Vol.2023-CG-191 No.8
2023/9/16

Sketch-Based Velocity Field Design using Latent Diffusion Model

Hengyuan Chang[1] Yichen Peng[1] Syuhei Sato[2] Haoran Xie[1]

Abstract: The fusion of velocity field creation and deep learning to enhance the effectiveness of flow simulation is both widely pursued and intricate. Existing methods are mainly based on generative adversarial networks(GANs) to generate the target velocity field with the sketch as input. However, the training of GANs, and balancing the generator and discriminator is an unstable process, which causes generated unstable samples. In this research, we propose an interactive 2D velocity field design generation framework. In our framework, the streamline sketch is used as a constraint condition, maps into the latent space with an encoder, constrains the denoising process and reconstructs the 2D velocity field by decoder. The results show that our framework generates velocity fields corresponding to the shape of given sketches, and also allows users to reconstruct velocity fields from hand-drawn sketches. We compared our results with the GAN-based model. The evaluation shows our framework is more robust than the GAN-based method.

Keywords: Velocity field generation, latent diffusion model, sketch-guided, auto-encoder

1. Introduction

Flow simulation is a foundational topic within computer graphics and holds significant importance in animation content creation. Presently, the execution of physical fluid simulation heavily relies on commercial applications or tools such as Blender, Houdini, ANSYS, as well as game engines like Unreal. However, the operations and processes of these tools are complex, this necessitates users to possess relevant knowledge and a certain level of design proficiency. This is unquestionably challenging for users without a professional background to utilize these tools to achieve their desired effects. For a majority of people, sketching offers the most convenient means of representation. By applying sketches as guidance to formulate the expected velocity field, the realization of fluid simulation presents a feasible approach to simplify the fluid simulation process.

Several studies have focused on the generation of velocity fields through sketches in recent years. Zhu et al.[1] proposed a numerical-based sketch system to illustrate different fluid systems through sketch editing. Xing et al. [2] proposed an approach using energy brushes to drive flow particles, resulting in the generation of corresponding velocity fields. These numerical-solver-based approaches generate satisfactory experimental results, but their complexity necessitates the utilization of multiple components and tools to achieve the expected simulations. In recent years, a growing trend involves integrating deep learning architectures such as GAN with physical simulation. Hu et al.[3] presented a sketch system grounded in conditional generative adversarial network (cGAN) to generate 2D velocity fields for 2D flow design while the sketch comprises merely 3 elements. Yan et al. [4] discussed a virtual reality (VR) sketch system utilizing cGAN to generate 3D velocity fields for 3D liquid splash generation, with the sketch encompassing a single type of strokes. Notably, these approaches collected hand-drawn sketches as their training data. Although the GAN-based method is capable of directly reconstructing expected velocity fields from input sketches, their training of GAN can be inherently unstable due to the balance of the adversary between generator and discriminator. More recently, the diffusion model (DM) [5] has achieved significant advancements in generation tasks, surpassing the capabilities of GAN architecture. The DM structure learns the reverse diffusion process to recover an image mixed with Gaussian noise. Unlike the generator and discriminator, DM sidesteps the adversarial process, which contributes to the heightened robustness compared to the GAN.

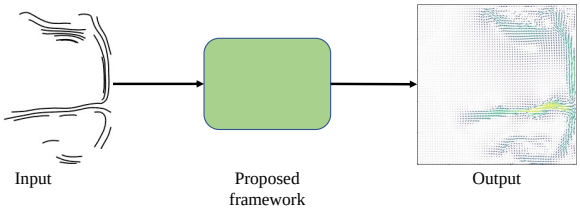

Fig. 1 This work can generate the velocity field from sketch input using diffusion model.

In this paper, we propose a sketch-based 2D velocity field design utilizing the latent diffusion model (LDM) [6]. The fundamental overview of this research is shown in Figure 1. The sketch data is configured as the input constrain condition, the 2D velocity field is generated passthrough the proposed framework. The details of framework is introduced in section 2. The LDM effectively condenses intricate data into a more simplified feature map or feature vector. This simplifies the analysis process and reduces

[1] Japan Advanced Institute of Science and Technology
[2] Hosei University

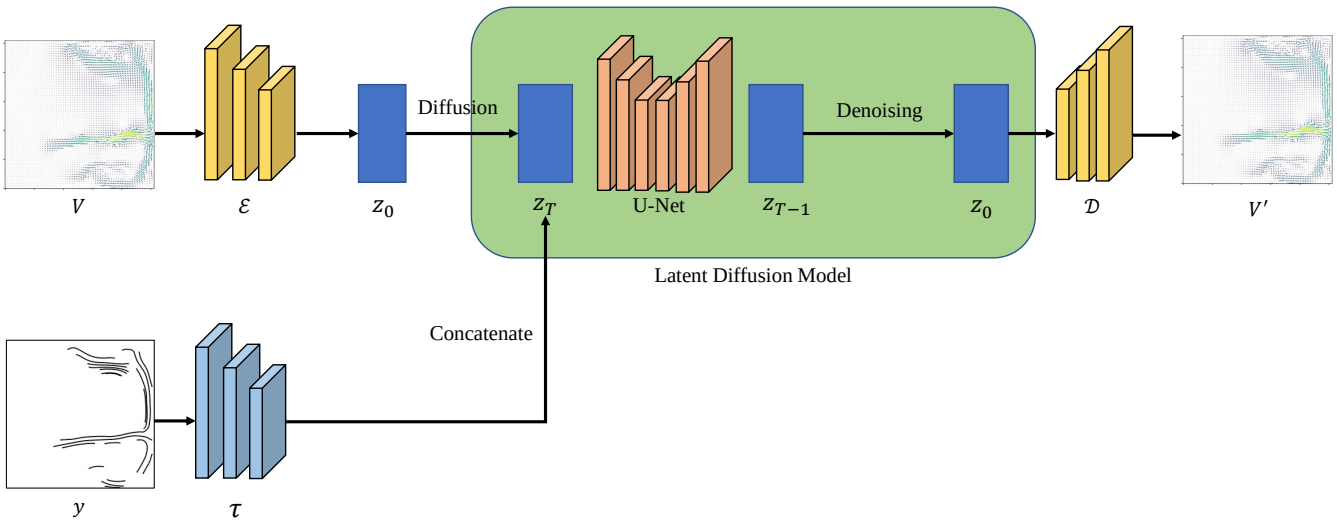

Fig. 2 The framework of the proposed sketch-based velocity field design.

the computational demands while keeping the main features of the original input. Notably, LDM is remarkably adaptable to constrain target generation using diverse input forms as conditions. We create a 2D velocity field and sketch dataset as there exists no open-source dataset tailored to our experiment. Additionally, We formulate two auto-encoders aimed at compressing both velocity fields and sketches into corresponding feature maps and reconstructing data from given feature maps. The contributions of this research are shown as follows: 1. We propose an LDM-based velocity field generation framework, which is the first attempt to apply LDM for velocity field generation design by inputting sketch strokes. 2. We conduct a comparison between the proposed framework and the cGAN structure. Furthermore, we validate the stability of the proposed framework within the context of 2D velocity field generation tasks.

2. Proposed Methods

In this section, we provide an overview of the proposed framework first. Subsequently, we elucidate approaches adopted for the generation of 2D velocity field data and corresponding sketch data. Lastly, we present a comprehensive discussion on the specifics of auto-encoders and LDM structure.

2.1 Overview

The proposed framework is shown in Figure 2. Inspired by sketch-guided diffusion models such as human face generation structure [7], we aim to apply sketches as a control condition for the generation of 2D flow velocity fields.

2.2 Data Generation

The velocity field data that applied in this research are extracted from 2D smoke simulation scenarios. In the smoke simulation, the particles are moved through advection, influenced by velocity $\mathbf{u}$ and pressure p at time step t. Assuming that $\mathbf{u}$ and p are given as initial conditions at time step 0, we can compute the updated velocity field by giving an inviscid Navier-Stokes model, which is shown below:

$$\begin{aligned} \frac{\partial u_x}{\partial t} &= -\mathbf{u}\cdot\nabla u_x - \frac{1}{\rho}\nabla p, \\ \frac{\partial u_y}{\partial t} &= -\mathbf{u}\cdot\nabla u_y - \frac{1}{\rho}\nabla p + b, \end{aligned} \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2}$$

Where u_x is the x direction of given velocity, u_y is the y direction of given velocity, b is the buoyancy. Given that the smoke simulation takes place inside a rectangle obstacle domain, where the boundary is regarded as solid, the boundary condition is defined as follows:

$$\mathbf{u}\cdot\mathbf{n} = 0 \tag{3}$$

where $\mathbf{n}$ is normal vector of the boundary. In this research, the semi-Lagrangian scheme [8] is applied to update the fluid field after inputting the velocity field.

The sketch data utilized herein comprises streamlines extracted from given 2D velocity field data. The streamlines are essentially curves that are tangential to velocity vectors in the velocity field. These can be used to describe the trajectory of any fluid particle in time. This presents a suitable and convenient approach for describing a complex velocity field. The Runge-Kutta method is a common approach to calculate the streamlines. The implemented approach in this research is the Fourth-Order Runge-Kutta method, chosen for its satisfactory in tracing the trajectories while the computation is in a low complexity with minimal parameters. The equations are defined as follows:

$$\begin{aligned} x_{n+1} &= x_n + \frac{h}{6}(k_{1x} + 2k_{2x} + 2k_{3x} + k_{4x}) \\ y_{n+1} &= y_n + \frac{h}{6}(k_{1y} + 2k_{2y} + 2k_{3y} + k_{4y}) \end{aligned} \tag{4}$$

$$\begin{cases} k_{4x}, k_{4y} = f(x_n + hk_{3x}, y_n + hk_{3y}) \\ k_{3x}, k_{3y} = f(x_n + \frac{h}{2}k_{2x}, y_n + \frac{h}{2}k_{2y}) \\ k_{2x}, k_{2y} = f(x_n + \frac{h}{2}k_{1x}, y_n + \frac{h}{2}k_{1y}) \\ k_{1x}, k_{1y} = f(x_n, y_n) \end{cases} \tag{5}$$

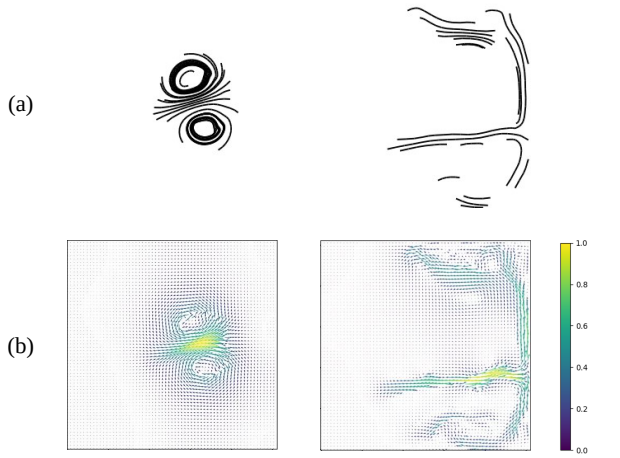

Fig. 3 The samples of velocity field and extracted sketch. Row (a) is extracted sketch data, and Row (b) is velocity field shown in vector field style.

where x_n, y_n are given as the positions of particle at current status n, x_{n+1}, y_{n+1} are the positions of particle at next status $n+1$. h is given the time step, k_{1x}, k_{1y} are the slopes in x and y direction at start point, $k_{2x}, k_{2y}, k_{3x}, k_{3y}$ are the slopes at the middle points, k_{4x}, k_{4y} are the slopes at the end point.

Samples of velocity field and sketch data are given as Figure 3 shows: where column (a) is velocity field samples, and column (b) is extracted sketch samples. Here we use Matplotlib tools to visualize the velocity field as a vector field. The direction of vector represents the velocity direction of each grid. The length and color of vector represent the speed of each grid. The color distributes from 0 to 1 linearly, the color is lighter, the speed is faster.

2.3 Velocity field & Sketch Auto-Encoder

With the training data generated, it is necessary to implement 2 auto-encoders for training. Here the widely used auto-encoder structure is adopted for integration into the proposed framework. This choice is informed by its ease of control and modification. Additionally, the auto-encoder exhibits strong performance in feature learning and data dimensionality reduction. These two auto-encoders share an identical structure due to the uniform size of velocity field and sketch. The structures of sketch auto-encoder and velocity field auto-encoder are shown in Figure 4. The loss function of sketch auto-encoder and velocity field auto-encoder are given as follows:

$$\begin{aligned} loss &= \|X_i - \hat{X}_i\|^2, \\ \hat{X}_i &= \mathcal{D}(\mathcal{E}(X_i)) \end{aligned} \tag{6}$$

where X_i is input data, $\hat{X}_i$ is output recovered data, $\mathcal{E}$ is the encode process, $\mathcal{D}$ is the decode process.

2.4 Latent Diffusion Model

Diffusion model (DM) has rapidly evolved in recent periods and has become one of the most outstanding structures among current generative models after GAN structure. The essence of DM is training a parameterized Markov chain to progressively eliminate Gaussian noise from data mixed with noise. The DM primarily comprises two processes: the noise addition process and the denoising process. The noise addition process infuses Gaussian noise into real data gradually, while the subsequent denoising process, as mentioned above, restores real data. The first process has no learning prerequisite since it follows mathematical laws, whereas the denoising process employs a neural network model for learning. However, DM performs sampling in high-dimensional data space, resulting in a massive computational workload. Hence, the LDM is involved in our research for expediting experiments. The main idea of LDM is to train the denoising process in a low-dimension latent space. This necessitates an encoder, responsible for compressing data into a feature map for transmission to the latent space; and a decoder to restore the feature map back to its original data form.

As Figure2 shows, the velocity field V is given where $V \in \mathbb{R}^{C\times H\times W}$, the encoder $\mathcal{E}$ encodes V into feature map z_0 where $z_0 \in \mathbb{R}^{c\times h\times w}$, z_0 is mixed with the Gaussian noise through the diffusion process to get z_T. In this research, the sketch y serves as an input condition, encoder τ encodes y into the middle representation $\tau(y)$. The $\tau(y)$ undergoes the concatenation with z_T to get a new z_T where $z_T \in \mathbb{R}^{(C+c)\times h\times w}$. During the reverse diffusion process, the U-Net is used to decrease the Gaussian noise within z_T, resulting in the derivation of reconstructed $z_0{}'$. At last, the decoder $\mathcal{D}$ decodes $z_0{}'$ to reconstruct velocity field V'. The loss function of LDM is given as follows:

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x),y,\epsilon\ \mathcal{N}(0,1),t}[\|\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y))\|_2^2] \tag{7}$$

where $\mathcal{E}(x)$ is encoded feature map, y is condition, $\tau_\theta(y)$ is encoded condition, $\epsilon_\theta(..., t)$ is neural backbone that usually is implemented as time conditional U-Net[9].

3. Experiments and Results

In this section, we introduce how the data generation process and network training are implemented. We compare our framework with cGAN model and provide both quantitative and qualitative evaluations to validate the quality of our approach.

3.1 Dataset Generation

The dataset generation process is executed on a Windows system equipped with an i9-12900K CPU. We implemented 2D smoke plume simulation scenes with Phiflow [10] which is a simulation tool prepared for the Python environment. The size of simulation scene is 256×256. The area is surrounded by a solid boundary, the smoke particle clusters are situated inside the area. The buoyancy is applied to make smoke ascend in the field. In our simulation, we randomly configure parameters including initial buoyancy strength and direction, initial smoke cluster position, and smoke cluster size to generate diverse smoke patterns. We simulate 1000 scenes, each span in 150 time steps. Notably, the representation of smoke field varies at different time steps. Therefore we extract velocity fields every 10 time steps after the time step reaches 30. We collect a total of 10000 velocity fields, with each velocity field being exported as .npy files. These data are organized in 2 channels with the size of 256×256. The 2 channels represent the x direction and y direction of velocity field. The

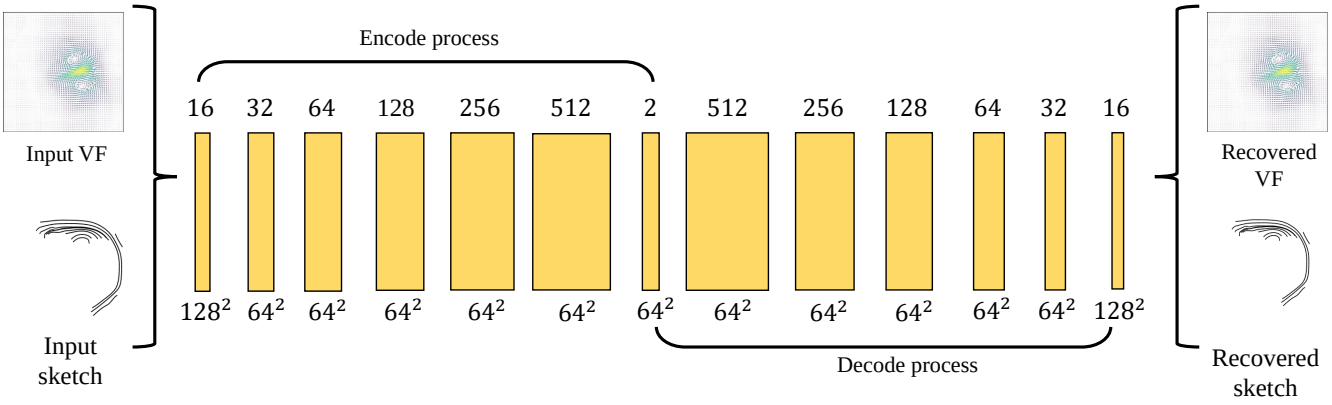

Fig. 4 The structures of auto-encoders for both velocity fields and sketch inputs.

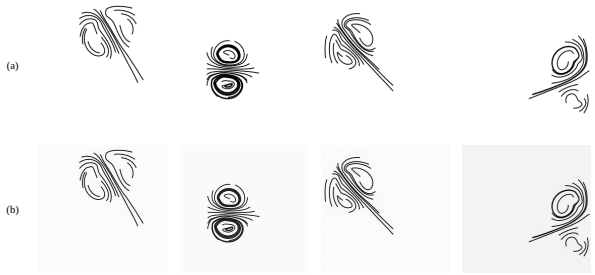

Fig. 5 Sketch reconstruction visualization result. Row (a) is real data, row (b) is reconstructed data.

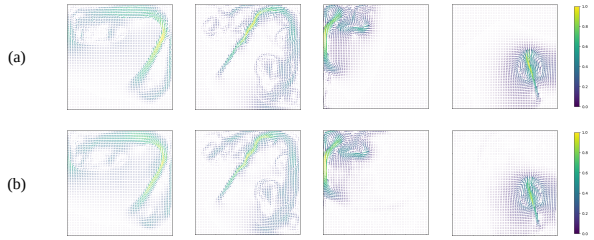

Fig. 6 Velocity field reconstruction visualization result. Row (a) is real vectorized velocity field data, row (d) is reconstructed vectorized velocity field data.

entire simulation time is about 25 hours. streamline images are generated as the corresponding sketches aligned with exported velocity fields. Considering the impracticality of tracing the trajectories for every individual particle, we adopt an approach of tracing the trajectories of particles inside the top 512 grids of velocity field with the highest velocities. This is achieved through the application of a filter mask to block unnecessary grids. The generated sketch data encompasses a single channel with the size of 256×256.

3.2 Auto-Encoder Implementation

The training of auto-encoders for both velocity field and sketch is conducted on a Linux system equipped with NVIDIA 3090 GPU. First, we focused on the sketch auto-encoder. The model is trained for 500 epochs with an Adam optimizer, and a batch size of 16 is adopted. The dataset is split into 8:2, with 8,000 data allocated for training and 2,000 data for testing. The result of sketch reconstruction is given in Figure 5. Notably, sketch data is accurately reconstructed. However, instability exists in the background color. Next is the velocity field auto-encoder. This model invokes the same parameters as the previously discussed sketch auto-encoder. Preceding the training, the normalization operation is operated on velocity field data to ensure the stability of training. The sketch reconstruction result is given in Figure 6. In Figure 6, we can see that velocity field data are reconstructed with a high degree of accuracy, albeit with minor discrepancies in details.

3.3 Comparison

Regarding the training of the LDM, a total of 250 epochs are executed using an Adam optimizer. Similar to previous steps, the batch size is 16, 8,000 data for training and 2,000 data for testing. We compare the proposed framework with Pix2Pix [11] to assess the performance of the proposed framework. For training Pix2Pix, the same dataset is applied, and partitioned into an 8:2 ratio for training and testing. The batch size is adjusted to 8, the total epoch is 300. Figure 7 illustrates a visual comparison between the two frameworks. In Figure 7, row (b) demonstrates that the velocity fields generated by Pix2Pix exist a large amount of noise. In contrast, our framework generates velocity fields that match the shape of input sketch conditions even if the strength of flow is not matched. The generation task is an ill-posed problem, it is impossible that sketches match the velocity field. In addition, we input some hand-drawn sketches to generate velocity fields with the proposed framework, the result is shown in Figure 8. This demonstrates even with sketch patterns not existing in the dataset, our framework successfully generates velocity fields that correspond to the shape of sketches. For a quantitative evaluation, refer to Table 1.

Table 1 MSE Loss comparison

	MSE Loss
Ours	**0.086**
Pix2Pix	26.372

4. Conclusion

In this work, we propose a sketch-guided 2D velocity field generation framework based on LDM. Our framework successfully generates velocity fields that align with provided sketches while maintaining a satisfactory level of accuracy in capturing the sketch's shape. We conducted a comparison against the basic cGAN architecture. The results revealed that our approach sig-

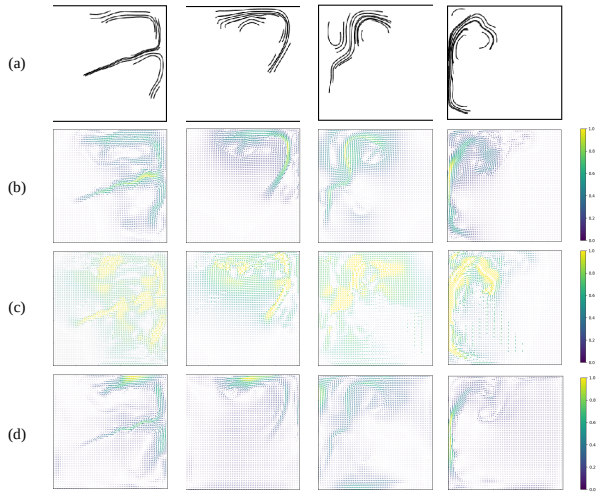

Fig. 7 Vectorized velocity field comparison result. The row (a) is real data, row (b) vectorized real velocity field, row (c) is vectorized velocity field generated by Pix2Pix, row (d) is vectorized velocity field generated by the proposed framework.

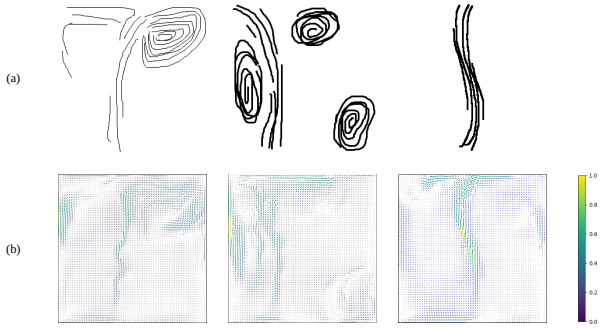

Fig. 8 Hand-drawn sketch-guided velocity field generation visualization result. Row (a) is input hand-drawn sketch, row (b) is generated velocity fields that shown as vector fields.

nificantly outperforms the compared cGAN architecture in terms of performance.

The proposed framework still has limitations outlined below. The flow types within our dataset remain limited. There is potential for improvement by incorporating additional factors like vortex positioning and rotation to exert influence over smoke simulations. Also, the proposed framework is insensitive to small flow patterns with vortexes as shown in Figure 9. The representation of the vortex needs optimization. The current study constitutes a one-stage model wherein the velocity field is generated directly from a streamlined sketch image. However, the process of converting a sketch into a velocity field involves multiple sub-steps. Notably, recent research [12] has proved that training these substeps individually generates better results compared to training as a single step.

For potential future work, it will be crucial work to reconfigure the current one-stage structure into a multi-stage framework for enhancing performance. Furthermore, strengthening the alignment between sketch data and velocity field is a pressing issue. In addition, we intend to enlarge the flow patterns for generating more velocity field types.

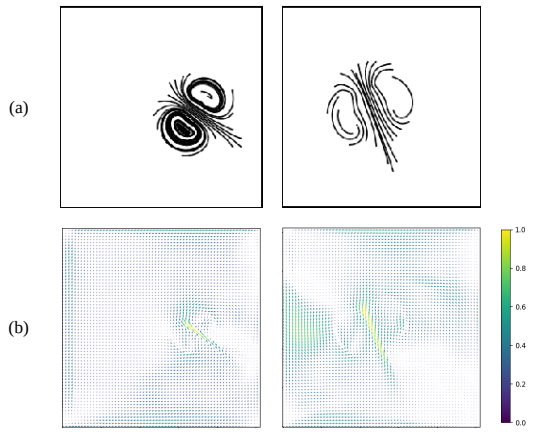

Fig. 9 Failed generations. The row (a) is input sketch, row (b) is generated vectorized velocity field.

References

[1] Zhu, B., Iwata, M., Haraguchi, R., Ashihara, T., Umetani, N., Igarashi, T. and Nakazawa, K.: Sketch-based dynamic illustration of fluid systems, *Proceedings of the 2011 SIGGRAPH Asia Conference*, pp.1–8 (2011).

[2] Xing, J., Kazi, R.H., Grossman, T., Wei, L.Y., Stam, J. and Fitzmaurice, G.: Energy-brushes: Interactive tools for illustrating stylized elemental dynamics, *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pp.755–766 (2016).

[3] Hu, Z., Xie, H., Fukusato, T., Sato, T. and Igarashi, T.: Sketch2VF: Sketch-based flow design with conditional generative adversarial network, *Computer Animation and Virtual Worlds*, Vol.30, No.3-4, e1889, Wiley Online Library, (2019).

[4] Yan, G., Chen, Z., Yang, J. and Wang, H.: Interactive liquid splash modeling by user sketches, *ACM Transactions on Graphics (TOG)*, Vol.39, No.6, pp.1–13 (2020).

[5] Ho, J., Jain, A. and Abbeel, P.: Denoising diffusion probabilistic models, *Advances in neural information processing systems* System Programming Series, Addison-Wesley, Vol.33, pp.6840–6851 (2020).

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

[7] Peng, Y., Zhao, C., Xie, H., Fukusato, T. and Miyata, K.: DiffFaceSketch: High-Fidelity Face Image Synthesis with Sketch-Guided Latent Diffusion Model, *arXiv preprint arXiv:2302.06908* (2023).

[8] Stam, J.: Stable fluids, *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* pp.121–128 (1999).

[9] Ronneberger, O., Fischer, P. and Brox, T.: U-net: Convolutional networks for biomedical image segmentation, *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp.234–241, Springer (2015).

[10] Holl, P., Koltun, V. and Thuerey, N.: Learning to control pdes with differentiable physics, *arXiv preprint arXiv:2001.07457*, (2020).

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

[12] Xie, H., Arihara, K., Sato, S. and Miyata, K.: Dualsmoke: Sketch-based smoke illustration design with two-stage generative model, *arXiv preprint arXiv:2208.10906*, (2022).

Appendix

A.1 Sketch Auto-Encoder Training and Testing Loss

The training loss chart and testing loss chart are given in Figure A·1 and Figure A·2. The train loss and test loss begin to converge in tens of epochs. The minimum of train loss and test loss are 0.0001758 and 0.0003304.

A.2 Velocity Field Auto-Encoder Training and Testing Loss

The training and testing loss are given in Figure A·3 and Figure A·4. The minimum of training and testing loss are 0.008314 and 0.02105.

A.3 LDM Training and Testing Loss

The training loss and testing loss are shown in Figure A·5 and Figure A·6. The minimum training and testing loss are 0.0836 and 0.08667.

Fig. A·1 Sketch auto-encoder training loss

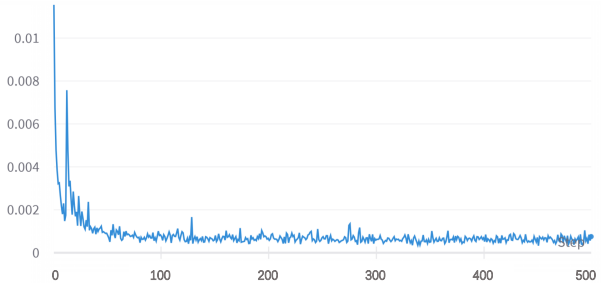

Fig. A·2 Sketch auto-encoder testing loss

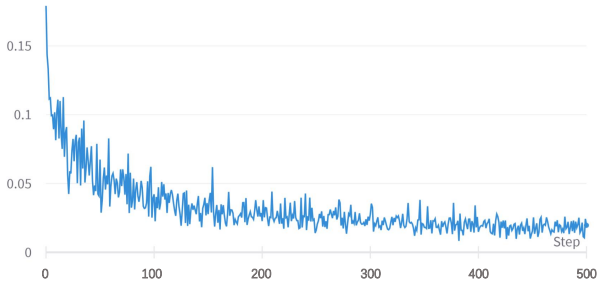

Fig. A·3 Velocity field auto-encoder training loss

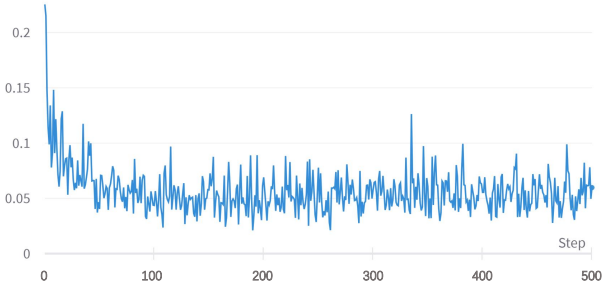

Fig. A·4 Velocity field auto-encoder testing loss

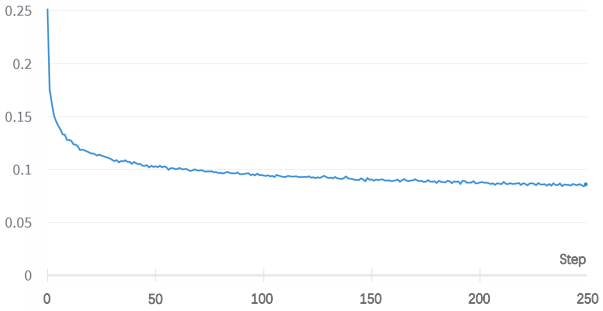

Fig. A·5 LDM training loss

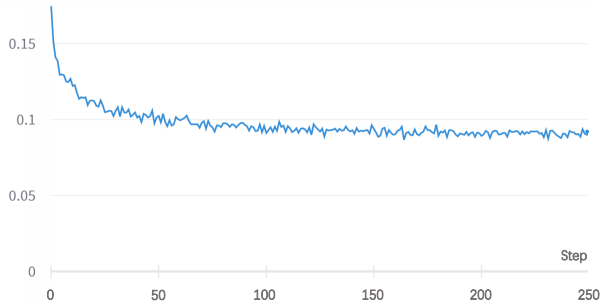

Fig. A·6 LDM testing loss