via LPIPS [ZIE*18] and CLIP similarity [RKH*21], with mean values

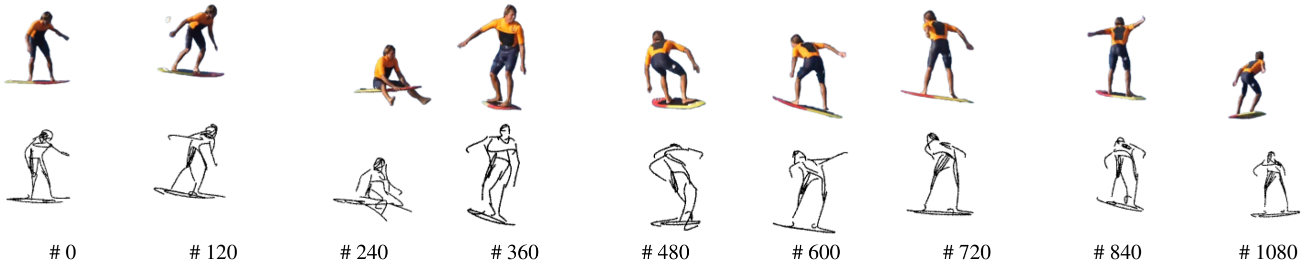


Figure 5: *The result of* **long** *Surfing video [HLC*25]. (6FPS input with 300 frames; 24FPS output with 1200 frames)*

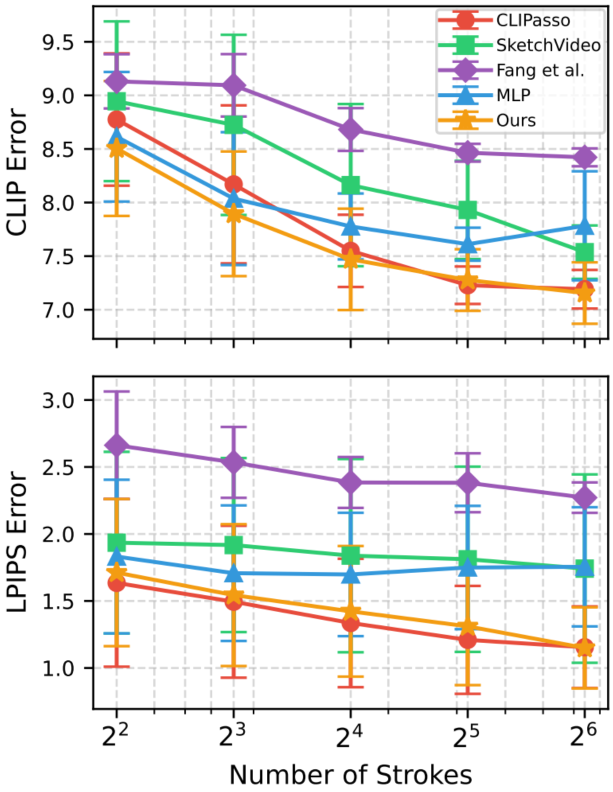


Figure 6: *The statistics of semantic error evaluation.*

and variances presented in Figure 6. Our method achieves comparable LPIPS scores to CLIPasso and outperforms SketchVideo, Fang et al., and LiveSketch-MLP; in the CLIP score, it is comparable to CLIPasso with more strokes and superior with fewer. However, since CLIPasso ignores temporal coherence and exhibits significant flickering, our approach has a clear overall advantage.

5.2. Experiments on longer videos

We conducted experiments on longer video sequences. Figure 5 shows the results of the surfing video with 300 frames and 6FPS from the LVOS dataset [HLC*25]. Additionally, experimental results of another 400-frame video [Yin25] are presented in Figure 13 of the appendix. Existing methods struggle to process such long sequences effectively, often failing due to computational resource constraints. In contrast, our approach achieves stable generation for such long videos by leveraging explicit sparse tracking point trajectories to guide stroke consistency, thus addressing a key limitation of previous methods. It is worth noting that benefiting from the continuity of DMT, our method provides a continuous temporal representation, enabling flexible adjustment of output frame rates, as shown in the surfing result.

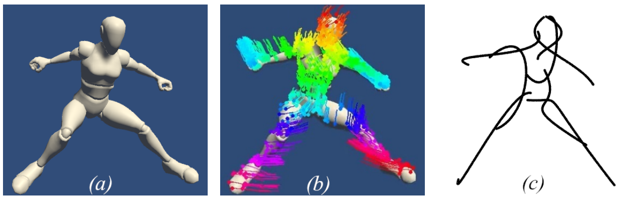
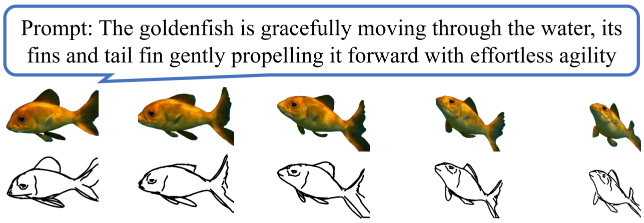


Figure 7: *Our result that converts a 3D animation to a sketch animation. Tracking information (b) is recorded during the 3D animation rendering (a), and (c) is one sketch result at this frame.*



Prompt: The goldenfish is gracefully moving through the water, its fins and tail fin gently propelling it forward with effortless agility

Figure 8: *Our result of text-driven sketch animation generation. The input prompt, the generated video, and the vector sketch animation are shown from top to bottom.*

5.3. 3D animation to sketch animation

To validate the compatibility of our method with 3D animation data, we established a data capture pipeline using the Unity engine. We used a skinned character model [Dou25] with skeletal animations [Igl25], using custom scripts to synchronize screen capture with sparse vertex trajectory recording. The experiment collected 500 frames while maintaining temporal consistency. Figure 7 shows such an example, taking the 3D animation as input.

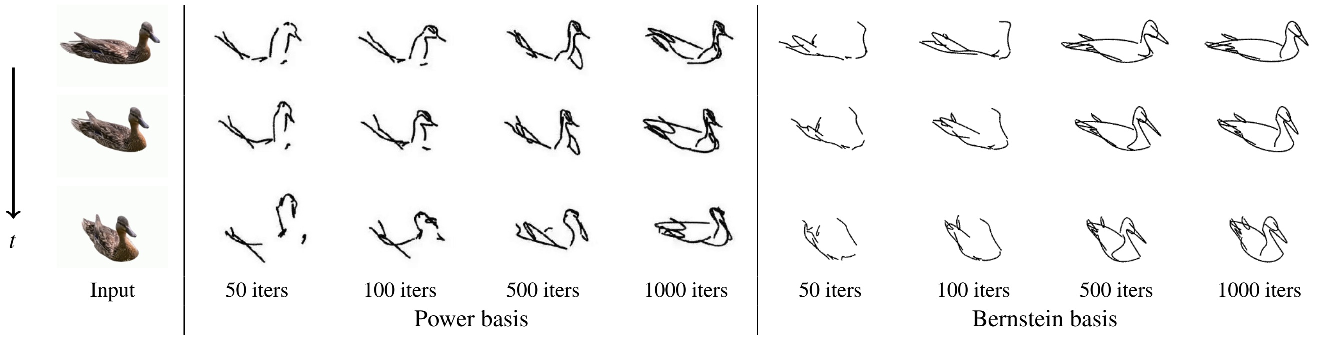


Figure 9: *Comparisons between power basis and Bernstein basis representations of DMT at 3 frames. The power basis shows slower convergence and distortions when t increases, while the Bernstein basis converges faster and avoids distortion artifacts in all frames.*

5.4. Text to sketch animation

We explored text-to-sketch animation generation. We first employ CogVideoX [YTZ*24] (the official CogVideoX-2B checkpoint) to generate an initial video from input text, then extract and crop the foreground object using a U2Net network [QZH*20], and finally convert the foreground video into vector sketch animation using our approach. Unlike methods that require additional sketch input [GVA*24, LXF*25, LHX*25], our approach is based solely on text prompts, as shown in Figure 8. Note that the frame interpolation capability can mitigate the framerate limitations of existing text-to-video models.

5.5. Ablation studies

5.5.1. Power basis vs. Bernstein basis

We conducted ablation studies comparing power basis and Bernstein basis representations for our DMT module. As shown in Figure 9, DMT using a power basis exhibits slower convergence, requiring more than 1,000 iterations to achieve reasonable results. Furthermore, when the parameter *t* approaches larger values (i.e., posterior frames), the power-base representation tends to produce excessive distortions, which can not be solved with more iterations. In contrast, the Bernstein basis representation delivers faster convergence, achieving comparable quality within 500 iterations, and maintains distortion-free results for all frames, demonstrating superior numerical stability and optimization robustness.

5.5.2. Motion heatmaps in initialization

We analyzed the role of motion heatmaps in stroke initialization through ablation studies. Figure 10 shows the impact on the probability distribution of stroke sampling. Without motion heatmaps, regions with frequent motion (e.g., legs) receive low probability density, leading to insufficient sampling when the number of strokes is strictly limited. Incorporating motion heatmaps significantly increases probability density in these regions, enabling better structural depiction even with a small number of strokes.

5.5.3. Loss functions

We evaluated the impact of the proposed temporal consistency loss on the quality of sketch animation results. Without consistency loss, strokes often correspond to different semantic regions across frames, causing severe temporal inconsistency, including semantic confusion and unnatural structural distortions. As shown in Figure 11, flamingo leg strokes may incorrectly align with abdominal regions when consistency constraints are absent. Incorporating the temporal consistency loss maintains proper semantic correspondence across frames, effectively mitigating these issues.

5.6. Performance

Method	frame	GPU Memory
SketchVideo	50	17.2 GB
LiveSketch-MLP	50	4.58 GB
Ours	50 / 400	**2.47 GB / 3.37 GB**
CoTracker (orig.)	50	11.38 GB
CoTracker (opt.)	50 / 400	**5.31 GB / 11.79 GB**

Table 1: *GPU memory consumption is reduced significantly for both sketch animation generation and tracking information computation, allowing our approach to support long videos.*

Although the generated sketch animations are lightweight vector graphics, the generation methods based on pre-trained models (e.g., CLIP, StableDiffusion) are memory-intensive. We evaluated performance on a workstation with an AMD EPYC 7B13 CPU and an NVIDIA GeForce RTX 3090 GPU. In the preprocessing stage, our memory optimization for CoTracker reduces its GPU memory consumption from 11.38 GB to 5.31 GB for a 50-frame video. As shown in Table 1, the proposed generation method further demonstrates significant memory efficiency, requiring only 2.47 GB for 50 frames—an 85% reduction over SketchVideo and a 46% reduction over MLP. Even for long sequences (400 frames), our method maintains high memory efficiency, consuming only 3.37 GB. This enables execution on consumer-grade hardware (e.g., laptops with GTX 1050 Ti Mobile GPU), which previous methods could not support.

5.7. User studies

We recruited 20 volunteers to participate in a pairwise comparison user study, comparing our method, CLIPasso [VPB*22], and

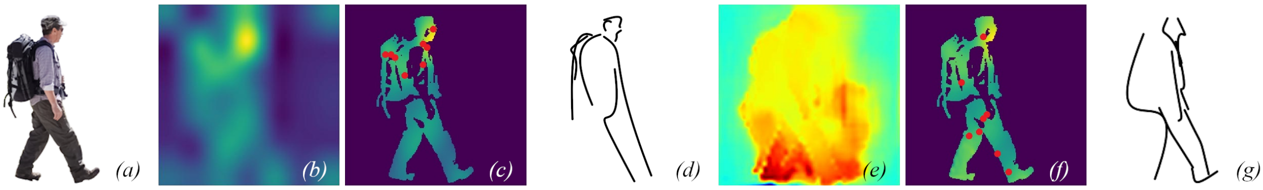
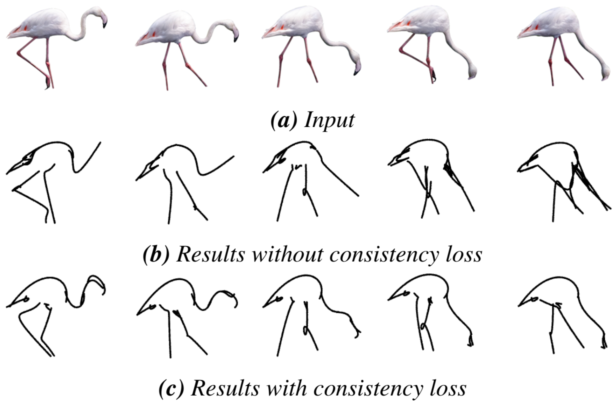


Figure 10: *Comparisons of stroke initialization strategies. (a) First frame of input video; (b) CLIP attention map; (c) Probability density map and sampled points without motion heatmap - insufficient sampling in the leg region; (d) Sketch result without motion heatmap that shows missing leg strokes; (e) Motion heatmap; (f) Probability density map integrated with motion heatmap - increased sampling in the leg region; (g) Our sketch result with motion heatmap, achieving a better depiction of structure.*



(a) *Input*

(b) *Results without consistency loss*

(c) *Results with consistency loss*

Figure 11: *Effect of consistency loss. Without consistency losses, strokes exhibit semantic misalignment across frames (e.g., leg strokes are confused with abdomen). The consistency loss maintains proper semantic correspondence.*

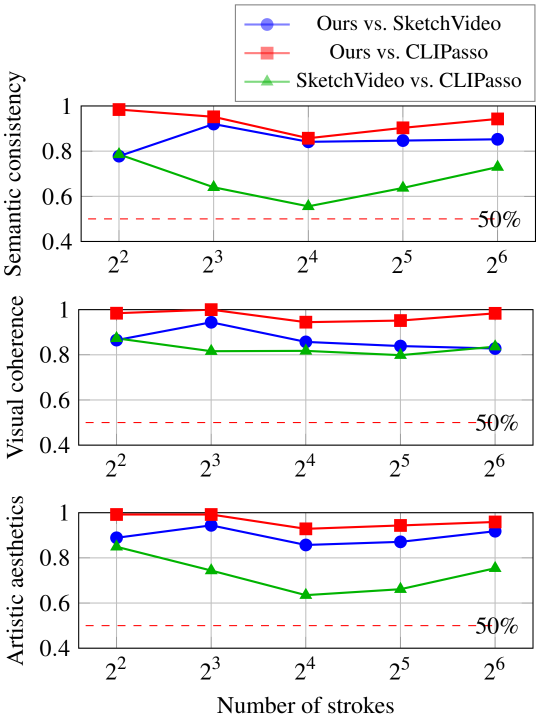


Figure 12: *Pairwise comparison user studies: preference proportion for the former option in forced choices.*

SketchVideo [ZCXP24]. For each input, we displayed the input video alongside output videos produced by two methods in a randomized order. Participants were asked to select the result they preferred in three terms: semantic consistency, temporal coherence, and artistic aesthetics. The statistics, summarized in Figure 12, demonstrate an evident advantage for our method on all three criteria. Notably, this advantage was more pronounced in scenarios with a small number of strokes, highlighting our method's efficacy in creating concise and expressive sketch animations.

5.8. Limitations

Our method exhibits several limitations that point toward valuable future research directions: 1) Our consistency loss relies on tracking information, which may be inaccurate in scenarios with rapid movements or severe occlusion. 2) Our iterative optimization process is computationally expensive; integrating diffusion models may improve efficiency [AFCO*25]. 3) Our approach focuses solely on foreground objects; one future direction would be to extend to full-screen videos [VACOS23, FC25].

6. Conclusion

We present an automatic generation approach for vector sketch animation based on a novel Differentiable Motion Trajectory (DMT), which represents the frame-wise movement of stroke control points using differentiable polynomial-based trajectories. DMT enables global semantic gradient propagation across multiple frames, significantly improving the semantic consistency and temporal coherence, and is compatible with long video inputs and produces highframerate output. Using a Bernstein basis, DMT can be stably optimized by balancing the sensitivity of polynomial parameters. Evaluations on DAVIS and LVOS datasets demonstrate the superiority of our approach over SOTA methods for both short and long videos. The robustness and compatibility of the proposed method are further validated through various ablation studies and successful applications in cross-domain scenarios, such as text-to-video generation and 3D animations. Future research directions could be enhancing generation efficiency, increasing tracking robustness, and extending to various artistic and multimedia applications.

7. Acknowledgments

We would like to express our sincere gratitude to Prof.Xiaonan Fang for generously providing the experimental data of the method [FC25]. We also acknowledge the authors of CLIPasso [VPB*22] and CoTracker [KMW*24] for making their code publicly available. This work is supported by the National Natural Science Foundation of China (Grant No. 62172367).

Appendix A: Feasibility Proof of Polynomial Approximation

This appendix provides the detailed mathematical proof supporting the feasibility of representing control point motion using polynomial functions.

Theorem 1. If a Bézier curve changes continuously over time, then its control points must also vary continuously.

Proof Assuming that the Bézier curve changes continuously over time, let the curve at time t be $C(u,t)$. Since the Bernstein basis functions are linearly independent, the control points $\{P_0, P_1, ..., P_m\}$ can be uniquely determined from the curve $C(u)$.
For a set of distinct values u, e.g. $u_0, u_1, ..., u_m$, if the curve $C(u)$ is continuous with respect to t for each u, then the corresponding points on the curve $C(u_0), C(u_1), ..., C(u_m)$ are also continuous with respect to t. These points on the curve are linear combinations of the control points, which we can express as:

$$\begin{bmatrix} C(u_0,t) \\ C(u_1,t) \\ \vdots \\ C(u_m,t) \end{bmatrix} = \begin{bmatrix} B_{m,0}(u_0) & B_{m,1}(u_0) & \cdots & B_{m,m}(u_0) \\ B_{m,0}(u_1) & B_{m,1}(u_1) & \cdots & B_{m,m}(u_1) \\ \vdots & \vdots & \ddots & \vdots \\ B_{m,0}(u_m) & B_{m,1}(u_m) & \cdots & B_{m,m}(u_m) \end{bmatrix} \cdot \begin{bmatrix} P_0(t) \\ P_1(t) \\ \vdots \\ P_m(t) \end{bmatrix}$$

Abbreviated as:

$$\mathbf{C}(t) = \mathbf{M} \cdot \mathbf{P}(t)$$

Here, $\mathbf{M}$ is an invertible matrix (when the u_k are chosen appropriately), allowing us to solve for the control points:

$$\mathbf{P}(t) = \mathbf{M}^{-1} \cdot \mathbf{C}(t)$$

Therefore, if the Bézier curve is continuous in t, its control points must also be continuous in t. □

Theorem 2. A continuously changing Bézier curve can be arbitrarily approximated by a Bézier curve whose control points are polynomial functions of time t.

Proof For an arbitrary control point P, its 2D coordinate vector at time parameter $t \in [0,1]$ can be represented by a function $P(t)$, which is continuous in t. According to the Weierstrass approximation theorem [Sto48], any continuous function can be arbitrarily approximated by polynomials. Thus, for each control point $P_i(t)$, there exists a polynomial function $Q_i(t)$ (with time t as the variable) such that $|P_i(t) - Q_i(t)| < \epsilon$ holds for all $t \in [0,1]$, where ϵ is an arbitrarily small positive number.
Define the approximate Bézier curve represented by $Q_i(t)$ as $D(u,t) = \sum_{i=0}^{m} B_{m,i}(u) Q_i(t)$. Since the Bernstein basis satisfies $\sum_{i=0}^{m} B_{m,i}(u) = 1$ and is non-negative, for any $u, t \in [0,1]$, we have:

$$\begin{aligned} |C(u,t) - D(u,t)| &= \left| \sum_{i=0}^{m} (P_i(t) - Q_i(t)) B_{m,i}(u) \right| \\ &\leq \sum_{i=0}^{m} |P_i(t) - Q_i(t)| B_{m,i}(u) \\ &\leq \epsilon \sum_{i=0}^{m} B_{m,i}(u) = \epsilon. \end{aligned}$$

This means the error between the approximate curve $D(u,t)$ and the original curve $C(u,t)$ does not exceed ϵ for any $u, t \in [0,1]$. Therefore, a continuously changing Bézier curve can be arbitrarily approximated by a Bézier curve whose control points are polynomial functions of time t. □

Appendix B: Stroke trajectory initialization

We compare three fitting strategies for initializing stroke control point motion trajectories: uniform sampling, least squares, and ridge regression.

polynomial interpolation selects $n+1$ frames from the video, uses normalized time t_i as nodes and solves linear equations to ensure the trajectory passes exactly through sample points in these frames. Though fitting is accurate at sampled frames, fitting error increases significantly at non-sampled frames when $n > 15$.

Least squares minimizes the sum of squared errors for polynomial coefficient estimation. However, when polynomial degree is high ($n > 40$), the Vandermonde matrix becomes ill-conditioned [Bel78], making solutions highly sensitive to noise and numerically unstable.

Ridge regression introduces L2 regularization [HK70] to address ill-conditioning in high-degree polynomial fitting ($n > 200$). Although Mean Absolute Error (MAE) is similar between least squares and ridge regression, the former produces coefficients with extremely large magnitudes, leading to oscillatory behavior and overfitting. Ridge regression constrains coefficient magnitudes, yielding stable and smooth trajectories.

Table 2 compares the three methods on the complex dance video under varying frame counts and polynomial degrees. Frame sampling shows large errors and coefficient magnitudes. Least squares achieves lower MAE but exhibits unstable trajectories due to large coefficients. Ridge regression maintains acceptable MAE (maximum 1.433 pixels) while keeping coefficient magnitudes around 10^4, demonstrating significantly better stability.

References

[ABR*14] Aubert M., Brumm A., Ramli M., Sutikna T., Saptomo E. W., Hakim B., Morwood M. J., van den Bergh G. D., Kinsley L., Dosseto A.: Pleistocene cave art from sulawesi, indonesia. *Nature 514*, 7521 (2014), 223–227. 2

[AFCO*25] Arar E., Frenkel Y., Cohen-Or D., Shamir A., Vinker Y.: Swiftsketch: A diffusion model for image-to-vector sketch generation. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers* (2025), pp. 1–12. 2, 10

frames	Max degree	Mean absolute error			Avg of abs values of coefficients		
		Frame Sampling	Least Squares	Ridge Regression	Frame Sampling	Least Squares	Ridge Regression
50	24	$2.849 \cdot 10^{2}$	$5.687 \cdot 10^{-2}$	$7.782 \cdot 10^{-2}$	$1.588 \cdot 10^{9}$	$1.510 \cdot 10^{7}$	$1.744 \cdot 10^{4}$
100	49	$1.485 \cdot 10^{14}$	$6.293 \cdot 10^{-2}$	$1.695 \cdot 10^{-1}$	$1.182 \cdot 10^{18}$	$1.699 \cdot 10^{13}$	$1.930 \cdot 10^{4}$
200	99	$5.857 \cdot 10^{42}$	$8.988 \cdot 10^{-2}$	$5.231 \cdot 10^{-1}$	$6.264 \cdot 10^{44}$	$3.425 \cdot 10^{13}$	$1.583 \cdot 10^{4}$
400	199	$8.320 \cdot 10^{127}$	$1.727 \cdot 10^{-1}$	1.433	$8.608 \cdot 10^{129}$	$2.936 \cdot 10^{13}$	$2.004 \cdot 10^{4}$

Table 2: *Comparison of trajectory fitting methods on complex motion video. Ridge regression provides the best trade-off between accuracy and stability.*

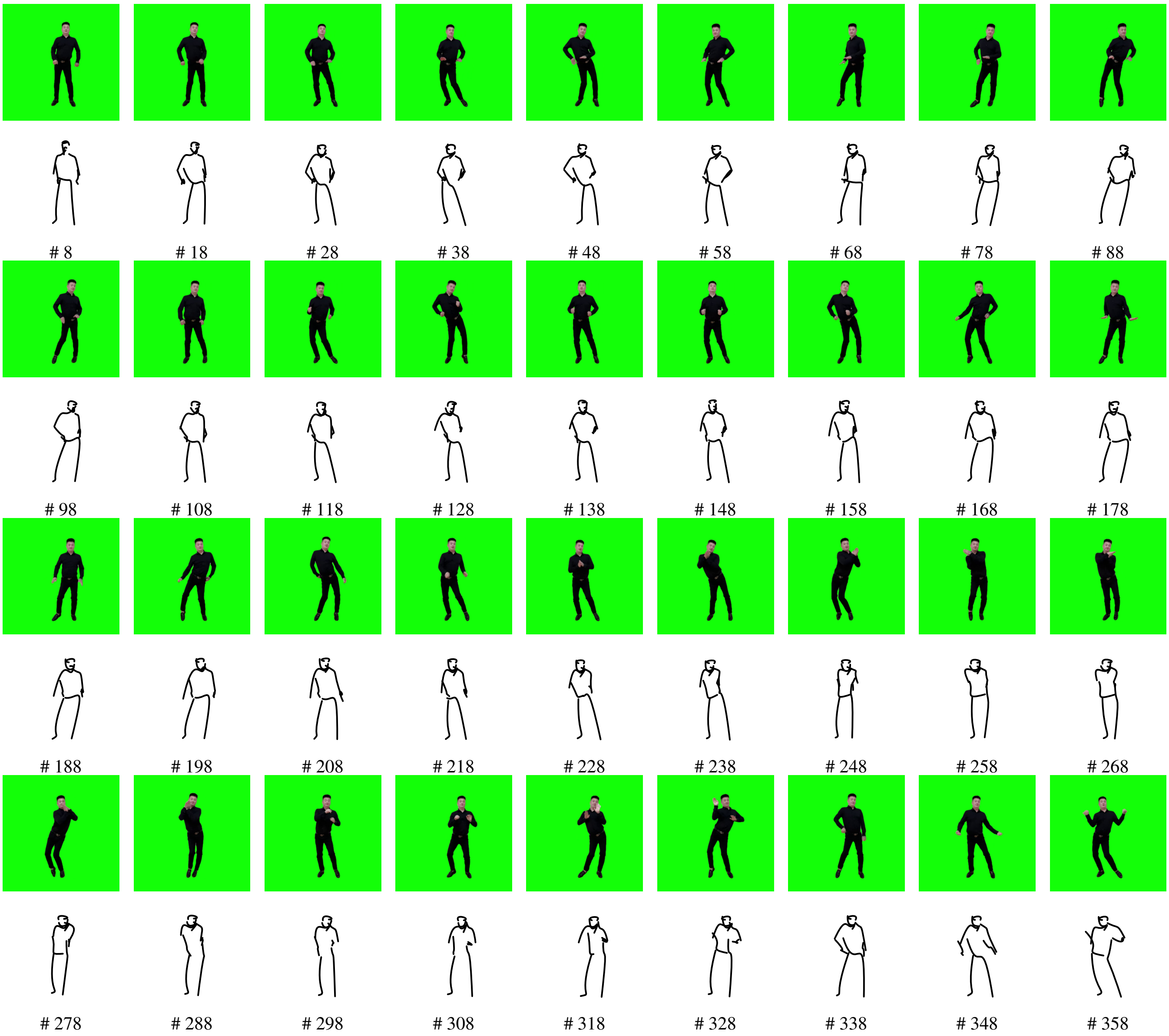


Figure 13: *The result of* ***long Dance video*** *with 400 frames [Yin25]*

[AHSS04] Agarwala A., Hertzmann A., Salesin D. H., Seitz S. M.: Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics (ToG) 23*, 3 (2004), 584–591. 3

[Bel78] Bell J. B.: Solutions of ill-posed problems., 1978. 11

[BLCD02] Bregler C., Loeb L., Chuang E., Deshpande H.: Turning to the masters: Motion capturing cartoons. *ACM Transactions on Graphics (TOG) 21*, 3 (2002), 399–407. 3

[BOH11] Bostock M., Ogievetsky V., Heer J.: D^3 data-driven documents. *IEEE transactions on visualization and computer graphics 17*, 12 (2011), 2301–2309. 2

[BSM*13] Berger I., Shamir A., Mahler M., Carter E., Hodgins J.: Style and abstraction in portrait sketching. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–12. 2

[Can09] Canny J.: A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, 6 (2009), 679–698. 6, 7

[CDAT20] Carlier A., Danelljan M., Alahi A., Timofte R.: Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems 33* (2020), 16351–16361. 2

[CWX*15] Cui Q., Wu Z., Xing C., Zhou Z., Wu W.: Target temperature driven dynamic flame animation. In *Eurographics (Short Papers)* (2015), pp. 45–48. 3

[CZE*23] Chen J., Zhu X., Even M., Basset J., Bénard P., Barla P.: Efficient interpolation of rough line drawings. In *Computer Graphics Forum* (2023), vol. 42, Wiley Online Library, p. e14946. 2

[DBDB78] De Boor C., De Boor C.: *A practical guide to splines*, vol. 27. springer New York, 1978. 4

[Dou25] DoubleL: RPG animations pack FREE. Unity Asset Store, 2025. Accessed: 2025-09-01. URL: https://assetstore.unity.com/packages/3d/animations/rpg-animations-pack-free-288783. 8

[DRVDP15] Dalstein B., Ronfard R., Van De Panne M.: Vector graphics animation with time-varying topology. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 1–12. 2, 3

[DYH*20] Das A., Yang Y., Hospedales T., Xiang T., Song Y.-Z.: Béziersketch: A generative model for scalable vector sketches. In *European conference on computer vision* (2020), Springer, pp. 632–647. 2

[Far12] Farouki R. T.: The bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design 29*, 6 (2012), 379–419. 3

[FBCW23] Fan J. E., Bainbridge W. A., Chamberlain R., Wammes J. D.: Drawing as a versatile cognitive tool. *Nature Reviews Psychology 2*, 9 (2023), 556–568. 2

[FC25] Fang X., Chang M.: Video sketching using multi-domain guidance and implicit encoding: X. fang, m. chang. *The Visual Computer* (2025), 1–12. 2, 3, 6, 7, 10, 11

[Fis22] Fisher R. A.: On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character 222*, 594-604 (1922), 309–368. 4

[FSW22] Frans K., Soros L., Witkowski O.: Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *Advances in Neural Information Processing Systems 35* (2022), 5207–5218. 2

[GBCB16] Goodfellow I., Bengio Y., Courville A., Bengio Y.: *Deep learning*, vol. 1. MIT press Cambridge, 2016. 4

[GRGC15] Guay M., Ronfard R., Gleicher M., Cani M.-P.: Space-time sketching of character animation. *ACM Transactions on Graphics (ToG) 34*, 4 (2015), 1–10. 3

[GSH*19] Gryaditskaya Y., Sypesteyn M., Hoftijzer J. W., Pont S. C., Durand F., Bousseau A.: Opensketch: a richly-annotated dataset of product design sketches. *ACM Trans. Graph. 38*, 6 (2019), 232–1. 2

[GVA*24] Gal R., Vinker Y., Alaluf Y., Bermano A., Cohen-Or D., Shamir A., Chechik G.: Breathing life into sketches using text-to-video priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 4325–4336. 3, 7, 9

[HE17] Ha D., Eck D.: A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477* (2017). 2

[HFW*22] Hinz T., Fisher M., Wang O., Shechtman E., Wermter S.: Charactergan: Few-shot keypoint character animation and reposing. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2022), pp. 1988–1997. 3

[HK70] Hoerl A. E., Kennard R. W.: Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics 12*, 1 (1970), 55–67. 11

[HLC*25] Hong L., Liu Z., Chen W., Tan C., Feng Y., Zhou X., Guo P., Li J., Chen Z., Gao S., et al.: Lvos: A benchmark for large-scale long-term video object segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2025). 2, 8

[HTF*09] Hastie T., Tibshirani R., Friedman J., et al.: The elements of statistical learning, 2009. 4

[Igl25] Iglesias K.: Human basic motions FREE. Unity Asset Store, 2025. Accessed: 2025-09-01. URL: https://assetstore.unity.com/packages/3d/animations/human-basic-motions-free-154271. 8

[IZZE17] Isola P., Zhu J.-Y., Zhou T., Efros A. A.: Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 1125–1134. 2

[JLJ*20] Jeruzalski T., Levin D. I., Jacobson A., Lalonde P., Norouzi M., Tagliasacchi A.: Nilbs: Neural inverse linear blend skinning. *arXiv preprint arXiv:2004.05980* (2020). 3

[JXA23] Jain A., Xie A., Abbeel P.: Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 1911–1920. 2

[Kha25] Khandelwal A.: Flexiclip: Locality-preserving free-form character animation. *arXiv preprint arXiv:2501.08676* (2025). 3

[KMW*24] Karaev N., Makarov I., Wang J., Neverova N., Vedaldi A., Rupprecht C.: Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. *arXiv preprint arXiv:2410.11831* (2024). 6, 11

[KOWD21] Kasten Y., Ofri D., Wang O., Dekel T.: Layered neural atlases for consistent video editing. *ACM Transactions on Graphics (TOG) 40*, 6 (2021), 1–12. 2, 3, 6

[LCKP25] Lee J., Choi C., Kim Y. M., Park J.: Recovering dynamic 3d sketches from videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference* (2025), pp. 12423–12432. 3

[LHX*25] Liang G., Hu J., Xing X., Zhang J., Yu Q.: Multi-object sketch animation with grouping and motion trajectory priors. *arXiv preprint arXiv:2508.15535* (2025). 3, 9

[LLGRK20] Li T.-M., Lukáč M., Gharbi M., Ragan-Kelley J.: Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG) 39*, 6 (2020), 1–15. 2, 4

[LLM*19] Li M., Lin Z., Mech R., Yumer E., Ramanan D.: Photo-sketching: Inferring contour drawings from images. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2019), IEEE, pp. 1403–1412. 2

[LNC*22] Liu Z., Ning J., Cao Y., Wei Y., Zhang Z., Lin S., Hu H.: Video swin transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 3202–3211. 6

[LXF*25] Liu J., Xin Z., Fu Y., Zhao R., Lan B., Li X.: Multi-object sketch animation by scene decomposition and motion planning. *arXiv preprint arXiv:2503.19351* (2025). 3, 9

[LZT*19] Liu L., Zheng Y., Tang D., Yuan Y., Fan C., Zhou K.: Neuroskinning: Automatic skin binding for production characters with deep graph networks. *ACM Transactions on Graphics (ToG) 38*, 4 (2019), 1–12. 3

[OWX*24] Ouyang H., Wang Q., Xiao Y., Bai Q., Zhang J., Zheng K., Zhou X., Chen Q., Shen Y.: Codef: Content deformation fields for temporally consistent video processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 8089–8099. 2, 3, 6

[PJBM22] Poole B., Jain A., Barron J. T., Mildenhall B.: Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988* (2022). 2, 3

[PTPC*17] Pont-Tuset J., Perazzi F., Caelles S., Arbeláez P., Sorkine-Hornung A., Van Gool L.: The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675* (2017). 2

[QZH*20] Qin X., Zhang Z., Huang C., Dehghan M., Zaiane O. R., Jagersand M.: U2-net: Going deeper with nested u-structure for salient object detection. *Pattern recognition 106* (2020), 107404. 5, 9

[RBL*22] Rombach R., Blattmann A., Lorenz D., Esser P., Ommer B.: High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 10684–10695. 2

[RGLM21] Reddy P., Gharbi M., Lukac M., Mitra N. J.: Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 7342–7351. 2

[RKH*21] Radford A., Kim J. W., Hallacy C., Ramesh A., Goh G., Agarwal S., Sastry G., Askell A., Mishkin P., Clark J., et al.: Learning transferable visual models from natural language supervision. In *International conference on machine learning* (2021), PmLR, pp. 8748–8763. 2, 4, 7

[RS24] Rai G., Sharma O.: Enhancing sketch animation: Text-to-video diffusion models with temporal consistency and rigidity constraints. *arXiv preprint arXiv:2411.19381* (2024). 3

[SBF*18] Su Q., Bai X., Fu H., Tai C.-L., Wang J.: Live sketch: Video-driven dynamic deformation of static drawings. In *Proceedings of the 2018 chi conference on human factors in computing systems* (2018), pp. 1–12. 3

[SCBS13] Santosa S., Chevalier F., Balakrishnan R., Singh K.: Direct space-time trajectory control for visual media editing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2013), pp. 1149–1158. 3

[See04] Seeger M.: Gaussian processes for machine learning. *International journal of neural systems 14*, 02 (2004), 69–106. 4

[SKC01] Su M.-S., Ko M.-T., Cheng K.-Y.: Control of feature-point-driven facial animation using a hypothetical face. In *Computer Graphics Forum* (2001), vol. 20, Wiley Online Library, pp. 179–189. 3

[SLT*19] Siarohin A., Lathuilière S., Tulyakov S., Ricci E., Sebe N.: First order motion model for image animation. *Advances in neural information processing systems 32* (2019). 3

[SME20] Song J., Meng C., Ermon S.: Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502* (2020). 3

[Sto48] Stone M. H.: The generalized weierstrass approximation theorem. *Mathematics Magazine 21*, 5 (1948), 237–254. 3, 11

[SZL*23] Smith H. J., Zheng Q., Li Y., Jain S., Hodgins J. K.: A method for animating children's drawings of the human figure. *ACM Transactions on Graphics 42*, 3 (2023), 1–15. 3

[TG22] Tian X., Günther T.: A survey of smooth vector graphics: Recent advances in repr esentation, creation, rasterization, and image vectorization. *IEEE Transactions on Visualization and Computer Graphics 30*, 3 (2022), 1652–1671. 2

[TWW*24] Tanveer M., Wang Y., Wang R., Zhao N., Mahdavi-Amiri A., Zhang H.: Anamodiff: 2d analogical motion diffusion via disentangled denoising. *arXiv preprint arXiv:2402.03549* (2024). 3

[VACOS23] Vinker Y., Alaluf Y., Cohen-Or D., Shamir A.: Clipascene: Scene sketching with different types and levels of abstraction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 4146–4156. 2, 10

[VPB*22] Vinker Y., Pajouheshgar E., Bo J. Y., Bachmann R. C., Bermano A. H., Cohen-Or D., Zamir A., Shamir A.: Clipasso: Semantically-aware object sketching. *ACM Transactions on Graphics (TOG) 41*, 4 (2022), 1–11. 2, 4, 5, 6, 7, 9, 11

[WKO12] Winnemöller H., Kyprianidis J. E., Olsen S. C.: Xdog: An extended difference-of-gaussians compendium including advanced image stylization. *Computers & Graphics 36*, 6 (2012), 740–753. 4

[WSML25] Wu R., Su W., Ma K., Liao J.: Aniclipart: Clipart animation with text-to-video priors. *International Journal of Computer Vision 133*, 6 (2025), 3149–3165. 3

[WYC*23] Wang J., Yuan H., Chen D., Zhang Y., Wang X., Zhang S.: Modelscope text-to-video technical report. *arXiv preprint arXiv:2308.06571* (2023). 3

[XT15] Xie S., Tu Z.: Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1395–1403. 6, 7

[XWZ*23] Xing X., Wang C., Zhou H., Zhang J., Yu Q., Xu D.: Diffsketcher: Text guided vector sketch synthesis through latent diffusion models. *Advances in Neural Information Processing Systems 36* (2023), 15869–15889. 2

[XZW*24] Xing X., Zhou H., Wang C., Zhang J., Xu D., Yu Q.: Svgdreamer: Text guided svg generation with diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 4546–4555. 2

[Yin25] YingliuZhizhu1: Brothers, i leave it to you, do as you see fit #yingliu zhizhu challenge. Bilibili, 2025. [Video; in Chinese]. URL: https://www.bilibili.com/video/BV1pmKEzRE5H. 8, 12

[YTZ*24] Yang Z., Teng J., Zheng W., Ding M., Huang S., Xu J., Yang Y., Hong W., Zhang X., Feng G., et al.: Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072* (2024). 9

[YYF*25] Ye X., Yao J.-H., Feng J., Mei S., Lan X., Chen S.: Vidanimator: User-guided stylized 3d character animation from human videos. *arXiv preprint arXiv:2508.01878* (2025). 3

[YZAS21] Yan W., Zhang Y., Abbeel P., Srinivas A.: Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157* (2021). 6

[ZCXP24] Zheng Y., Cun X., Xia M., Pun C.-M.: Sketch video synthesis. In *Computer Graphics Forum* (2024), vol. 43, Wiley Online Library, p. e15044. 2, 3, 5, 7, 10

[ZCZ*09] Zhang S.-H., Chen T., Zhang Y.-F., Hu S.-M., Martin R. R.: Vectorizing cartoon animations. *IEEE Transactions on Visualization and Computer Graphics 15*, 4 (2009), 618–629. 2

[ZIE*18] Zhang R., Isola P., Efros A. A., Shechtman E., Wang O.: The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 586–595. 7

[ZZLZ15] Zhang K., Zhang L., Lam K.-M., Zhang D.: A level set approach to image segmentation with intensity inhomogeneity. *IEEE transactions on cybernetics 46*, 2 (2015), 546–557. 4