Vol.2013-CG-153 No.23
Vol.2013-CVIM-189 No.23
2013/11/29

Personal Style Learning in Sumi-e Stroke-based Rendering by Inverse Reinforcement Learning

Ning Xie[1,a)] Tingting Zhao[1,b)] Masashi Sugiyama[1,c)]

Abstract: We consider the problem of automatically generating sumi-e style drawings using machine learning techniques. In our previous work, we regarded a brush as a computer agent and trained the brush to generate smooth strokes to fill given boundaries under a pre-designed cost function. In this paper, we extend this approach and propose to also learn the cost function from a user's real brush stroke data by inverse reinforcement learning. This extension allows the brush agent to imitate the personal drawing style of a user. The effectiveness of our method is demonstrated through experiments.

1. Introduction

Computer graphics has become a popular medium for various purposes such as scientific data visualization, industrial manufacturing, art, and communication. While traditional computer graphics techniques were dedicated to photorealistic rendering, non-photorealistic rendering such as painterly rendering has gathered a great deal of attention because of its artistic and scientific values. In painterly rendering, stroke placement is a common challenge and significant effort has been made to investigate how to draw a stroke with realistic brush texture in a desired shape and how to organize multiple strokes [3].

In this paper, we consider the problem of automatically generating sumi-e style drawings. In our previous work [7], we have developed a highly practical framework for generating expressive appearance of brush strokes using the machine learning technique called *reinforcement learning* [6]. In this framework, a brush is regarded as a computer agent and it is trained to generate smooth strokes to fill given boundaries under a pre-designed cost function. Our system was demonstrated to produce high-quality paintings, given that the cost function is properly designed. However, we have often experienced difficulty to appropriately control tuning parameters included in the cost function.

Recently, there has been an increase interest in personal artistic stylization to synthesize unique strokes or paintings [5], [9]. Most of the studies use a stylus digital pen as an input device, and produce line drawings in the style of a particular artist. However, because the tip of the stylus digital pen is tiny, these methods are not suited to produce sumi-e brush strokes which use a hairy thick tuft brush.

In this paper, we extend our machine learning approach, and propose to also learn the cost function automatically from a user's real brush stroke data. This allows the brush agent to imitate a user's style without manually tweaking control parameters. More specifically, we first gather real brush stroke data from users by using a simply device shown in Fig. 1. This device measures the brush configuration during stroke drawing motions such as the motion attitude, the pose, and locomotion of the brush. We then learn the cost function from the brush stroke data by the machine learning method called *inverse reinforcement learning* [1]. Once the cost function is determined, the optimal control policy of the brush agent is learned by the state-of-the-art reinforcement learning algorithm, IW-PGPE (importance-weighted policy gradients with parameter-based exploration) [8]. IW-PGPE allows us to accurately learn the policy function by efficiently reusing previously corrected data, which is a preferable property in our sumi-e application because collecting data is highly costly. Experimental results show the effectiveness of the proposed method in producing stroke placement with a personalized style.

Our virtual stroke generation system also allows us to synthesize new drawings from photographs in the style of an individual artist. We also demonstrate through experiments the usefulness of our system for converting photographs to computer-generated sumi-e drawings in a given style and abstraction level.

2. Training Data Collection and Processing

In this section, we explain how training data is collected and how it is processed.

2.1 Device for Capturing User's Real Strokes

To learn the stroke drawing style of a particular user, we first collect real stroke data from a user's brush motion and resulting drawings on the paper. In each data-gathering session, we display a reference list of eight basic stroke patterns shown in Fig. 2. A user is instructed to draw several strokes for each pattern on the glass panel of our stroke capturing device using an inked sumi-e soft hairy tuft brush, as shown in Fig. 1. In our experiment, seven

[1] Tokyo Institute of Technology, Tokyo, 152-8550 Japan
[a)] xie@sg.cs.titech.ac.jp
[b)] tingting@sg.cs.titech.ac.jp
[c)] sugi@cs.titech.ac.jp

(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 2 Real data collected from two users for eight basic patterns. Each raw corresponds to each user, where difference in their drawing styles can be observed.

(a) (b) (c) (d)

Fig. 3 Footprint capturing process. (a) Captured footprint. (b) Footprint extraction. (c) The bounding box of the footprint and the principle axis. (d) A matched template of the footprint.

Fig. 1 Illustration of our footprint capturing device.

participants are invited for data gathering.

We constructed a device to measure the dynamics of drawing strokes by recording the brush motion, as shown in Fig. 1. A digital single-lens reflex (DSLR) camera is mounted at the bottom of the frame of the device under ordinary in-door lighting without any automatic camera calibration in real-time. In this experiment, we use a traditional Japanese calligraphy paper placed on the transparent glass panel on the top of the device. To capture the real brush motion, the participants dipped the brush into the traditional calligraphy ink before drawing strokes.

2.2 Data Processing

We split the recorded video of the stroke drawing into frames to analyze the movement of stroke drawing (Fig. 3 (a)). To each frame, we apply the model-based tracking technique [2] and detect the configuration of the brush footprint such as the brush movement information (the velocity, heading direction, and pose) and the relative location information to the target desired shape over time (Fig. 3 (b)). We then apply principal component analysis (PCA) [4] and extract the principal axis of the footprint which decides the direction of the footprint, as shown in Fig. 3 (c). Finally, as shown in Fig. 3 (d), the configuration of the footprint is determined by matching the template of the footprint which consists of a tip Q and a circle with center C and radius r.

Each stroke $\mathbf{L}$ consists of a sequence of footprints F with respect to the time step t:

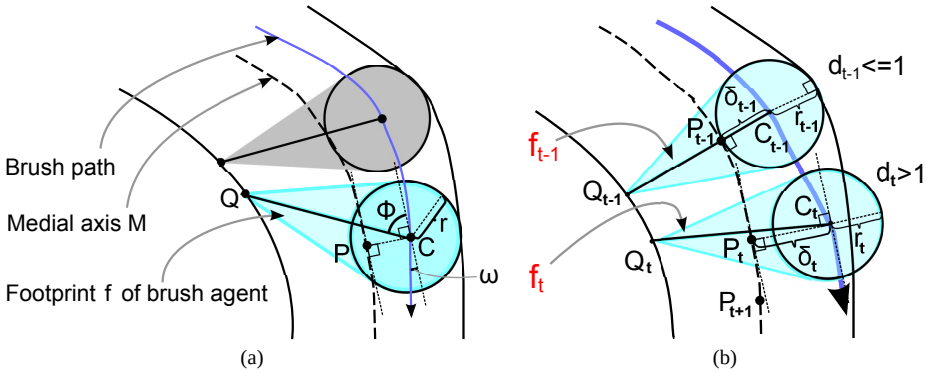
$$\mathbf{L} = \{F_1, F_2, \ldots, F_t\}.$$

3. Brush Agent Training by Reinforcement Learning

In this section, we explain how the problem of automatic stroke drawing can be handled by reinforcement learning [6].

A control rule of the brush agent is called a policy in reinforcement learning, which is a mapping from a current state $\boldsymbol{s}$ of the agent to an action $\boldsymbol{a}$ to be taken. The goal of reinforcement learning is to obtain the best policy that minimizes a cost function. In reinforcement learning, the negative cost function is called the reward function and the reward function is maximized equivalently.

We define the state feature $\boldsymbol{s}$ based on the current surrounding



(a) (b)

Fig. 4 Illustration of the design of states: (a) Brush agent and its path. The brush agent consists of a tip Q and a circle with center C and radius r. (b) The ratio d of the offset distance δ over the radius r. Footprint f_{t-1} is inside the drawing area. The circle with center C_{t-1} and the tip Q_{t-1} touch the boundary on each side. In this case, $\delta_{t-1} \leq r_{t-1}$ and $d_{t-1} \in [0, 1]$. On the other hand, f_t goes over the boundary, and $\delta_t > r_t$ and $d_t > 1$. In our implementation, we restrict d to be in $[-2, 2]$. P is the nearest point on medial axis M to C.

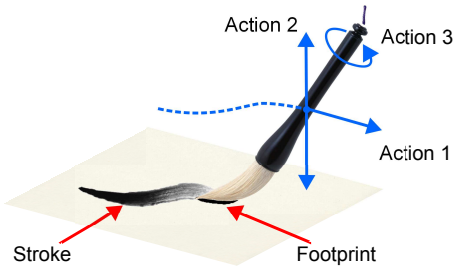


Fig. 5 A stroke is generated by moving the brush with three actions: Action 1 is regulating the direction of the brush movement, Action 2 is pushing down/lifting up the brush, and Action 3 is rotating the brush handle. (b) Captured footprint.

shape and the upcoming shape as

$$\boldsymbol{s} = (\omega, \phi, d, \kappa_1, \kappa_2, l)^\top,$$

where each feature element is defined as follows:

- $\omega \in (-\pi, \pi]$: The angle of the velocity vector of the brush agent relative to the medial axis of the stroke (see Fig. 4(a)).
- $\phi \in (-\pi, \pi]$: The heading direction of the brush agent relative to the medial axis (see Fig. 4(a)).
- $d \in [-2, 2]$: The ratio of offset distance δ (see Fig. 4(b)) from the center C of the brush agent to the nearest point P on the medial axis M over the radius r of the brush agent ($|d| = \delta/r$). d takes positive/negative values when the center of the brush agent is on the left-/right-hand side of the medial axis:
  - d takes the value 0 when the center of the brush agent is on the medial axis.
  - d takes a value in $[-1, 1]$ when the brush agent is inside the boundaries (for example, d_{t-1} in Fig. 4(b)).
  - The value of d is in $[-2, -1)$ or in $(1, 2]$ when the brush agent goes over the boundary of one side (for example, d_t in Fig. 4(b)).

  In our system, the center of the agent is restricted within the shape. Therefore, the extreme value of d is ± 2 when the center of the agent is on the boundary.
- $\kappa_1, \kappa_2 \in [0, 1)$: κ_1 provides the current surrounding information on the point P_t, whereas κ_2 provides the upcoming shape information on point P_{t+1}, as illustrated in Fig. 4(b). The values are calculated as

$$|\kappa_i| = \frac{2}{\pi} \arctan\left(\frac{\alpha}{\sqrt{r_i'}}\right),$$

  where α is the parameter to control the sensitivity to the curvature and r_i' is the radius of the curve. More specifically, the value takes 0/negative/positive when the shape is straight/left-curved/right-curved, and the larger the value is, the tighter the curve is. Throughout this paper, we use a fixed value $\alpha = 0.05$.
- $l \in \{0, 1\}$: A binary label that indicates whether the agent moves to a region covered by the previous footprints or not. $l = 0$ means that the agent moves to a region covered by the previous footprint. Otherwise, $l = 1$ means that it moves to an uncovered region.

On the other hand, the action $\boldsymbol{a}$ is defined as a 3-dimensional vector as follows (see Fig. 5):

- Action 1: Movement of the brush on the canvas paper.
- Action 2: Scaling up/down of the footprint.
- Action 3: Rotation of the heading direction of the brush.

The cost function is learned by the maximum-margin inverse reinforce learning method [1] from a user's drawing data. Then, we apply the IW-PGPE (importance-weighted policy gradients with parameter-based exploration [8]) reinforcement learning method to obtain the optimal policies under the learned cost function. The agent collects $N = 300$ episodic samples with trajectory length $T = 32$. The discount factor is set at $\gamma = 0.99$, and the learning rate ε is set at $0.1/\|\nabla_{\boldsymbol{\rho}} J_{\boldsymbol{\rho}}\|$.

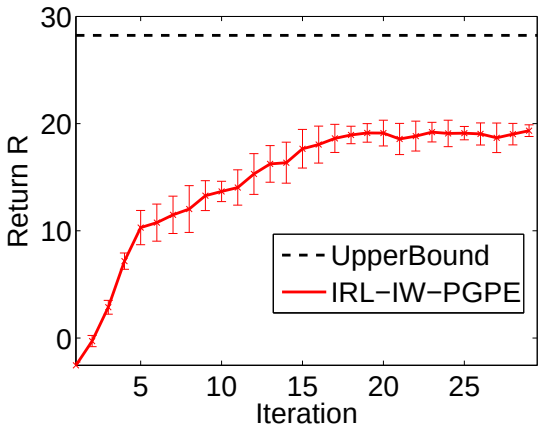


Fig. 6 Return averaged over 16 runs for proposed method and the upper limit of the return value. The error bars denote the standard deviation over 16 runs.

4. Experiments

In this section, we report experimental results. We investigate the return by running the trained agent averaged over 16 runs as the functions of the policy-update iterations. The return at each trial is computed over 300 training episode samples.

Fig. 6 illustrates the performance of policies learned by the proposed method. The graph shows that the average return increases in an early stage and then it converges at around 20th iterations.

Fig. 7 compares the brush trajectories obtained by the method proposed in the current paper (i.e., the reward function is learned from a user's data) and the method used in the previous work [7] (i.e., the reward function is manually designed by the authors). This shows that the proposed method keeps the attitude of the brush more stably than the existing method.

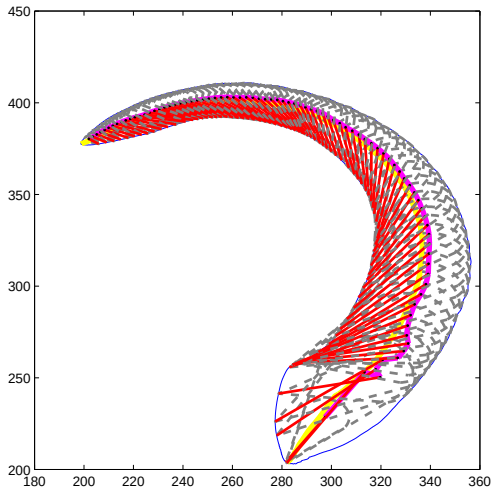
Fig. 8 compares the stroke-drawing processes by a real user, the agent trained with the learned reward function, and the agent trained with the manually designed reward function. This shows that the proposed method imitates the real user's stroke drawing better than the existing method.

Finally, we apply our proposed method to automatic conversion of photos into a sumi-e style. We manually sketched boundaries of desired strokes and let the brush agent fill the shapes with smooth strokes. The results are shown in Fig 9, demonstrating that the proposed method is promising.

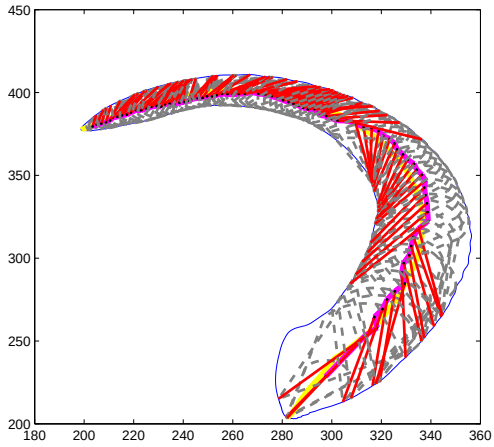
5. Conclusion and Future Work

In this paper, we considered the problem of automatically generating sumi-e style drawings based on reinforcement learning. Our contributions in this papers were as follows: (a) we constructed a device for capturing a user's real brush strokes, (b) we developed a method to measure the brush configuration during stroke drawing motions such as motion attitude, the footprint pose, and the locomotion of the brush, (c) we proposed to learn the reward function using brush stroke data captured from real users by inverse reinforcement learning, and (d) we demonstrated the usefulness of our proposed system in experiments.

Acknowledgments NX and MS were supported by KAKENHI 23120004 and Kayamori Foundation of Information Science Advancement, and TZ was supported by the MEXT scholarship.



(a) Trained with learned reward function.

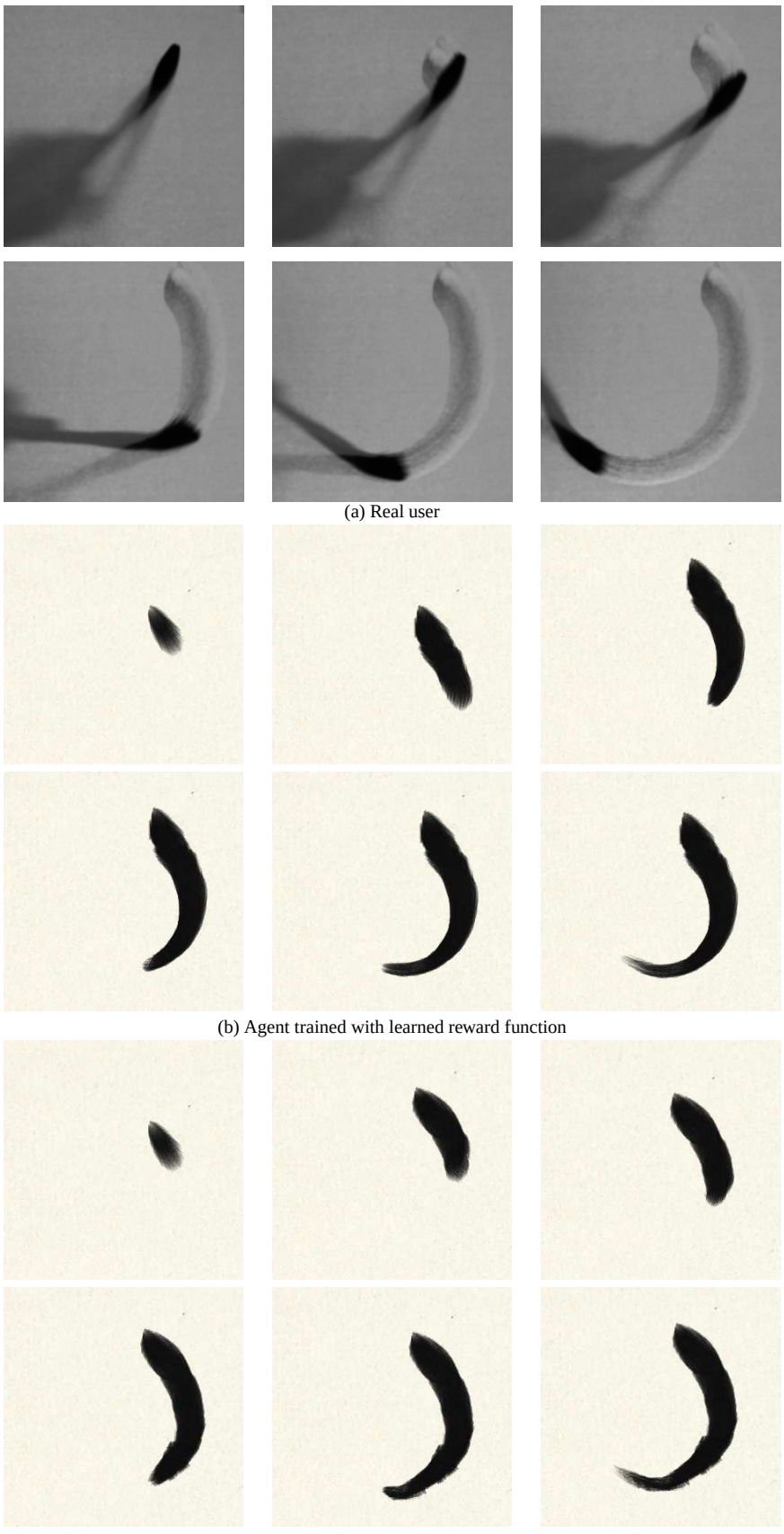


(b) Trained with manually designed reward function.

Fig. 7 Comparison of brush trajectories.

References

[1] Abbeel, P. and Ng, A. Y.: Apprenticeship learning via inverse reinforcement learning, *ICML* (2004).
[2] Davies, E.: *Machine Vision: Theory, Algorithms, Practicalities, Third Edition*, Elsevier (2005).
[3] Fu, H., Zhou, S., Liu, L. and Mitra, N.: Animated Construction of Line Drawings, *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH ASIA)*, Vol. 30, No. 6 (2011).
[4] Jolliffe, I. T.: *Principal Component Analysis*, Springer Verlag, New York (2002).
[5] Lu, J., Yu, F., Finkelstein, A. and DiVerdi, S.: HelpingHand: example-based stroke stylization, *ACM Trans. Graph.*, Vol. 31, No. 4, pp. 46:1–46:10 (online), DOI: 10.1145/2185520.2185542 (2012).
[6] Sutton, R. S. and Barto, A. G.: *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, MA (1998).
[7] Xie, N., Hachiya, H. and Sugiyama, M.: Artist Agent: A Reinforcement Learning Approach to Automatic Stroke Generation in Oriental Ink Painting, *ICML* (2012).
[8] Zhao, T., Hachiya, H., Tangkaratt, V., Morimoto, J. and Sugiyama, M.: Efficient Sample Reuse in Policy Gradients with Parameter-Based Exploration., *Neural Computation*, Vol. 25, No. 6, pp. 1512–1547 (2013).
[9] Zitnick, C. L.: Handwriting beautification using token means, *ACM Trans. Graph.*, Vol. 32, No. 4, pp. 53:1–53:8 (online), DOI: 10.1145/2461912.2461985 (2013).



(a) Real user

(b) Agent trained with learned reward function

(c) Agent trained with manually designed reward function

Fig. 8 Comparison of stroke-drawing processes.

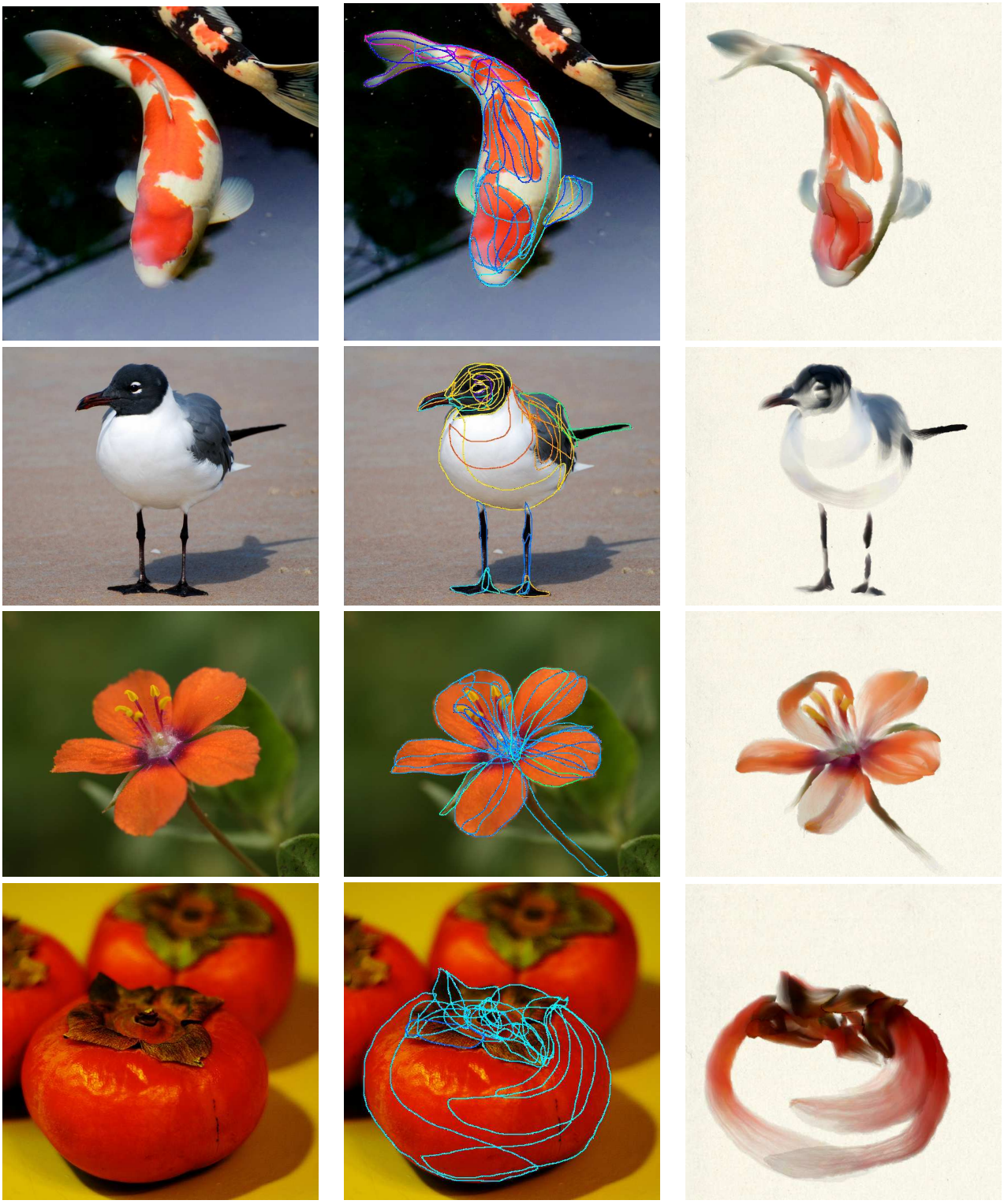


Fig. 9 Results of photo conversion into Sumi-e Style.