SkEncode: Sketch Encoder for Sketch-to-video

Joonki Min Jonghyeon An Chanwoo Kim

Seoul National University

{minc69, anjonghyeon11, chanwoo.kim}@snu.ac.kr

Abstract

Sketches offer a simple way to understand images in a general sense. Turning sketches into videos requires a deeper understanding beyond basic image comprehension. Traditional methods for handling images have limitations when applied to sketches. Recently, there has been a surge in research focused on developing models and techniques tailored specifically to the sketch domain to overcome these limitations. Converting sketches into videos requires a better understanding of the video medium within this research area. Although various methods exist for sketch-to-video conversion, text-based video generation has gained popularity due to its ease of use. In this study, we explore different ways to improve the process of turning static sketches into videos using text prompts. Existing methods face several challenges: maintaining the unique identity of sketches, accurately representing multiple objects, and adapting to slight variations in sketch characteristics. To address these challenges, we propose applying various techniques used in video generation models to the task of sketch-to-video generation. Our aim is to enhance the quality of sketch videos, maintain sketch identity, and create longer sketch-preserving videos that better reflect the textual content. Through these improvements, we seek to overcome the existing challenges and produce more coherent and representative sketch videos.

1. Introduction

Abstract sketches serve as a simplified representation of images, portraying static landscapes or objects, as well as dynamically moving subjects. In this paper, we aim to effectively transform such sketches into videos, thereby enhancing the process of creating moving videos.

Sketches come in various forms, with the handling method varying according to their complexity. To establish a more precise problem setting, this paper focuses on handling sketches composed of vector representations. Similar to sketch generation models, we will represent these sketches' vector representations through cubic Bèzier curves [3, 19, 20]. This vector-based representation is more concise compared to the traditional pixel-based method and offers relative freedom concerning resolution and scaling changes. Additionally, cubic Bèzier curves, represented by four control points, facilitate expressing line movements or changes corresponding to various actions relatively easily.

This paper aims to create appropriate videos depicting movements when provided with sketches in vector representation along with text prompts describing actions. To achieve this, we will utilize various pretrained text-to-video diffusion models [17]. To effectively use these pretrained models, an optimization process is required to input sketches appropriately into the text-to-video model and optimize them into videos tailored to the text prompts.

Existing sketch-to-video models [3, 25] utilizing similar approaches exhibit various shortcomings. They only smoothly generate videos for specific sketch formats and fail to produce distinct results for multiple objects. Moreover, certain prompts or sketches may not adequately maintain form or identity.

To address these issues, we check the MLPs used in existing models to apply text-to-video models to sketches more effectively. These MLPs convert sketches into embedding forms, adjust movements, and calculate SDS loss based on the results, within text-to-video model. Thus, if the layer connecting sketches and text-to-video is more effective in conveying sketch information and adjusting movements, more complete videos can be obtained. We refer to these layers as "Sketch Encoders" in this paper and will focus on designing them more effectively.

To design a more effective sketch encoder, we start by identifying the shortcomings of existing models and partially improving them. Firstly, we will enhance positional encoding to make it more suitable for the sketch domain during the process of converting sketches into embeddings. As sketches have simpler and more regular forms, grouping embeddings based on frame numbers or bèzier curve positions and providing positional encoding according to these rules will more effectively reflect sketch characteristics and create high-quality videos. Furthermore, to generate videos

for multiple objects more effectively, we will apply a clustering layer to distinguish objects in sketches and design them to separately learn movements for each object. Lastly, the final sketch encoder will be validated and improved through various types of sketch datasets to create a more generalized structure of the sketch-to-video model.

The final results of the sketch encoder and the sketch-to-video model developed through this process will be compared with recent sketch-to-video papers[3, 25]. If we need more, text-to-video papers[1, 21] can also make examples by using various sketch datasets and text prompts.

2. Related Work

2.1. Sketch

Research on sketches, ranging from relatively detailed sketches depicting landscapes or objects to those composed of very few lines, has been conducted for some time as a category of images. Particularly, methods utilizing sketches in image synthesis processes[2] or in image generation[8, 12, 23] have used sketches as auxiliary elements in generating more concrete images. Recent attempts also involve utilizing sketches in image generation processes using diffusion[9, 22]. Additionally, there have been attempts to use sketches composed of a few lines to provide rough shapes as tools for image editing[5, 28]. In this study, sketches will be used as baseline images for generating videos.

Sketches are used for various purposes depending on their intended use, and they come in various forms. Sketches can be divided based on differences in form, such as pixel representation and vector representation, as well as details or drawing methods. In this study, we will design a model that can generalize well to any form of sketch without limitations imposed by the various purposes and forms of sketches, ensuring effective sketch-to-video generation.

2.2. Video Generation

With the emergence of various models for image generation, video generation models initially extended image-based GAN methods to generate videos[7, 10]. While research has progressed on generating videos using diffusion[4], these studies often focus on generating videos on specific topics, resulting in relatively limited diversity in the generated content.

More recently, research has been conducted on models for generating text-based videos, building on previous studies on text-to-image and leveraging learned data on images and videos to perform text-to-video generation[16]. Various diffusion-based methods for video generation have also been studied[14, 15, 24, 27]. These text-to-video models are trained using images as conditions[21], captioned images as conditions[13], or both text and images for content preservation[1].

Furthermore, in the realm of video generation, research utilizing masked diffusion for interactive video generation[6] according to user intentions shows that user intentions can be increasingly reflected in various ways during the video generation process. In this study, we will effectively apply such video generation models to the sketch domain, designing and improving encoders that can be applied to these video generation models.

2.3. Sketch-to-video Generation

Creating moving videos through sketches is a relatively recent area of research. Attempts to apply sketches to the video domain have used methods similar to those that employed sketches in image editing or as auxiliary elements in video editing[11]. There have also been studies on synthesizing new videos by combining given videos with sketches as a basis[29]. In the field of sketch animation, existing sketch animations have been completed[26] or motion sequences[18] of videos have been provided alongside sketches to animate them.

More recently, research has emerged that utilizes the text-to-video prior of video generation models to apply it to the sketch domain, that make very simple sketches composed of several strokes move[3]. Similarly, research has been conducted to utilize sketches as elements expressing motion, extracting their characteristics to create smooth movements in clipart animation[25]. However, these studies often restrict themselves to specific forms of sketches, struggle to handle multiple objects well, fail to maintain the form of sketches, or do not smoothly express movements in longer videos. In this study, we will address the limitations of existing sketch-to-video models and propose a model that produces higher-quality videos.

3. Method

Our method is based on the video generation process of existing sketch-to-video models. Users provide an SVG format sketch and a text prompt describing specific actions. Through this, we will generate a video in the SVG format corresponding to the described action in the text prompt.

We follow the problem setting of existing sketch-to-video models [3]. The input vector image is depicted as a series of strokes, each stroke consisting of multiple control points. Each control point is represented by its coordinates $p = (x, y) \in \mathbb{R}^2$. The set of control points in a single frame is denoted by $P = \{p_1, \ldots, p_N\} \in \mathbb{R}^{N \times 2}$, where N is the total number of points in the input sketch. This number stays constant across all generated frames. A video with K frames is defined as a sequence of K such sets of control points, denoted by $Z = \{P_k\}_{k=1}^{K} \in \mathbb{R}^{K \cdot N \times 2}$. Let P_{init} represent the set of points in the initial sketch. We replicate P_{init} K times to create the initial set of frames Z_{init}. The goal is to convert

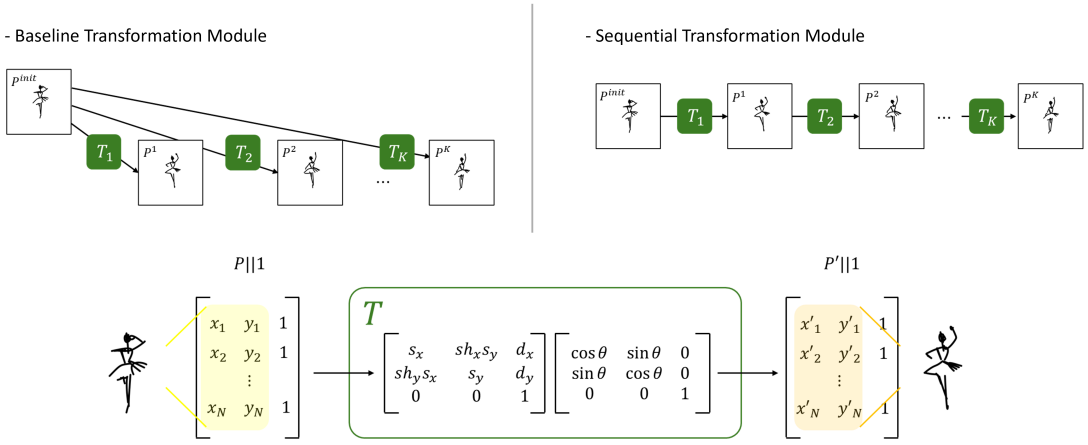

Figure 1. Transformation Module. The input to the transformation module is the set of control points P, and its output is the set of displaced control points, where the displacements correspond to some portion of prompted actions. This module utilizes a learned matrix to perform scaling, shearing, rotation, and translation on the coordinates of the control points. In the baseline transformation module, transformations are applied simultaneously to P^{init} for each frame. Conversely, in our sequential transformation module, each transformation is applied to P^{k-1} to produce P^k, ensuring a more consistent and coherent motion sequence.

this static sequence into an animated sequence that follows the motion described in the text prompt. We frame this task as learning a set of 2D displacements $\Delta Z = \{\Delta p_i^j\}_{i \in N, j \in K}$, which indicate the displacement of each point p_i^j for each frame j.

The output video $Z = Z_{init} + \Delta Z$, should maintain the identity of the sketch while smoothly depicting the actions described in the text prompt. Additionally, these videos should (1) handle multiple objects well, (2) maintain shapes better, and (3) generally perform well with various sketch formats compared to existing video models.

3.1. Identity-preserving Encoder

To improve existing methodologies, we first propose an identity-preserving positional encoder. Given that the output video is a sequence of sketch frames, each frame must consistently depict the same object as the input sketch. Therefore, it is crucial for each stroke in every frame to consistently represent the same part of the object throughout the sequence. For example, if a stroke delineates an arm in the input sketch, it should continue to represent the arm across all subsequent frames, even if its shape is altered to animate the arm's movement. Any alteration causing the stroke to depict a different part of the object, such as a leg, would lead to deformation and significantly compromise identity preservation.

A sketch comprises multiple strokes, each defined by a set of control points. To maintain stroke-specific information across frames, we explicitly encode positional information using sinusoidal positional embeddings. Previous methods [3] employed a positional embedding $E_{base} \in \mathbb{R}^{K \cdot N \times d}$, where d represents the embedding dimension. This approach injects different positional embeddings for the same stroke in different frames, risking loss of stroke consistency across frames. To address this, we propose (1) positional encoding per control point and (2) positional encoding per stroke. The positional encoding per control point is represented as $E_p \in \mathbb{R}^{N \times d}$, and the positional encoding per stroke as $E_s \in \mathbb{R}^{N/N_s \times d}$, where N_s is the number of control points per stroke. The former ensures consistent encoding for the same control points across frames, while the latter provides consistent encoding for the same stroke throughout all frames.

3.2. Sequential Transformation Module

To generate an animation that corresponds to the text prompt, we utilize a "neural displacement field", as [3] does. This is a compact network M that takes the initial point set Z_{init} as input and predicts their displacements, denoted as $M(Z_{init}) = \Delta Z$. The network M comprises two parts: M_l and M_g. M_l generates an offset ΔZ_l for each control point in Z_{init}. On the other hand, M_g predicts global transformation matrices $\{T_{base}^k\}_{k=1}^K$. They are applied to generate global transformations for each frame starting from Z_{init}:

$$\Delta p_{i,global}^k = T_{base}^k \odot p_i^{init} - p_i^{init}. \quad (1)$$

To ensure realistic motion and stability, the transformation matrices are constrained to induce only scaling, shearing, rotation, and translation. Moreover, these transformations are applied simultaneously at all points in a frame rather than on individual points, enhancing overall animation stability. However, confining transformations to originate solely from the initial frame poses challenges for dynamic movement, re-

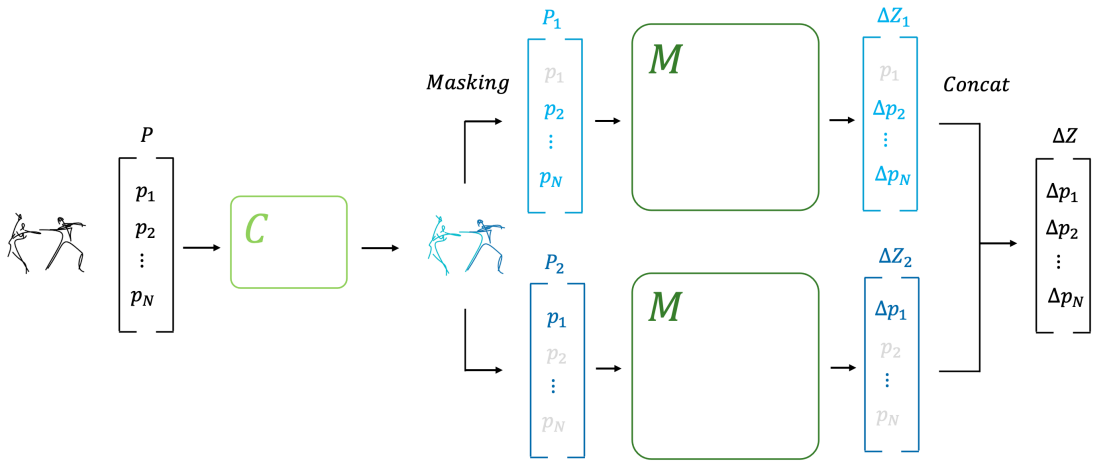

Figure 2. When control point inputs are fed into clustering module C, it performs clustering, retains only the control points belonging to each cluster, and masks the rest, resulting in control points for each cluster P_1, P_2. Each control point then passes through Model M, generating movements in different ways $\Delta Z_1, \Delta Z_2$. The learned results are concatenated to produce the overall movement ΔZ.

stricting motions from deviating significantly from the initial points. Hence, we propose a novel Sequential Transformation Module to address this limitation

$$\Delta p^k_{i,global} = T^k_{seq} \odot p^{k-1}_i - p^{k-1}_i, \tag{2}$$

where $p^0_i = p^{init}_i$. By originating each transformation from the previous frame rather than the initial frame, the movements become more consecutive. This sequential approach allows for a more coherent transition between frames. Furthermore, a series of incremental transformations enhances the ability to depict larger overall motions. The overall difference between two transformation modules and the details are described in Fig. 1.

3.3. Clustering for Multiple Objects

One of the key challenges in the process of generating videos from sketches is handling multiple objects. To address this, we use masking techniques and various clustering algorithms. While multiple objects have been addressed in traditional image-to-video models, they still pose challenges in sketch data. We aim to modularize existing ideas to better suit simpler sketch datasets by (1) using the center of the four control points of the Bezier curve as the clustering criterion to distinguish individual objects, (2) creating a module that segments each object by applying various clustering algorithms, and (3) generating independent movements by training the encoder with masked individual objects.(See Fig. 2).

We considered which of the four points of a Bézier curve best represents the corresponding stroke during the clustering process. Among the four points, the starting point and the end point are on the stroke. However, if we determine the middle point by considering only these two points, we cannot take into account the curve's direction of bending. Therefore, we performed clustering based on the positions calculated by considering the bending direction of the cubic Bézier curve, specifically the midpoint of the two control points along with the starting and end points.

To find the best clustering algorithm, we make a experiment with several commonly used clustering algorithms, K-means, DBSCAN, and meanshift. We then evaluate the results of these algorithms on simple sketches to determine which algorithm would be the most suitable. Based on this evaluation, we selected the appropriate algorithm to be used in our sketch clustering module.

4. Result

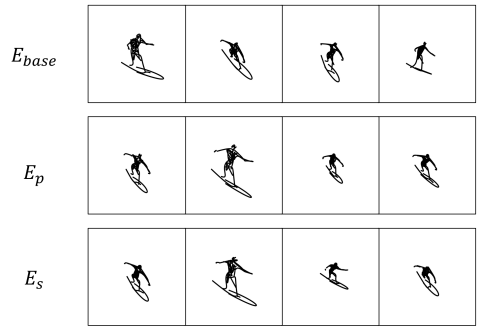

Figure 3. Qualitative results of three types of positional encoders: E_{base}, E_p, and E_s.

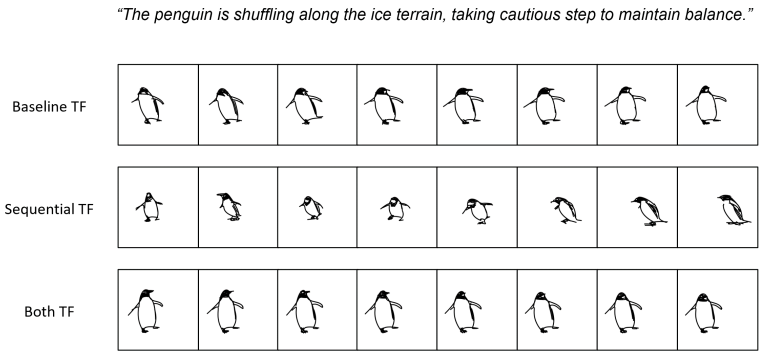

Figure 4. Qualitative results of three transformation modules: Baseline, Sequential, and Both.

4.1. Identitiy-preserving Encoder

Our positional encoder is explicitly designed to distinguish each stroke across all frames, ensuring precise tracking of every stroke throughout the sequence. To assess the effectiveness of our proposed positional encoder in maintaining the integrity of the input sketch, even in challenging scenarios, we conduct experiments with sketches featuring multiple objects. In these cases, each stroke corresponding to an object must consistently represent the same object across all frames. Any stroke representing a different part of another object would significantly compromise the identity of the input sketch. We select a sketch of a man on a surfboard, which includes two objects—a man and a board. Using the prompt "A surfer riding and maneuvering on waves on a surfboard," we compare the baseline positional encoding methods: (1) E_{base}, applied to every control point on every frame; (2) E_p, applied to control points within a single frame; and (3) E_s, applied to strokes for the same sketch and prompt. The results are nearly identical, irrespective of the type of positional encoder employed. (See Fig. 3)

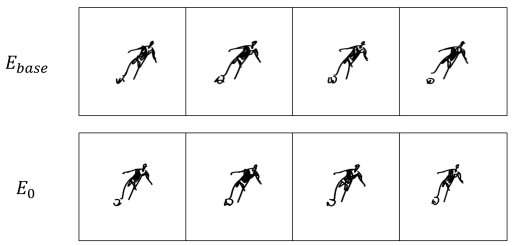

Figure 5. Qualitative results of with/without positional encoders: E_{base} and E_0.

We investigate the reasons behind the suboptimal performance of the positional encoder by examining the initial input layer of the model in detail. The model first duplicates the input sketch into K frames, creating an embedding vector that is a concatenation of the repeated input frame's embedding vectors. This process inherently performs a role similar to the per-point positional encoder, E_p. Consequently, the positional encoder may not be essential. To validate this hypothesis, we intentionally remove the encoder by setting all its elements to zero, $E_0 = 0$, and compare the results with the original encoder, E_{base}. The results appear nearly identical (See Fig.5).

4.2. Sequential Transformation Module

The sequential transformation module is designed to facilitate smooth animations by ensuring seamless transitions from frame to frame. This approach simplifies the creation of consecutive frames compared to simultaneously creating frames starting from an initial frame. To demonstrate its effectiveness, we applied the same prompt to three different transformation modules: the baseline, our sequential transformation module, and a mixture of both (see Fig. 4). The results indicate that our method produces the most dynamic and smooth 'shuffling along' action prompt. When combining both transformations, the baseline tends to disrupt the sequential transformation's smoothness. Hence, it is more reasonable to only use sequential transformation module. For further analysis on its capability, we design two more experiments considering intensity and directionality of movement.

Intensity of Movement To evaluate the capability of our module to represent both dynamic motions with strong transitions and calm actions with minor motions, we applied two distinct prompts describing dynamic and calm actions respectively to the same sketch. As illustrated in Fig. 6, our module successfully generates both types of actions accurately. In contrast, the baseline module produces only calm actions, even when the prompt specifies dynamic motions.

Directionality of Movement Since a sketch is a 2D representation of an object, much of its 3D information is inherently lost. However, animating a sketch requires transformations that consider 3D axes. To evaluate the capability of expressing movements based on 3D axes, we incorpo-

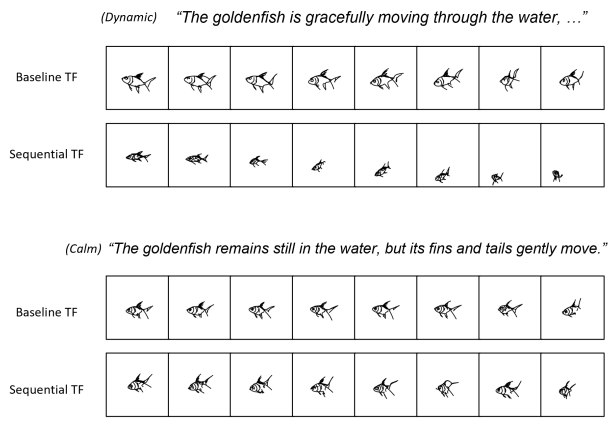

Figure 6. Qualitative results of dynamic and calm prompts.

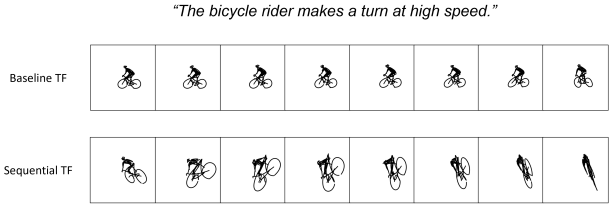

Figure 7. Qualitative results of directive prompts.

rated motions with directionality. As illustrated in Fig. 7, our module successfully generates movements that include 3D transitions. In contrast, the baseline module produces only subtle movements, failing to fully capture 3D transitions.

4.3. Clutsering for Multiple Objects

In this study, we addressed the task of generating sketch animations based on text prompts. Specifically, we investigated a method to handle multiple objects within a single sketch by clustering the objects, training them separately, and generating animations that exhibit independent movements. To achieve this, we experimented with three clustering algorithms: K-means, DBSCAN, and Meanshift.

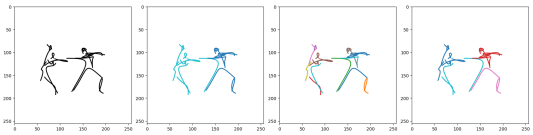

Figure 8. From left to right: original sketch, segmentation results with K-means, DBSCAN, and Meanshift.

Although the K-means algorithm requires specifying the number of objects in a multiple-object sketch, unlike the other two algorithms, it demonstrated relatively superior performance among the three.(Fig. 8) We combined the segmentation results from the K-means algorithm with an improved encoder to generate the sketch videos.

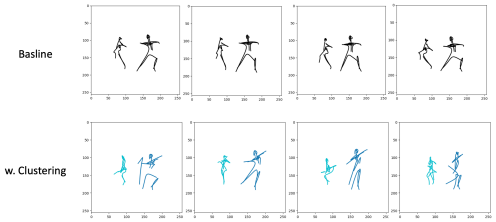

Figure 9. Animation of a sketch with multi-object segmentation using the K-means algorithm. Baseline model makes movement like both people are in single-object, while clustering module makes different movement for each person.

The K-means algorithm is an ideal choice for real-time applications due to its shorter training time compared to other clustering algorithms. Additionally, it has shown to provide the best performance in partitioning multiple objects within a sketch and generating high-quality animations with independent movements for each object. As shown in the figure, unlike the previous model where two objects moved as if they were a single object, the results using clustering module demonstrated that the movements of the two people appeared independently.(Fig. 9)

5. Conclusion and Future works

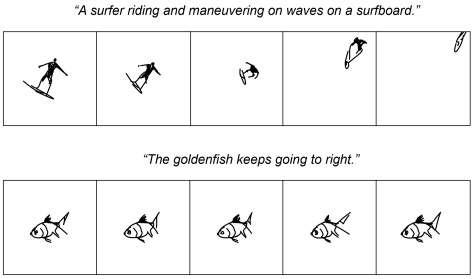

Figure 10. Qualitative results of suboptimal cases (1) & (2)

Through the experimental results, we observe that the existing positional encoding does not play a significant role, indicating that enhancing positional encoding requires a more sophisticated design. Conversely, the sequential transformation module demonstrates its ability to generate smoother and more plausible movements compared to the baseline module. However, it occasionally amplifies movements excessively, causing the object in the sketch to entirely disappear from the frames. This is considered a trade-off for achieving more dynamic movements.(See Fig.10-(1)) To mitigate this issue, we temporarily adjust hyperparameters

related to the magnitudes of transition and scaling. Nonetheless, further analysis is needed to identify fundamental solutions. Additionally, some explicit action prompts fail to translate into the intended output movements. We hypothesize that this is due to the difficulty of precisely matching the initial sketch with the prompt, which is more complex than merely adding some motions. (See Fig.10-(2))

Through experiments with the clustering module, we confirmed that when a sketch composed of multiple objects is input, clustering can distinguish the objects within the sketch. By separately training each object through masking, we were able to successfully generate distinct movements for each. We found that among several algorithms, k-Means was the most suitable for sketches with a relatively small number of strokes. Since each object is trained and generated separately, it provides a flexible and scalable model that can be applied to various sketch datasets in the future, allowing for the addition of new objects or individual adjustments to existing object attributes.

Although our research demonstrated the potential to create sketch videos composed of multiple objects through clustering, several issues remain to be addressed. First, the clustering module uses k-means, requiring us to manually specify the number of clusters. Most sketches depicted a single object, and sketches containing more than two objects were rare. However, in a clustering module handling multiple objects, the necessity for users to specify the number of objects poses a problem. Therefore, it is necessary to find sketch-oriented solutions at the model level to address this issue. Second, as seen in the example where a stroke was misclassified, improvements to the clustering algorithm are needed. Although we conducted additional experiments using hierarchical clustering, as shown in the Fig.11, merely changing the algorithm without altering the number of specified clusters did not lead to improvements. Thus, we need to tune the clustering algorithm to be more suitable for the sketch domain, for example, by redefining the distance between the centroids of the strokes. Lastly, in the training process, only one text prompt is input, which poses a problem when each clustered sketch needs to be trained with a single text prompt like the existing model. In the example scene with two people dancing, where both are performing the same action, this issue does not arise. However, if a sketch contains two objects, such as a ball and a person or a bird and a flower, requiring different movements, training each object cluster with the same text prompt will cause problems. To solve this issue, we could consider allowing different text inputs to describe the movements of each object when multiple objects are present.

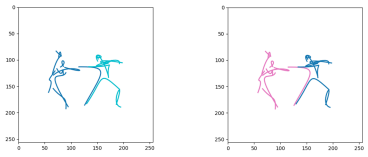

Figure 11. Clustering result. left: K-means, right: Hierarchical clustering. Both clustering algorithms show same results.

References

[1] Haoxin Chen, Menghan Xia, Yingqing He, Yong Zhang, Xiaodong Cun, Shaoshu Yang, Jinbo Xing, Yaofang Liu, Qifeng Chen, Xintao Wang, Chao Weng, and Ying Shan. Videocrafter1: Open diffusion models for high-quality video generation, 2023. 2

[2] Wengling Chen and James Hays. Sketchygan: Towards diverse and realistic sketch to image synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[3] Rinon Gal, Yael Vinker, Yuval Alaluf, Amit H. Bermano, Daniel Cohen-Or, Ariel Shamir, and Gal Chechik. Breathing life into sketches using text-to-video priors. 2023. 1, 2, 3

[4] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models, 2022. 2

[5] Wei Huang Yibing Song Xintong Han Jing Liao Bing Jiang Hongyu Liu, Ziyu Wan and Wei Liu. Deflocnet: Deep image editing via flexible low level controls. In *CVPR*, 2021. 2

[6] Yash Jain, Anshul Nasery, Vibhav Vineet, and Harkirat Behl. Peekaboo: Interactive video generation via masked-diffusion. *arXiv preprint arXiv:2312.07509*, 2023. 2

[7] Doyeon Kim, Donggyu Joo, and Junmo Kim. Tivgan: Text to image to video generation with step-by-step evolutionary generator. *IEEE Access*, 8:153113–153122, 2020. 2

[8] Subhadeep Koley, Ayan Kumar Bhunia, Aneeshan Sain, Pinaki Nath Chowdhury, Tao Xiang, and Yi-Zhe Song. Picture that sketch: Photorealistic image generation from abstract sketches, 2023. 2

[9] Subhadeep Koley, Ayan Kumar Bhunia, Deeptanshu Sekhri, Aneeshan Sain, Pinaki Nath Chowdhury, Tao Xiang, and Yi-Zhe Song. It's all about your sketch: Democratising sketch control in diffusion models, 2024. 2

[10] Yitong Li, Martin Min, Dinghan Shen, David Carlson, and Lawrence Carin. Video generation from text. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018. 2

[11] Feng-Lin Liu, Shu-Yu Chen, Yu-Kun Lai, Chunpeng Li, Yue-Ren Jiang, Hongbo Fu, and Lin Gao. DeepFaceVideoEditing: Sketch-based deep editing of face videos. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2022)*, 41(4): 167:1–167:16, 2022. 2

[12] Yongyi Lu, Shangzhe Wu, Yu-Wing Tai, and Chi-Keung Tang. Image generation from sketch constraint using contextual gan. In *European Conference on Computer Vision*, 2017. 2

[13] Zhengxiong Luo, Dayou Chen, Yingya Zhang, Yan Huang, Liang Wang, Yujun Shen, Deli Zhao, Jingren Zhou, and Tieniu Tan. Videofusion: Decomposed diffusion models for high-quality video generation. *CoRR*, abs/2303.08320, 2023. 2

[14] Zhengxiong Luo, Dayou Chen, Yingya Zhang, Yan Huang, Liangsheng Wang, Yujun Shen, Deli Zhao, Jinren Zhou, and Tien-Ping Tan. Videofusion: Decomposed diffusion models for high-quality video generation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10209–10218, 2023. 2

[15] Haomiao Ni, Changhao Shi, Kai Li, Sharon X Huang, and Martin Renqiang Min. Conditional image-to-video generation with latent flow diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18444–18455, 2023. 2

[16] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data, 2022. 2

[17] Uriel Singer, Shelly Sheynin, Adam Polyak, Oron Ashual, Iurii Makarov, Filippos Kokkinos, Naman Goyal, Andrea Vedaldi, Devi Parikh, Justin Johnson, and Yaniv Taigman. Text-to-4d dynamic scene generation, 2023. 1

[18] Qingkun Su, Xue Bai, Hongbo Fu, Chiew-Lan Tai, and Jue Wang. Live sketch: Video-driven dynamic deformation of static drawings. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 1–12, New York, NY, USA, 2018. Association for Computing Machinery. 2

[19] Yael Vinker, Ehsan Pajouheshgar, Jessica Y. Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. Clipasso: Semantically-aware object sketching. *ACM Trans. Graph.*, 41(4), 2022. 1

[20] Yael Vinker, Yuval Alaluf, Daniel Cohen-Or, and Ariel Shamir. Clipascene: Scene sketching with different types and levels of abstraction, 2023. 1

[21] Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. Modelscope text-to-video technical report, 2023. 2

[22] Qiang Wang, Di Kong, Fengyin Lin, and Yonggang Qi. Diffsketching: Sketch control image synthesis with diffusion models. In *33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022*. BMVA Press, 2022. 2

[23] Sheng-Yu Wang, David Bau, and Jun-Yan Zhu. Sketch your own gan. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 2

[24] Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Stan Weixian Lei, Yuchao Gu, Yufei Shi, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7623–7633, 2023. 2

[25] Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. Aniclipart: Clipart animation with text-to-video priors, 2024. 1, 2

[26] Jun Xing, Li-Yi Wei, Takaaki Shiratori, and Koji Yatani. Autocomplete hand-drawn animations. *ACM Trans. Graph.*, 34 (6), 2015. 2

[27] Sihyun Yu, Kihyuk Sohn, Subin Kim, and Jinwoo Shin. Video probabilistic diffusion models in projected latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 2

[28] Yu Zeng, Zhe Lin, and Vishal M. Patel. Sketchedit: Mask-free local image manipulation with partial sketches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 2

[29] Yudian Zheng, Xiaodong Cun, Menghan Xia, and Chi-Man Pun. Sketch video synthesis. 2023. 2