2014 April 22

Reinforcement Learning and its Applications in Computer Graphics

Ning Xie

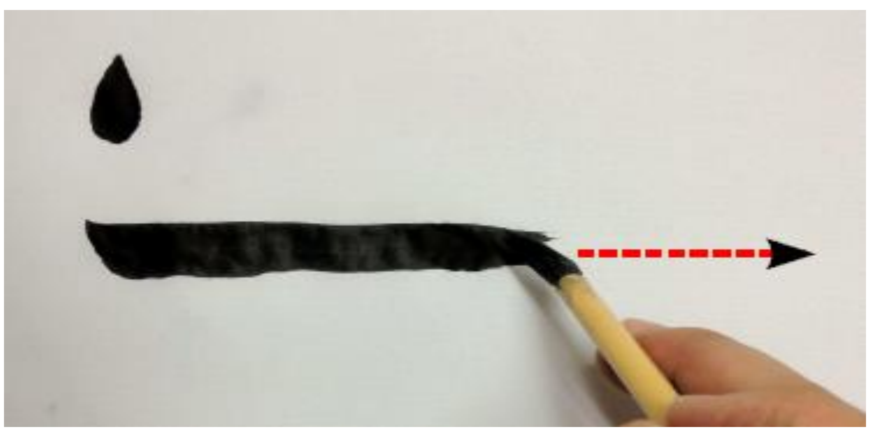
Sugiyama Lab

xie@sg.cs.titech.ac.jp

What is Reinforcement Learning

- A subfield of machine learning
  - "Sampling based methods to solve optimal control problems" – Richard Sutton
- A learning problem
  - In an interactive unknown environment, an RL agent learns the optimal policy for taking actions to control the system.

Example: Control a virtual painting brush

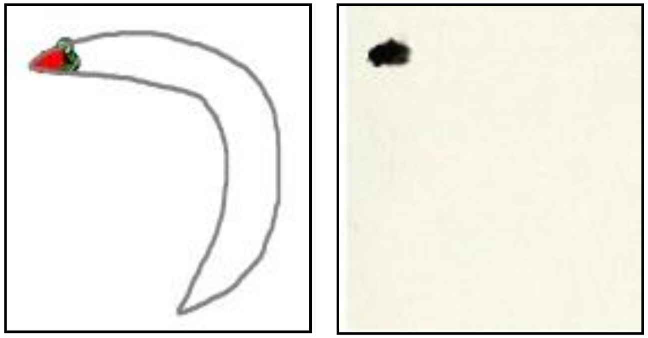


Input

Output

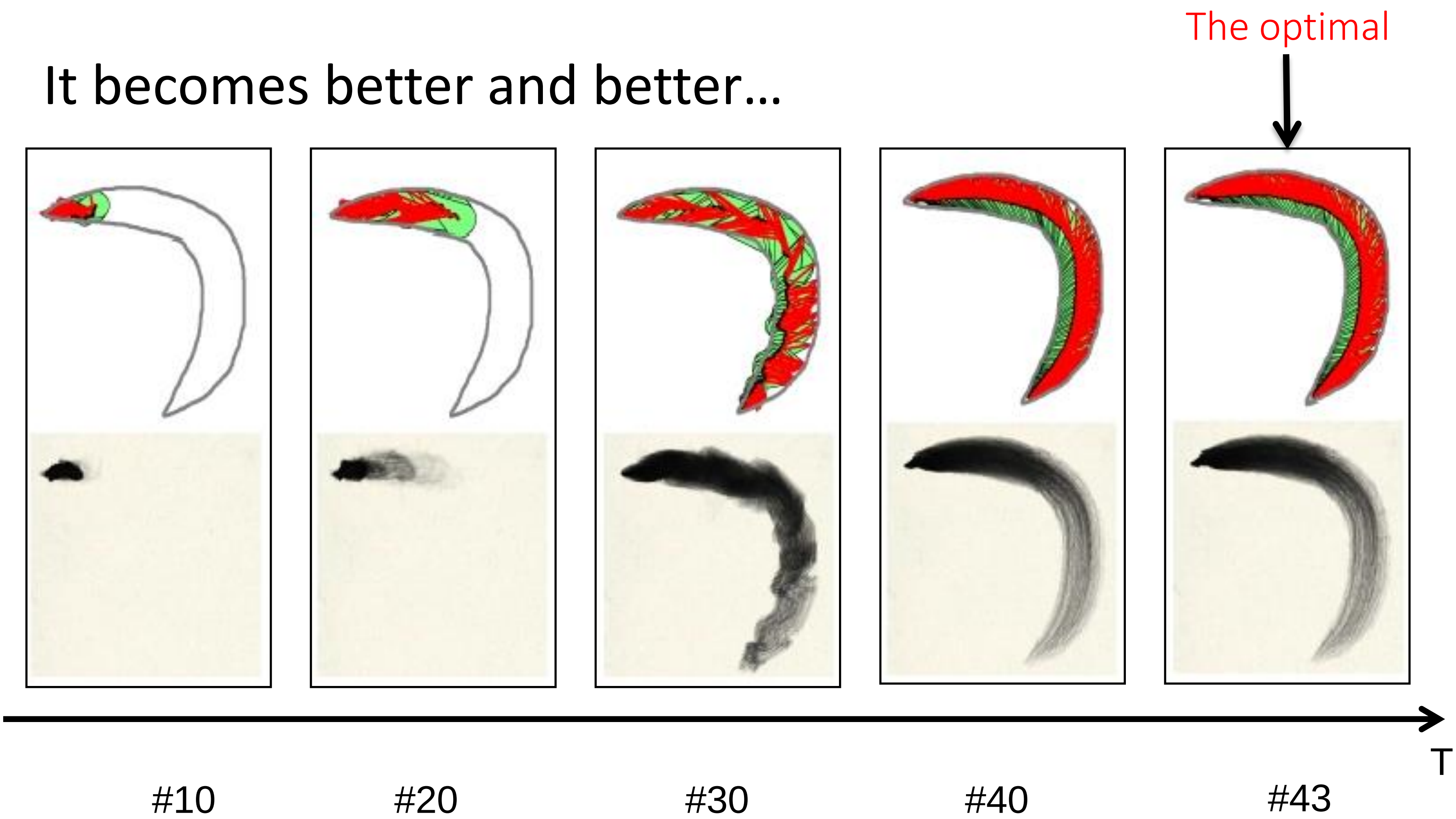
Initial situation

Obviously, it is not that good…



In Learning Process

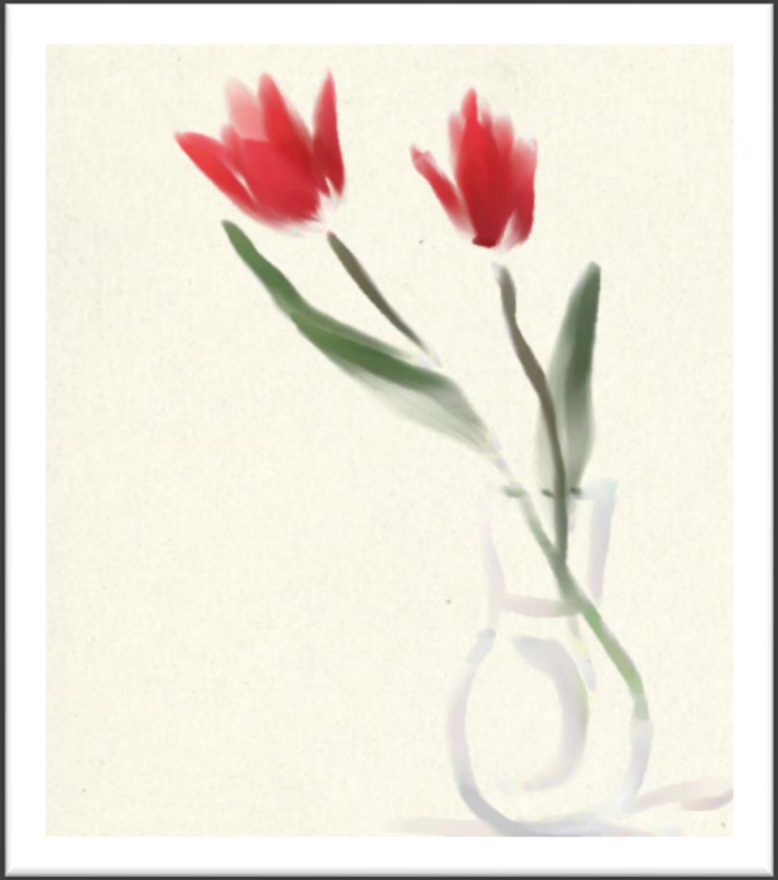
It becomes better and better…



In Learning Process

Drawn by our Agent

Input photo



Rendering result

Many Real World Applications

- Industrial control
- Production control
- Automotive control
- Autonomous vehicles control
- Logistics
- Telecommunication networks
- Sensor networks
- Robotics
- Finance
- …

Agenda

1. RL Background

2. Algorithms for RL

3. Application Example

Agenda

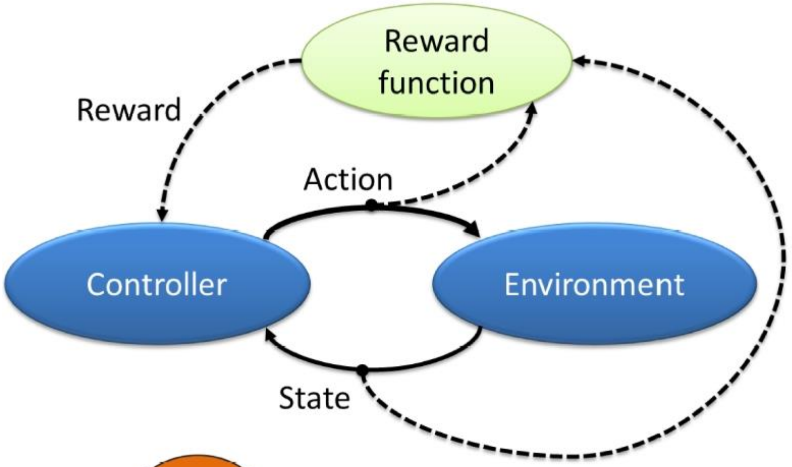
1. RL Background
2. Algorithms for RL
3. Application Example

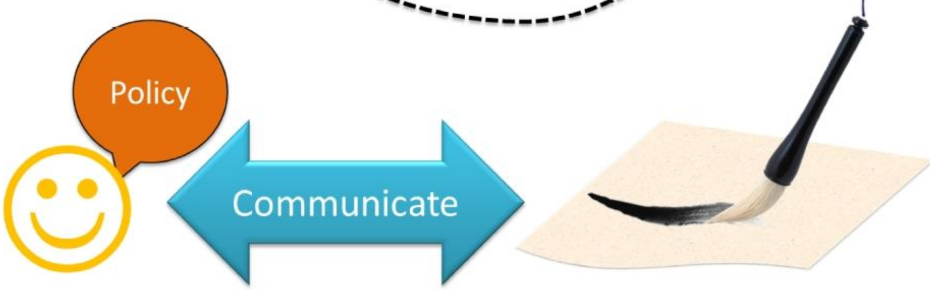
Abstract Control Model

- Adapt and improve the behavior of an agent through trial and error interactions with some environment.(Kealbing et al., 1996)
- **It likes baby learning process**
- Goal
  - Find the optimal policy so as to maximize the cumulative rewards.

Abstract Control Model

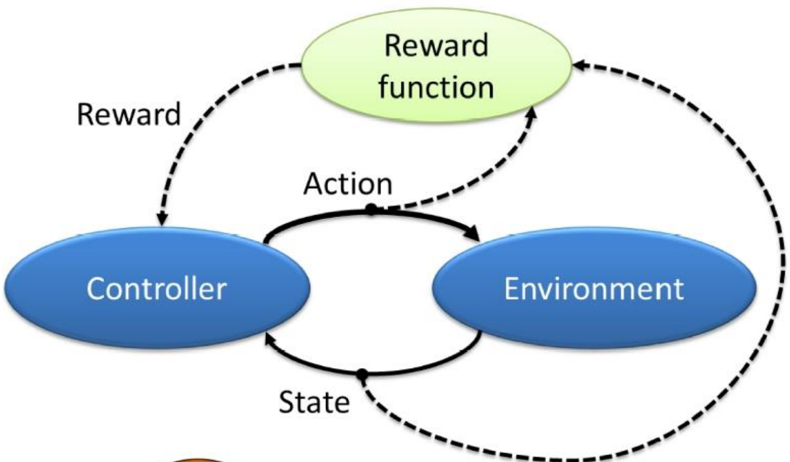
- Controller
  - A decision-making algorithm
- Environment
  - A visualized 2D brush model
  - A paper canvas
  - An painting composition
- Reward function
  - The rule to generate the quality of states transition given an action

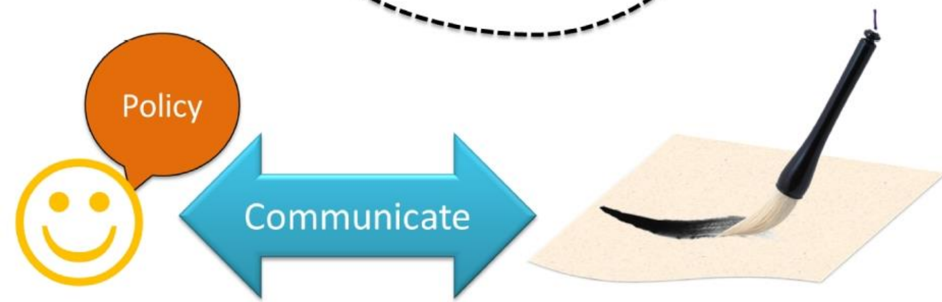




Abstract Control Model

- **Action**
  - Change the environment so as to influence the process of stroke generation
- **State**
  - Characterize the environment
- **Reward**
  - Quality of the state transition on its immediate performance (0<r<1)
- **Policy**
  - The behavior of the agent





Mathematical Formulation

Markov Decision Process

$$(\mathcal{S}, \mathcal{A}, p_{\mathrm{I}}, p_{\mathrm{T}}, r)$$

$\mathcal{S}$: A set of states

$\mathcal{A}$: A set of actions

p_{I} : Initial state probability density

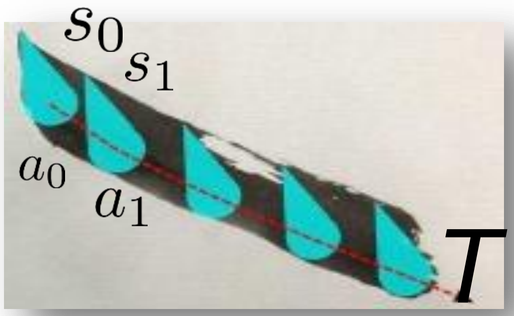
$p_{\mathrm{T}}(\boldsymbol{s}', |\boldsymbol{s}, a)$: Transition probability density

$r(\boldsymbol{s}, a, \boldsymbol{s}')$: Immediate reward function

- Trajectory: $h = (\boldsymbol{s}_1, a_1, \ldots, \boldsymbol{s}_T, a_T, \boldsymbol{s}_{T+1})$

T: length of trajectory

The dynamics of an MDP



- Start in some state s_0 and get to choose some action $a_0 \in A$ to take in the MDP.
- The agent then transits into state s_1 according to the transition probability density, $s_1 \sim \boldsymbol{p}_T(s_1 | s_0, a_0)$. Then, the agent makes a decision to pick up the next action a_1. The state will then transit to $s_2 \sim \boldsymbol{p}_T(s_2 | s_1, a_1)$. The process goes on…

The dynamics of an MDP

- The process can be represented as follows:

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} \dots$$

- The sum of discounted rewards over steps is represent as:

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots .$$

- We also call it, return along the trajectory h:

$$R(h) \equiv \sum_{t=1}^{T} \gamma^{t-1} r(\mathbf{s}_{t,}, a_t, \mathbf{s}_{t+1})$$

γ : discount factor for future reward.

$$(0 < \gamma < 1)$$

Policy

- The agent behaves following the policy as an action a in chosen in according to the current state s.
- Two ways to represent
  - Deterministic policy (since an action is selected with probability of 1) $a = \pi(s) \in \mathcal{R}$
  - Stochastic policy (since an action is selected with a certain probability). $\pi(a|s) \equiv prob(a|s) \in [0, 1]$
- The goal of RL is to find out the optimal policy

$$\max_{\pi} \left[\mathbb{E} \left(\sum_{t=0}^{\infty} \gamma^{t-1} R(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}) \right) \right]$$

Value Function

- Define the value function for a policy π according to

$$V^\pi(s_0) = \mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^{t-1} R(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1} \right)$$

The expected sum of discounted rewards upon starting in the state s_0, and taking actions according to π.

$$s_0 \xrightarrow{\pi(s_0)} s_1 \xrightarrow{\pi(s_1)} s_2 \xrightarrow{\pi(s_2)} s_3 \cdots$$

- We define the optimal value function

$$V^*(s) = \max_\pi V^\pi(s)$$

State-action Value Function

- Q function
  - The expected sum of discounted rewards when the agent takes action a_0 in state s_0 following policy π

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{\pi(s_1)} s_2 \xrightarrow{\pi(s_2)} s_3 \cdots$$

$$Q^\pi(s_0, a_0) = \mathbb{E}\left(R(s_0, a_0, s_1) + \sum_{t=1}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \right)$$

- The optimal policy π^*

$$\pi^*(s) = \operatorname*{argmax}_a Q^*(s, a)$$

$$Q^*(s, a) = \max_\pi Q^\pi(s, a)$$

Agenda

1. RL Background

2. Algorithms for RL

    i. Policy Iteration Method

    ii. Policy Search Method

3. Application Example

Agenda

1. RL Background

2. Algorithms for RL

   i. Policy Iteration Method

   ii. Policy Search Method

3. Application Example

Policy Iteration Method

- Use Q function to obtain the optimal policy

Policy Evaluation (Critic) Step:
Calculate the value function $Q^{\pi^L}(s,a)$ according to the current policy π^L

Policy Improvement (Actor) step:
Policy update according to $Q^{\pi^L}(s,a)$

$$\pi(s)^{L+1} \leftarrow \arg\max_a Q^{\pi^L}(s,a)$$

- The process is repeated until policy converges as

$$||\pi^{L+1}(a|s) - \pi^L(a|s)|| \leq \kappa, \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A},$$

Solving Using Bellman Equation

$$Q^\pi(s_0, a_0) = \mathbb{E}\left(R(s_0, a_0, s_1) + \sum_{t=1}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \right)$$

- Bellman equation

$$Q^\pi(s, a) = \mathbb{E}[\quad R(s, a, s') \quad + \quad \gamma Q^\pi(s', \pi(s')) \quad]$$

- However, it is computational intractable
  - The number of state-action pairs $|\mathcal{A}| \times |\mathcal{S}|$ is very large
  - $|\mathcal{S}|$or$|\mathcal{A}|$becomes infinite when state space or action space is continuous
- To overcome the problem, Q function approximation techniques is used.

Linear Q Function Approximation

- The linear model for approximating the Q function $Q^{\pi}(s, a)$

$$\widehat{Q}^{\pi}(s, a; \boldsymbol{\omega}) \equiv \sum_{b=1}^{B} \omega_b \phi_b(s, a) = \boldsymbol{\omega}^{\top} \boldsymbol{\phi}(s, a)$$

- $\mathcal{B}$ is usually set much smaller than $|\mathcal{A}| \times |\mathcal{S}|$ so as to reduce the computational cost on Q function computation.
- To obtain the parameter $\boldsymbol{w}$, *Bellman residual* is used which is as an error criterion of value function approximation. (Schoknecht, 2003; Lagoudakis&Parr, 2003)

Least-squares Policy Iteration (LSPI)

- Using *Bellman-residual-minimizing-approximation* in policy evaluation.

LSPI $(D,\ k,\ \phi,\ \gamma,\ \epsilon,\ \pi_0)$ // Learns a policy from samples

// D : Source of samples (s, a, r, s')
// k : Number of basis functions
// ϕ : Basis functions
// γ : Discount factor
// ϵ : Stopping criterion
// π_0 : Initial policy, given as w_0 (default: $w_0 = 0$)

$\pi' \leftarrow \pi_0$ // $w' \leftarrow w_0$

repeat
  $\pi \leftarrow \pi'$ // $w \leftarrow w'$
  $\pi' \leftarrow \mathbf{LSTD}Q\ (D,\ k,\ \phi,\ \gamma,\ \pi)$ // $w' \leftarrow \mathbf{LSTD}Q\ (D,\ k,\ \phi,\ \gamma,\ w)$
until $(\pi \approx \pi')$ // **until** $(\|w - w'\| < \epsilon)$

return π // **return** w

The Limitation of Policy Iteration Method

Policy Evaluation (Critic) Step:
Calculate the value function $Q^{\pi^L}(s,a)$ according to the current policy π^L

Policy Improvement (Actor) step:
Policy update according to $Q^{\pi^L}(s,a)$

$$\pi(s)^{L+1} \leftarrow \arg\max_a Q^{\pi^L}(s,a)$$

- When state or action is continuous, it is difficult to compute $\arg\max_a Q^{\pi^L}(s,a)$.
- Shall we directly compute the policy without Q function computation?

Agenda

1. RL Background

2. Algorithms for RL

   i. Policy Iteration Method

   ii. Policy Search Method

3. Application Example

Agenda

1. RL Background

2. Algorithms for RL

   i. Policy Iteration Method

   ii. Policy Search Method

3. Application Example

Policy Search Method

- Policy is expressed by a parametric model as $\pi(a|s;\boldsymbol{\theta})$
- Gaussian model:

$$\pi(a|\boldsymbol{s};\boldsymbol{\theta}) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(a-\boldsymbol{\mu}^\top\boldsymbol{s})^2}{2\sigma^2}\right)$$

$\boldsymbol{\theta} = (\boldsymbol{\mu}^\top, \sigma)^\top$: Policy parameter

- The expected return:

$$J(\boldsymbol{\theta}) = \mathbb{E}\left(\sum_{t=0}^{\infty}\gamma^t R(s_t, a_t, s_{t+1})\right)$$

- The goal is to find out the optimal policy parameter $\boldsymbol{\theta}^*$

$$\boldsymbol{\theta}^* \equiv \mathrm{argmax}\ J(\boldsymbol{\theta})$$

Policy Search Method

- Policy gradient method is one of the most popular policy search methods. (Williams,1992)
- Update parameter along the gradient ascents:

$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + \varepsilon \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$	: Gradient of expected
ε	: Learning rate

- The expectation is approximated by the empirical average:

$$\nabla_{\boldsymbol{\theta}} \widehat{J}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi(a_t^{(n)} | \boldsymbol{s}_t^{(n)}, \boldsymbol{\theta}) R(h^{(n)})$$

$\{h^{(n)}\}_{n=1}^{N}$	: are *N* episodic samples with *T*

$$h^{(n)} = (\boldsymbol{s}_1^{(n)}, a_1^{(n)}, \ldots, \boldsymbol{s}_T^{(n)}, a_T^{(n)}, \boldsymbol{s}_{T+1}^{(n)})$$

Policy Search Method

$$\pi(a|\boldsymbol{s};\boldsymbol{\theta}) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(a-\boldsymbol{\mu}^\top \boldsymbol{s})^2}{2\sigma^2}\right)$$

$\boldsymbol{\theta} = (\boldsymbol{\mu}^\top, \sigma)^\top$: Policy parameter

The policy gradient $\nabla_{\boldsymbol{\theta}} \widehat{J}(\boldsymbol{\theta})$ are expressed as

$$\nabla_{\boldsymbol{\mu}} J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}(R(h^{(n)}) - b)\sum_{t=1}^{T}\frac{(a_t^{(n)} - \boldsymbol{\mu}^\top \boldsymbol{s}_t^{(n)})\boldsymbol{s}_t^{(n)}}{\sigma^2},$$

$$\nabla_{\sigma} J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}(R(h^{(n)}) - b)\sum_{t=1}^{T}\frac{\left(a_t^{(n)} - \boldsymbol{\mu}^\top \boldsymbol{s}_t^{(n)}\right)^2 - \sigma^2}{\sigma^3}$$

Policy Search Method

- The optimal baseline is introduced for minimizing the variance of gradient estimates. (Peters and Schaal, 2006)

$$b^* = \underset{b}{\operatorname{argmin}} \mathbf{Var}[\nabla_{\boldsymbol{\theta}} \widehat{J}(\boldsymbol{\theta})]$$
$$\simeq \frac{\frac{1}{N}\sum_{n=1}^{N} R(h^{(n)}) \left\| \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi(a_t^{(n)} | \boldsymbol{s}_t^{(n)}; \boldsymbol{\theta}) \right\|^2}{\frac{1}{N}\sum_{n=1}^{N} \left\| \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi(a_t^{(n)} | \boldsymbol{s}_t^{(n)}; \boldsymbol{\theta}) \right\|^2}$$

- The policy parameter $\boldsymbol{\theta} = (\boldsymbol{\mu}^\top, \sigma)^\top$ is updated as

$$\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \varepsilon \nabla_{\boldsymbol{\mu}} J(\boldsymbol{\theta})$$
$$\sigma \leftarrow \sigma + \varepsilon \nabla_{\sigma} J(\boldsymbol{\theta})$$

- Data collection and policy update are repeated

Agenda

1. RL Background
2. Algorithms for RL
3. Application Example

Agenda

1. RL Background
2. Algorithms for RL
3. Application Example

ICML 2012

Artist Agent:
A Reinforcement Learning Approach to Automatic Stroke Generation in Oriental Ink Painting

Ning Xie
Hirotaka Hachiya
Masashi Sugiyama

Department of Computer Science
Tokyo Institute of Technology

Stroke-based Painterly Rendering

- An automatic approach to creating a painting image by placing discrete strokes.
- Painting styles
  1. Western painting
     - Many overlapped basic strokes in multiple layers
  2. Oriental ink painting
     - A few non-overlapping, long and curved strokes in a single layer

(Zhao et al., Siggraph, 2010)

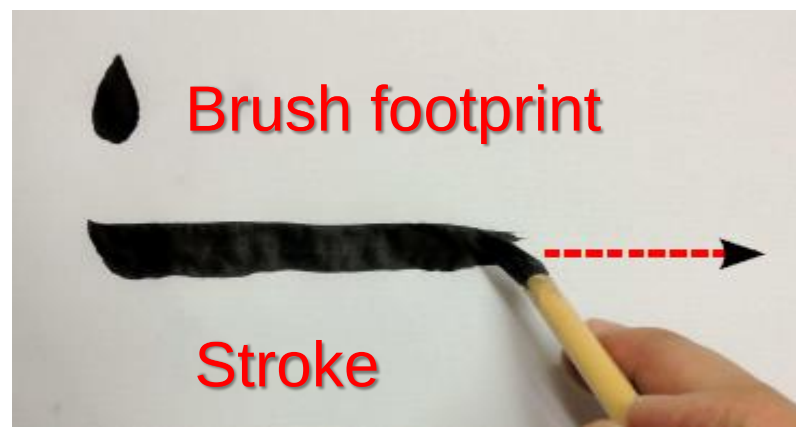
(Ning et al., NPAR, 2010)

Pipeline

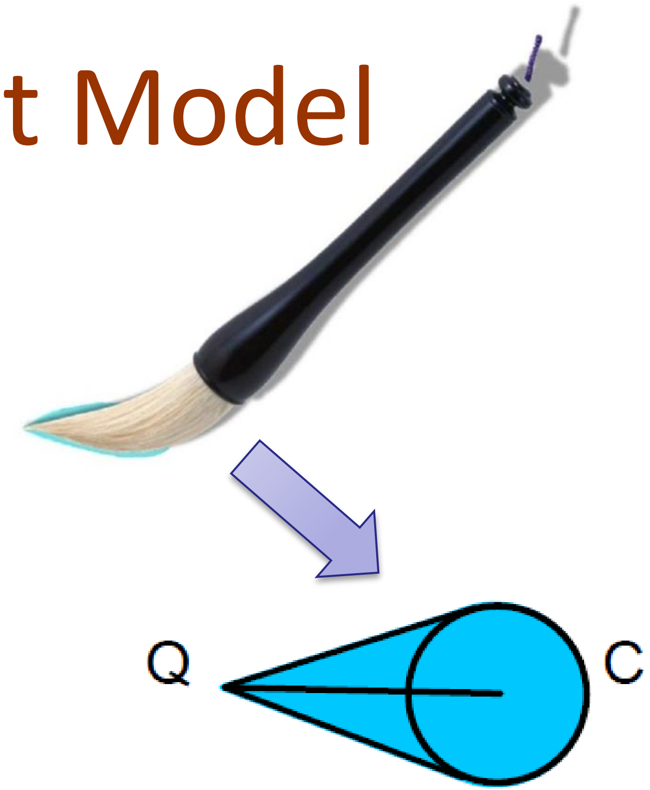
- Introduction
- **Our strategy**
- Model in Reinforcement Learning Framework
- Design of Action, State, Reward and Training Scheme
- Experiments
- Conclusion

Our Strategy

- Challenges
  1. How to obtain the arbitrary shape?
     - **Solution:** Automatic control of a brush model to generate strokes
  2. How to render the natural texture?
     - **Solution:** Simultaneously render the brush footprint texture along the trajectory of the brush



2D Brush Footprint Model



- A footprint f consists of
  - A circle $\{C, r\}$
  - A link $\overline{CQ}$
- A stroke

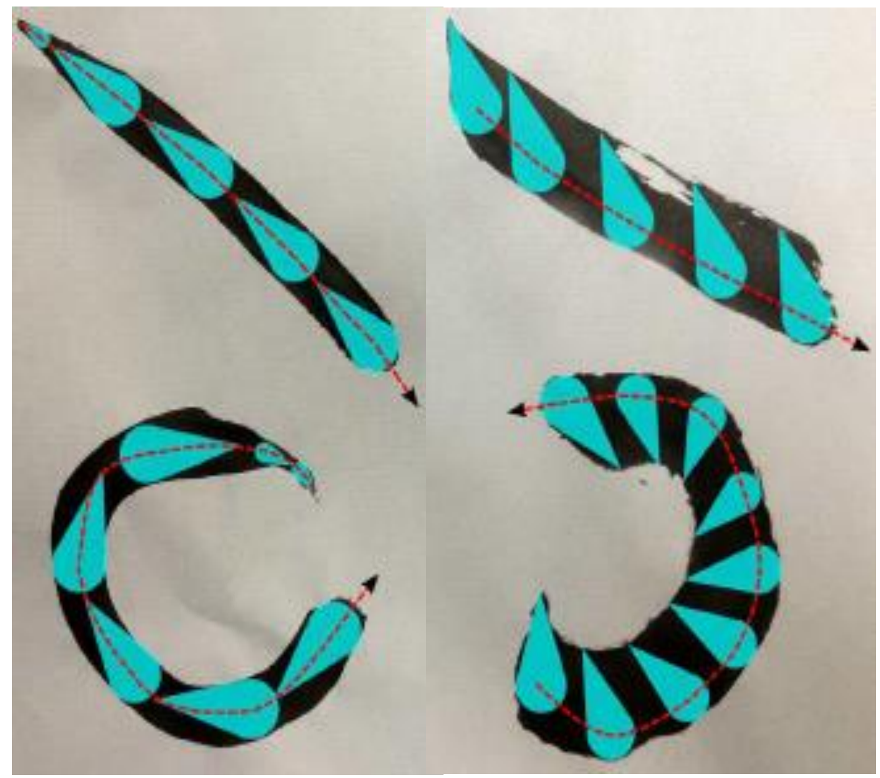
$$F = (f_1, f_2, \dots, f_T)$$

Pipeline

- Introduction
- Our strategy
- **Model in Reinforcement Learning Framework**
- Design of Action, State, Reward and Training Scheme
- Experiments
- Conclusion

Automatic Stroke Generation Problem

- Model 2D brush as an intelligent agent
- Formulate the stroke generation as a Markov decision process (MDP)
- Target
  - Agent learns the optimal control policy to draw strokes

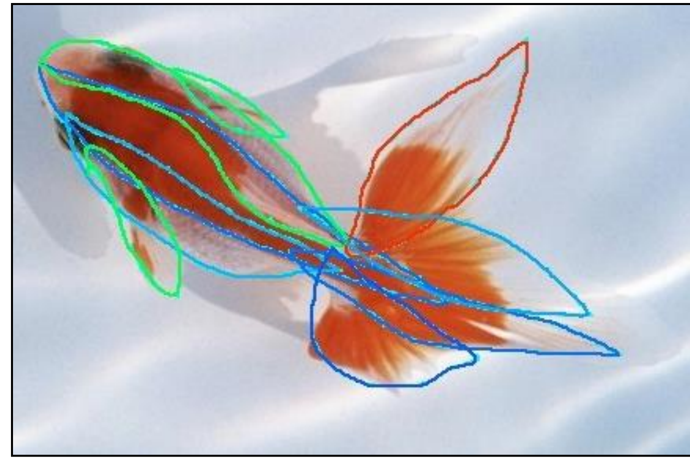


Upright brush style

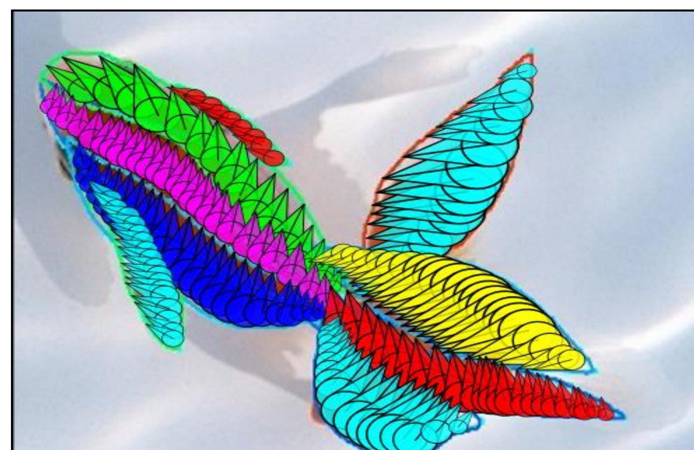
Oblique brush style

Solution with Reinforcement Learning

- System input
  - Painting composition: boundaries placed on the target image.
- The optimal policy should
  - Control the direction of movement
  - Keep a stable posture during sweeping over an arbitrary stroke shape



Painting composition



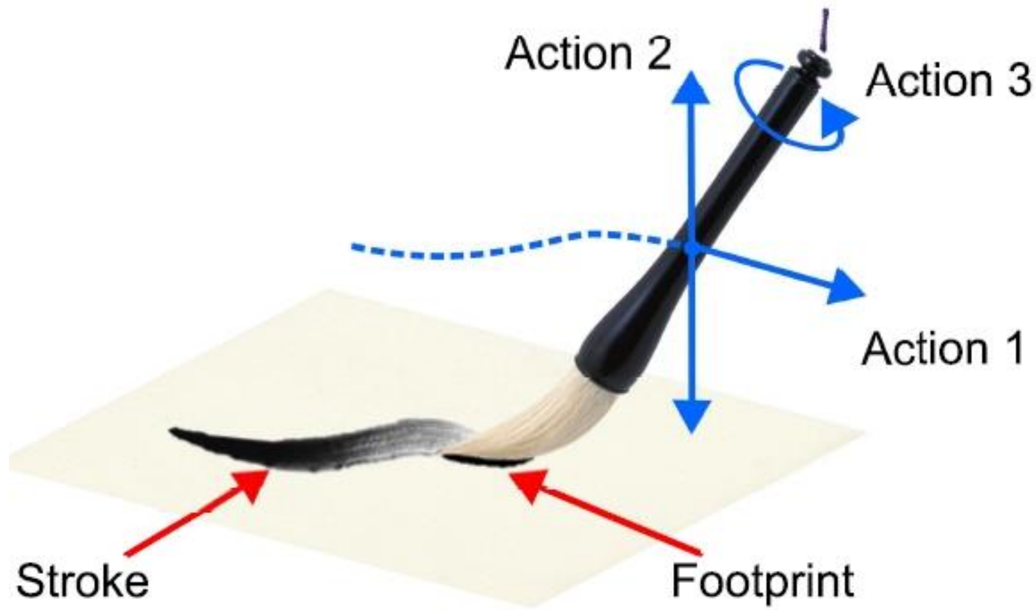
Brush trajectories with footprints

Pipeline

- Introduction
- Our strategy
- Model in Reinforcement Learning Framework
- **Design of Action, State, Reward and Training Scheme**
- Experiments
- Conclusion

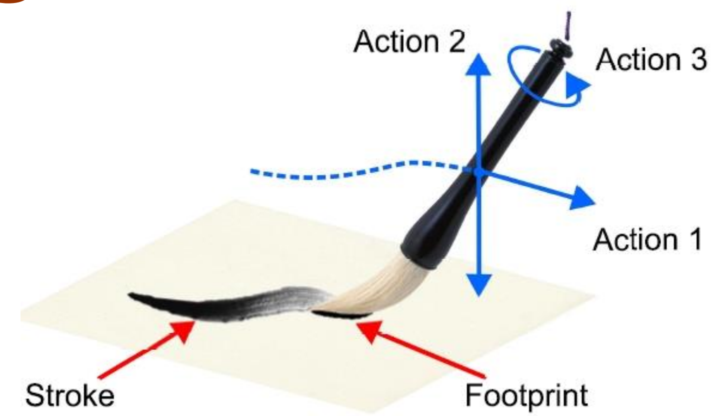
Action Design

- 3-dimensional action to control the brush:
  - Action 1: Movement of the brush on the canvas
  - Action 2: Scaling up/down of the footprint
  - Action 3: Rotation of the heading direction of the brush

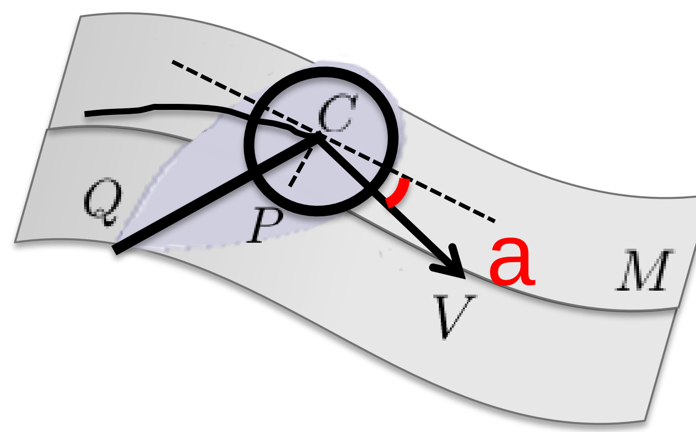


Action Design

- Action 1 is the primary action, since properly covering the whole region is the most important



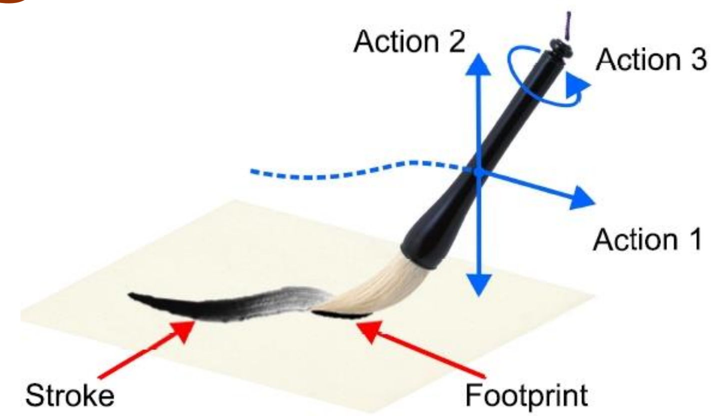
- Setup
  1. The offset angle $a \in [\pi, -\pi)$ of velocity vector relative to the medial axis M
  2. Speed is $r/3$ so as to achieve stable performance in different shape scales.



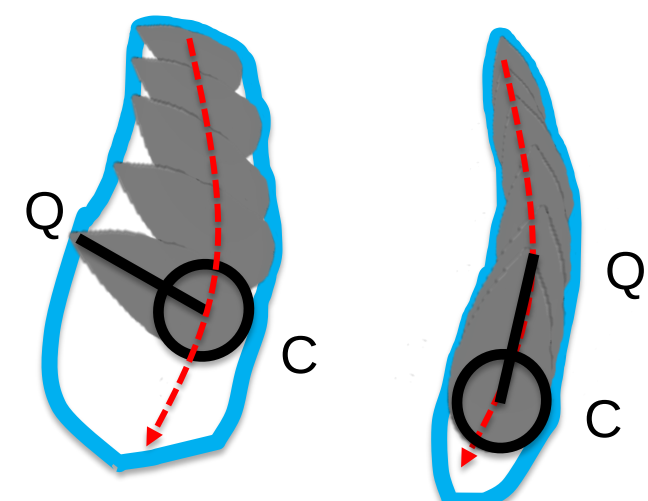
- **In order to reduce the uncertainty of multi-dimensional actions, only Action 1 is determined by the Gaussian policy function**

Action Design

- Action 2 and 3 are used to cover the local region after taking Action 1.



- **Setup**
  - Action 2 is defined as the direction of $\overrightarrow{CQ}$
  - Action 3 is defined as the length of $\overrightarrow{CQ}$

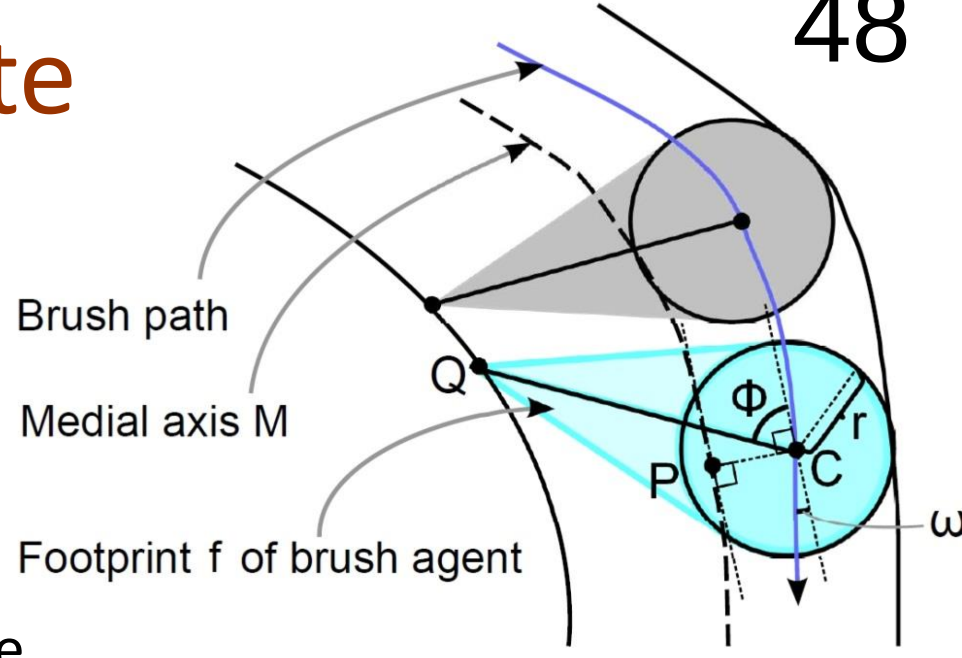


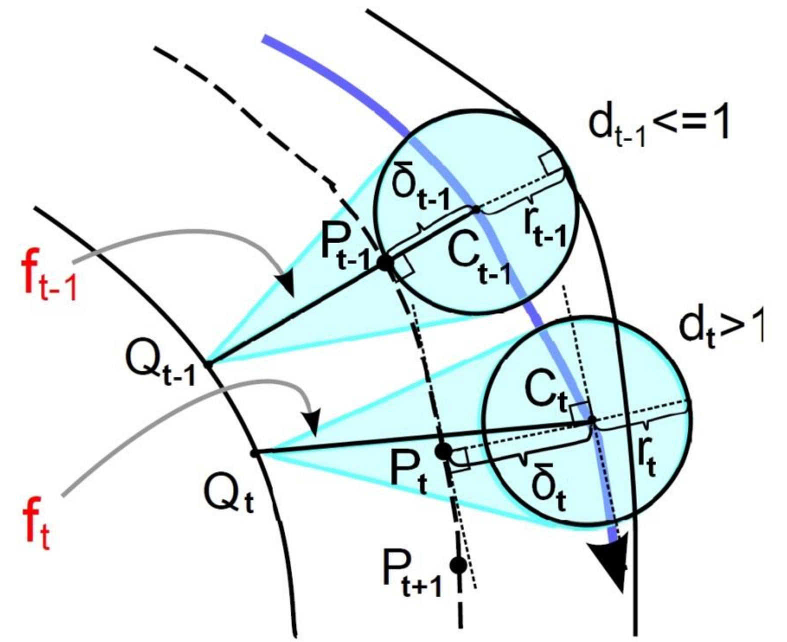
State Design

- *Global* and *local* measurement are regarded as a state in terms of an MDP
- Global measurement
  - Calculate the relative measurement and visualization
- Local measurement
  - Calculate a reward and a policy
  - **Purpose of using the local relative state:**
    - Avoid the influence of the coordinate transformation to the policy update so as to learn a drawing policy that can be generalized to new shapes

Local Relative State

- $\mathbf{s} = (\omega, \phi, d, \kappa_1, \kappa_2, f)^\top$
- Surrounding shape
  - ω : Relative angle of the velocity vector
  - ϕ : Relative heading direction
  - d : Ratio of the offset distance over radius of the bottom of the footprint
- Upcoming shape
  - κ_1 and κ_2 : Curvature of the medial axis
  - f : Flag that indicates whether the agent moves to the covered region (0) or not (1).





Design of Reward Function

- Smoother the brush stroke is, the higher the reward is

$$R(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}) = \begin{cases} 0 & \text{if } f_t = f_{t+1} \text{ or } l = 0, \\ \dfrac{1 + (|\kappa_1(t)| + |\kappa_2(t)|)/2}{\lambda_1 E_{\text{location}}^{(t)} + \lambda_2 E_{\text{posture}}^{(t)}} & \text{otherwise,} \end{cases}$$

$E_{\text{location}}^{(t)}$: measures the quality of the location of the brush agent with respect to the medial axis

$E_{\text{posture}}^{(t)}$: measures the quality of the posture of the brush agent based on the neighboring footprints

$$E_{\text{location}}^{(t)} = \begin{cases} \tau_1 |\omega_t| + \tau_2 (|d_t| + W) & d_t \in [-2, -1) \cup (1, 2], \\ \tau_1 |\omega_t| + \tau_2 |d_t| & d_t \in [-1, 1] \end{cases}$$

Design of Reward Function

- Smoother the brush stroke is, the higher the reward is

$$R(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}) = \begin{cases} 0 & \text{if } f_t = f_{t+1} \text{ or } l = 0, \\ \dfrac{1 + (|\kappa_1(t)| + |\kappa_2(t)|)/2}{\lambda_1 E^{(t)}_{\mathrm{location}} + \lambda_2 E^{(t)}_{\mathrm{posture}}} & \text{otherwise,} \end{cases}$$

$E^{(t)}_{\mathrm{location}}$: measures the quality of the location of the brush agent with respect to the medial axis

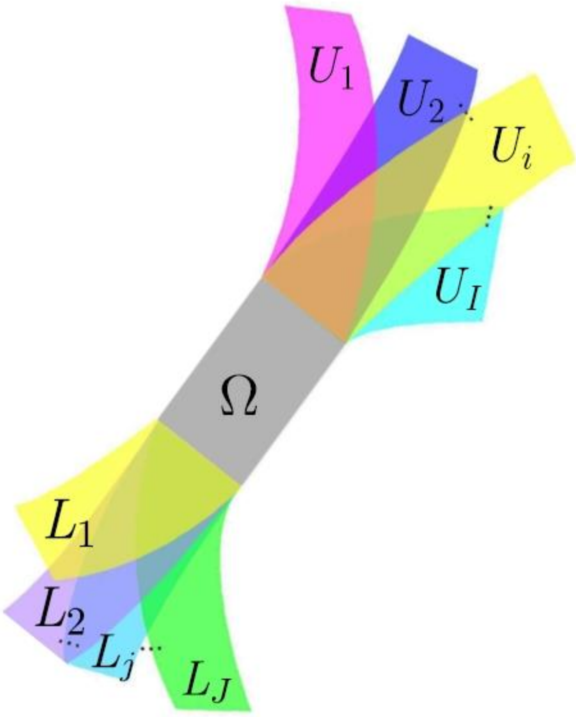
$E^{(t)}_{\mathrm{posture}}$: measures the quality of the posture of the brush agent based on the neighboring footprints

$$E^{(t)}_{\mathrm{posture}} = \zeta_1 \Delta\omega_t + \zeta_2 \Delta\phi_t + \zeta_3 \Delta d_t$$

$$\Delta x_t = \begin{cases} 1 & \text{if } x_t = x_{t-1} = 0, \\ \dfrac{(x_t - x_{t-1})^2}{(|x_t| + |x_{t-1}|)^2} & \text{otherwise.} \end{cases}$$

Efficient Training Scheme

- We propose to train the agent based on partial shapes instead of the entire shapes
- Advantages:
  1. Significantly increases the number and variation of training samples
  2. Enhances the policy to generalize the strokes in new shapes, since they may contains the partial shapes same with the training samplings even the entire shapes are different



Pipeline

- Introduction
- Our strategy
- Model in Reinforcement Learning Framework
- Design of Action, State, Reward and Training Scheme
- **Experiments**
- Conclusion

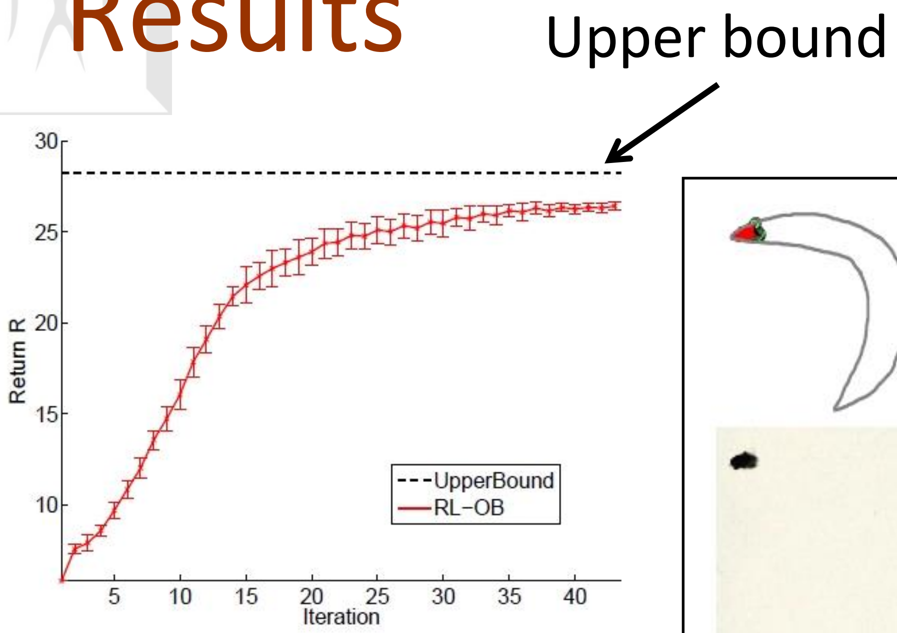
Experiment Setup

- Train the polices separately for
  - Upright brush style
  - Oblique style
- The length of episode: $T = 32$
- The number of episodes: $N = 300$
- The discount factor: $\gamma = 0.99$
- The learning rate: $\varepsilon = 0.1/\|\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}\|$
- Training data set of 80 classic strokes

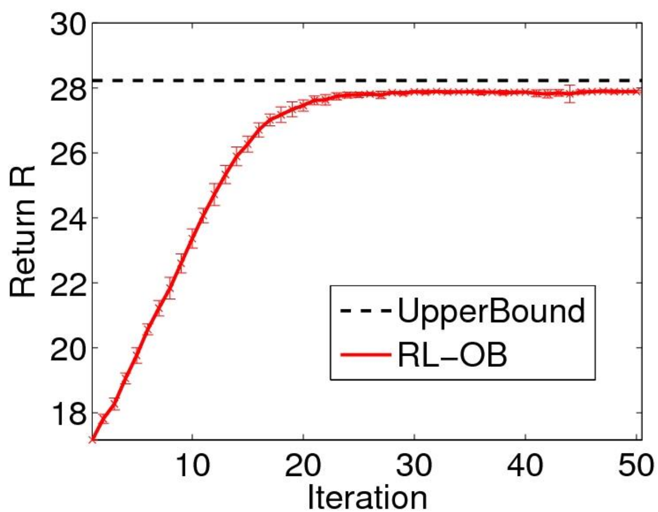
Training data:
80 real classical strokes

- The parameters of the reward function
  - For the oblique brush style: $\begin{cases} \lambda_1 = 0.5 \\ \lambda_2 = 0.5 \end{cases}$ $\begin{cases} \tau_1 = 0.5 \\ \tau_2 = 0.5 \end{cases}$ $\begin{cases} \zeta_1 = 1/3 \\ \zeta_2 = 1/3 \\ \zeta_3 = 1/3 \end{cases}$
  - For the upright brush style: $\begin{cases} \lambda_1 = 0.9 \\ \lambda_2 = 0.1 \end{cases}$ $\begin{cases} \tau_1 = 0.5 \\ \tau_2 = 0.5 \end{cases}$ $\begin{cases} \zeta_1 = 1/3 \\ \zeta_2 = 1/3 \\ \zeta_3 = 1/3 \end{cases}$

Results



Policy training for **oblique style**



Policy training for **upright style**

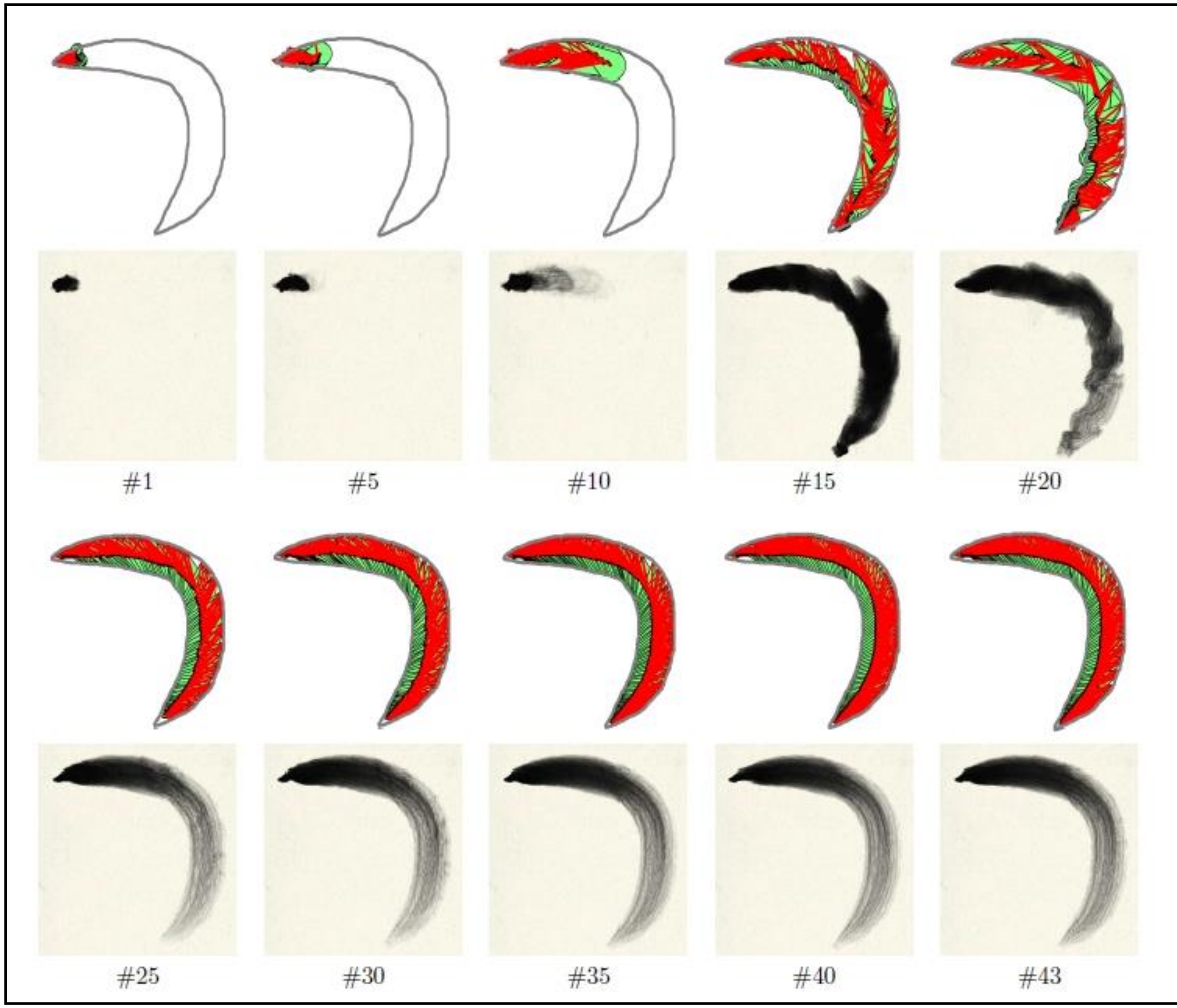


Illustration of training iterations

Comparison

- The DP method (Ning et. al., NPAR, 2010)
  - Fast $(3.95 * 10\text{s})$ but low return (-0.60)
  - High return (26.27) but too slow $(2.08 * 10^3\text{s})$
- Our policy gradient method
  - Fast $(4.00 * 10\text{s})$ and high return (26.44)
  - **50 times faster** than DP

METHOD	# CAND.	R	TIME(S)
DP	5	-0.60	$\mathbf{3.95 * 10^1}$
	10	-0.10	$1.01 * 10^2$
	20	6.54	$2.10 * 10^2$
	30	12.17	$3.25 * 10^2$
	40	20.03	$4.49 * 10^2$
	50	20.66	$5.73 * 10^2$
	60	22.35	$6.27 * 10^2$
	70	22.33	$7.48 * 10^2$
	80	24.42	$8.58 * 10^2$
	90	25.48	$9.74 * 10^2$
	100	25.08	$1.08 * 10^3$
	110	25.80	$1.19 * 10^3$
	120	25.22	$1.30 * 10^3$
	130	25.43	$1.40 * 10^3$
	140	26.01	$1.47 * 10^3$
	150	24.50	$1.68 * 10^3$
	160	25.49	$1.90 * 10^3$
	170	25.89	$2.03 * 10^3$
	180	**26.27**	$2.08 * 10^3$
	190	26.04	$2.30 * 10^3$
	200	24.11	$2.30 * 10^3$
RL	$\varnothing$	**26.44**	$\mathbf{4.00 * 10^1}$

Results

- Apply trained policy on new shapes

Summary

We applied reinforcement learning to oriental ink stroke generation

- Contributions:
- **CG:** A novel method for Stroke-based Painterly Rendering
- **RL:** A new application field of simulating the behaviors of artists

•To ensure the policy perform on arbitrary shapes in various scales,

1. Designed the states, actions and the reward functions of the brush agent to be **relative** to its local surroundings
2. Trained the agent with **partial** shapes of classical strokes

•The experiments demonstrated

1. Our RL method compared positively with an existing energy-based dynamic programming approach
2. The policies are able to generate smooth and natural strokes in arbitrary shapes

Learn Reward Function

[Xie et al., ICML2012]

In our previous work

- Trained the agent to draw strokes using reinforcement learning (RL) under *a manually pre-designed* reward function.

Reward function provides the most succinct and transferable definition of the task

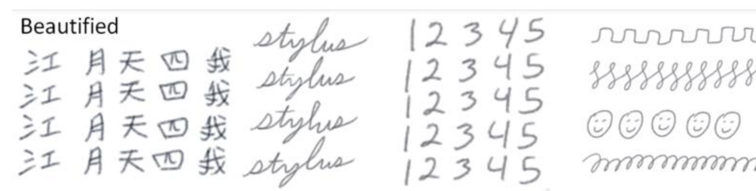
In this work

- Learn **the reward 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.

Personal Artistic Stylization

> Most of the studies
1. Used a stylus pen as input device
2. Produced line drawings in the style of a particular artist

HelpingHand: example- based stroke stylization [Lu et. al. 2012]



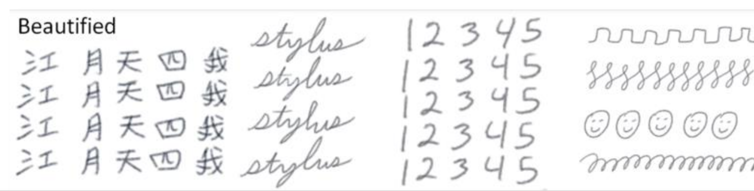
Handwriting Beautification Using Token Means [Zitnick. 2012]

Personal Artistic Stylization

> Most of the studies

1. Used a stylus pen as input device
2. Produced line drawings in the style of a particular artist

HelpingHand: example- based stroke stylization [Lu et. al. 2012]



Problem

However, they are not suitable to produce sumi-e brush strokes.

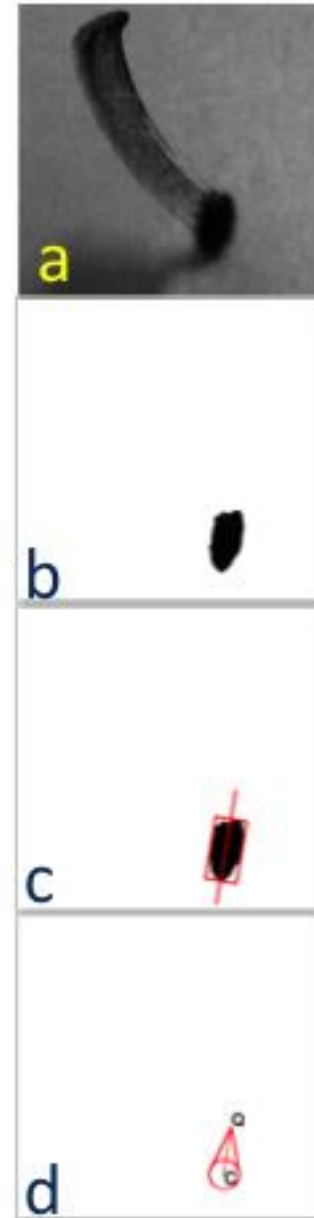
Device for Capture User's Real Strokes

- The device to measure the dynamics of drawing strokes
  - A digital single-lens reflex (DSLR) camera
  - Traditional Japanese calligraphy paper and ink

The reference list of eight basic stroke patterns

Data Processing

a. Split the recorded video of the stroke drawing
b. Detect the configuration of the footprint using the model-based tracking technique
c. Calculate the direction of the footprint
d. Match the template to detect the configuration of the footprint



Brush Agent Training by Reinforcement Learning

- Stroke generation is as a Markov Decision Process (MDP)
- **Target**

  Agent learns the optimal control policy to imitate the personal drawing style of a user

Design of Reward Function

Reward function

$$R(s_t, a_t, s_{t+1}) = \begin{cases} 0 & \text{if } f_t = f_{t+1} \text{ or } l = 0, \\ 1/C(s_t, a_t, s_{t+1}), & \text{otherwise} \end{cases}$$

Cost function

$$C(s_t, a_t, s_{t+1})$$

$$= \alpha_1 |\omega_{t+1}| + \alpha_2 |d_{t+1}| + \alpha_3 \Delta\omega_{t,t+1} + \alpha_4 \Delta\phi_{t,t+1} + \alpha_5 \Delta d_{t,t+1}$$

Cost of location + Cost of posture

$$\Delta x_{t+1} = \begin{cases} 1 & \text{if } x_t = x_{t+1} = 0, \\ \frac{(x_t - x_{t+1})^2}{(|x_t| + |x_{t+1}|)^2} & \text{otherwise.} \end{cases}$$

Design of Reward Function

Reward function

How to set the weights $\alpha_1, \alpha_2, \ldots, \alpha_5$ of multiple metric?

- We propose to use the maximum-margin Inverse Reinforcement Learning (IRL) method [Abbeel and Ng, 2004] to infer the appropriate values of the weights in the reward from user's real drawing data so as to learn the personal style.

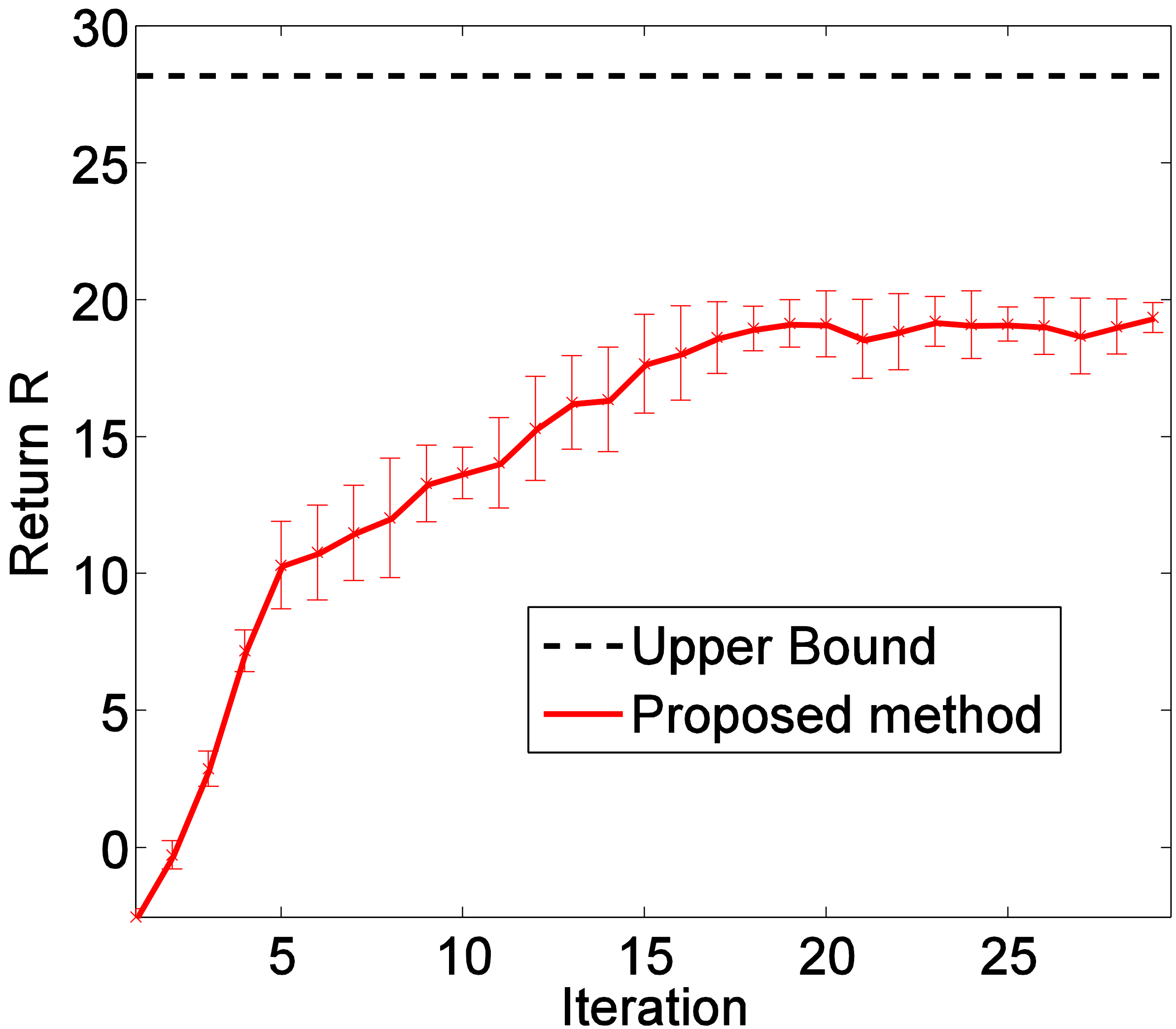
Cost of location + Cost of posture

$$\Delta x_{t+1} = \begin{cases} 1 & \text{if } x_t = x_{t+1} = 0, \\ \frac{(x_t - x_{t+1})^2}{(|x_t| + |x_{t+1}|)^2} & \text{otherwise.} \end{cases}$$

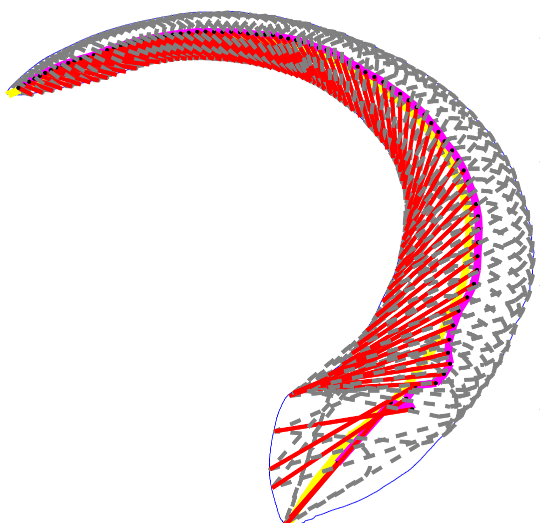
Experiment Setup

- The length of episode: $T = 32$
- The number of episodes: $N = 300$
- The discount factor: $\gamma = 0.99$
- The learning rate: $\varepsilon = 0.1/\|\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}\|$
- Training data set of 8 classic strokes styles

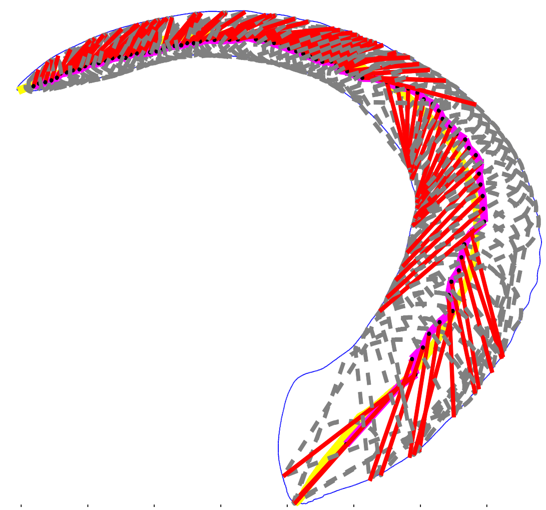
Averaged Return



Comparison of Brush Trajectory

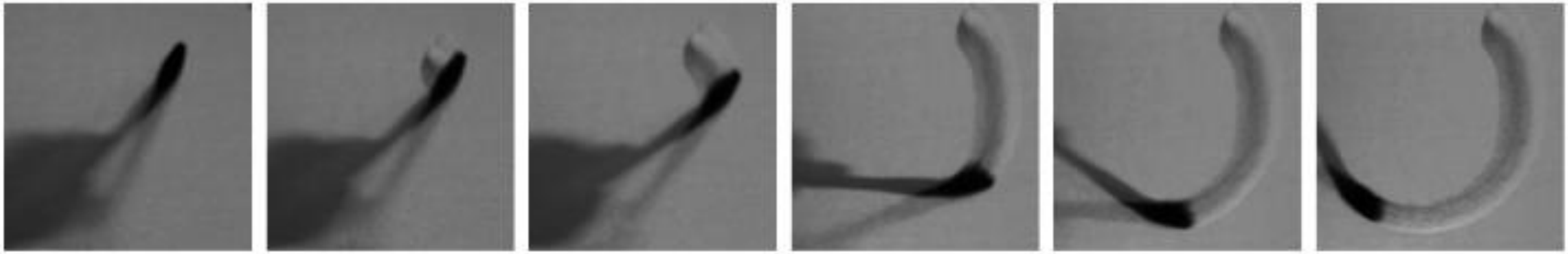


Trained with learned reward function

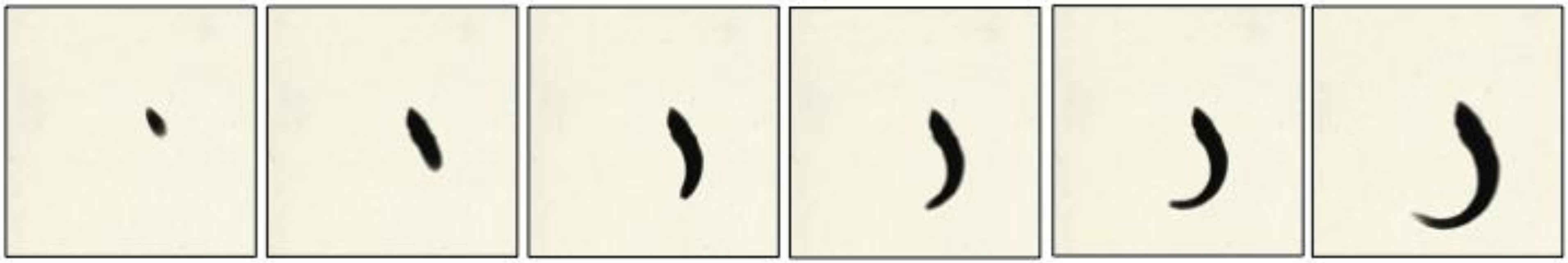


Trained with manually Designed Reward function [Xie at. al 2012]

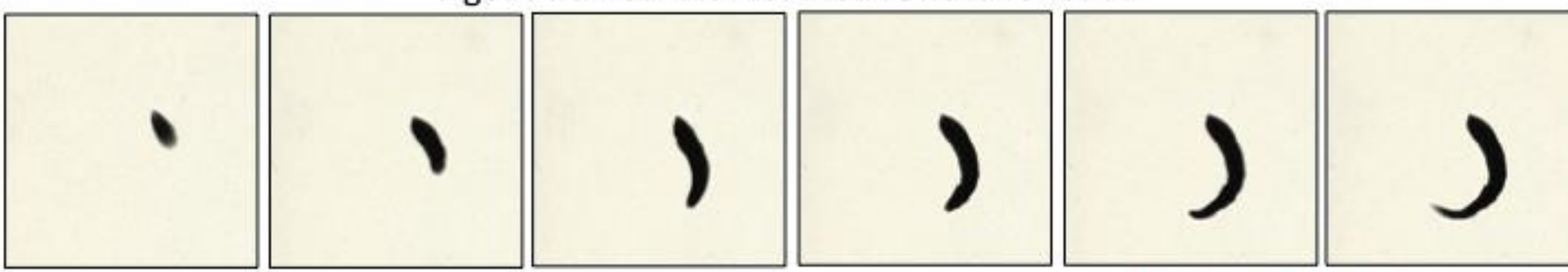
Comparison of Stroke-drawing Process



Real user

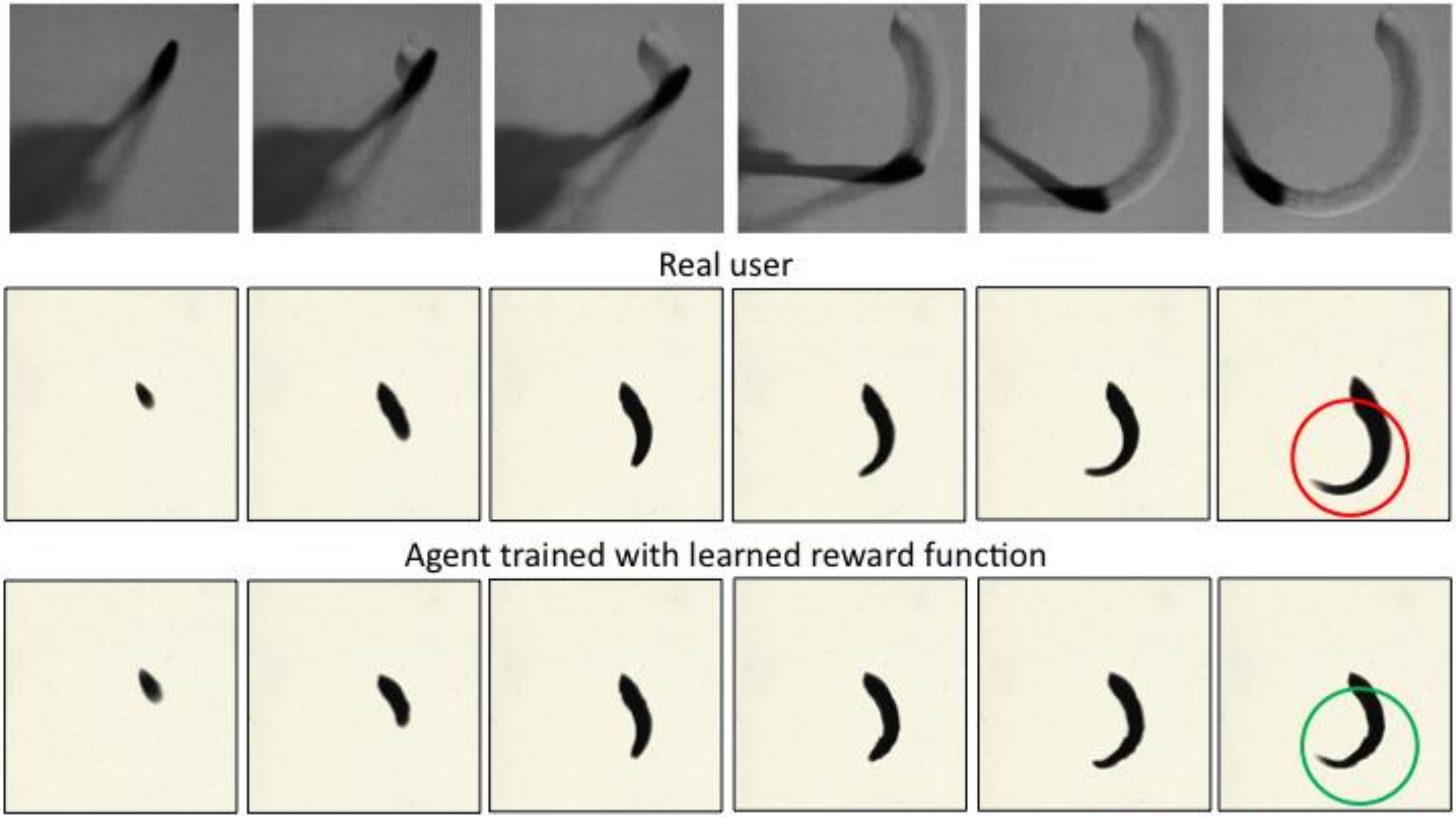


Agent trained with learned reward function



Agent trained with manually designed reward function [Xie et. al 2012]

Comparison of Stroke-drawing Process

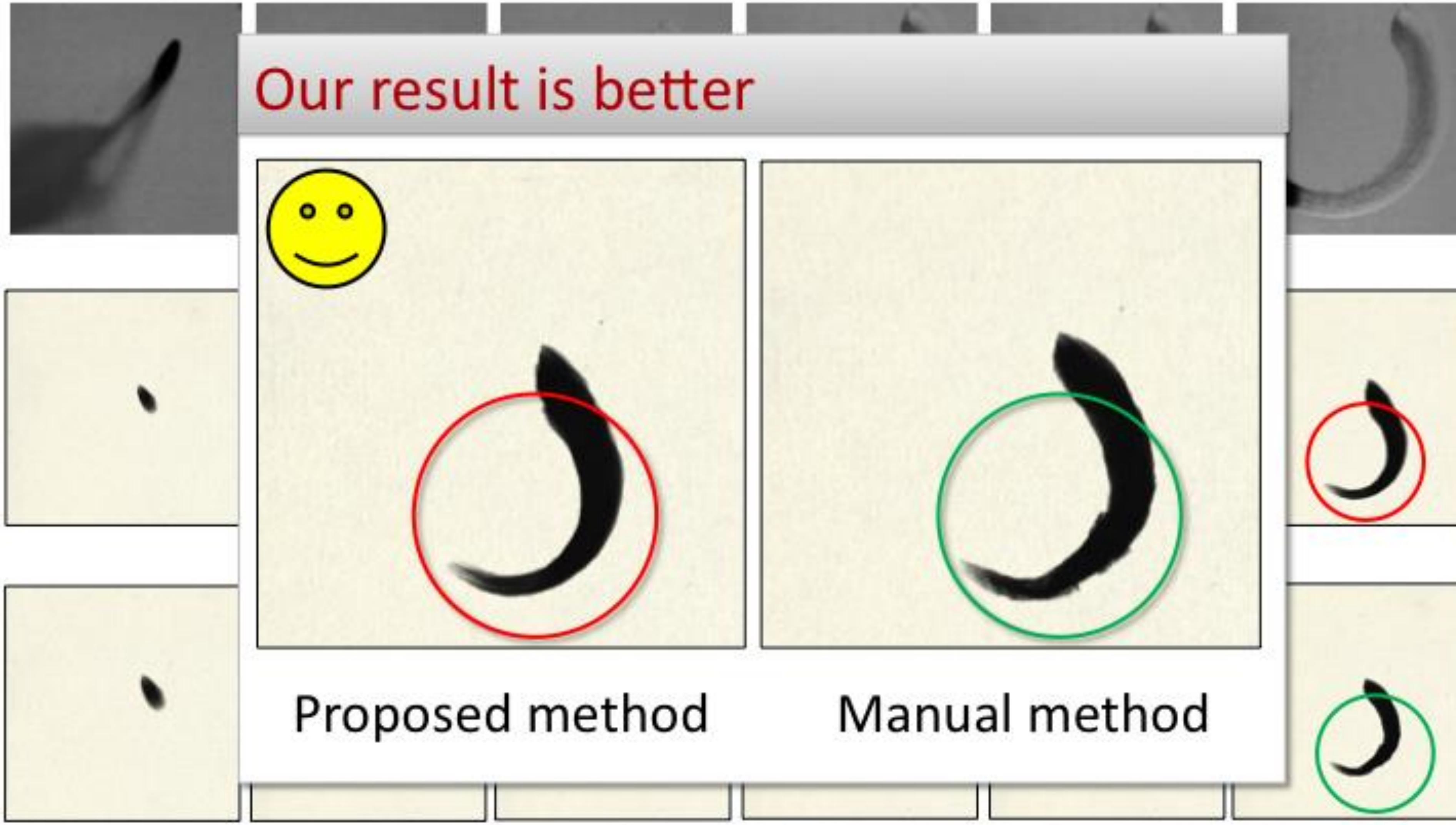


Real user

Agent trained with learned reward function

Agent trained with manually designed reward function [Xie et. al 2012]

Comparison of Stroke-drawing Process



Agent trained with manually designed reward function [Xie et. al 2012]

Results of Photo Conversion into Sumi-e Style

Summary

- Automatic sumi-e style drawings generation based on RL
- Our contributions
  1. The device for capturing real strokes
  2. The method to measure the brush configuration
  3. The reward function reconstruction from real users so as to learn the personal style

Homework

- Please write you opinion about this special lecture today.
  1. Send your report in the pdf format (within one week) to sugi@cs.titech.ac.jp or
  2. Directly submit the printed report to the lecturer next week.