creative commons
COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

이학석사학위논문

심층 강화학습 기반의 인공지능 화가 설계

Design of Artificial Drawing Artist Using Deep Reinforcement Learning

2022 년 8 월

서울대학교 대학원

협동과정 인지과학전공

이 강 훈

이학석사학위논문

심층 강화학습 기반의 인공지능 화가 설계

Design of Artificial Drawing Artist Using Deep Reinforcement Learning

2022 년 8 월

서울대학교 대학원

협동과정 인지과학전공

이 강 훈

심층 강화학습 기반의 인공지능 화가 설계

Design of Artificial Drawing Artist Using Deep Reinforcement Learning

지도교수 장 병 탁

이 논문을 이학석사 학위논문으로 제출함

2022 년 6 월

서울대학교 대학원

협동과정 인지과학전공

이 강 훈

이강훈의 이학석사 학위논문을 인준함

2022 년 8 월

위 원 장	김 현 진
부위원장	장 병 탁
위 원	권 가 진

초록

드로잉은 미적 감각과 심사숙고 능력을 동원한 인간 고유의 예술적 표현 방법이다. 그동안 컴퓨터 그래픽을 활용한 규칙기반의 여러 자동 드로잉 기법이 개발되었지만 다양한 드로잉 기법을 묘사하는 데에서 그 정교함과 예술성에 한계가 있었다. 그러나 최근에는 딥러닝 기술이 드로잉에 응용되면서 이전의 간극을 채우기 시작하였고, 예술 영역에서 인간과 인공지능의 경계가 빠르게 무너지고 있다. Stroke-based rendering (SBR)은 컴퓨터를 이용해서 스트로크라고 불리우는 붓터치와 같은 드로잉 요소의 위치나 색감 등을 고려해 가며 순차적으로 그림을 완성하는 문제이다. 본 학위논문에서는 주어진 목표 이미지에 대하여 자동적으로 SBR을 수행할 수 있도록 에이전트를 강화학습하는 내용을 다룬다. 그림의 표현 기법은 예술적 의도에 따라 다양하기 때문에, 각 기법에 강화학습이 적용되는 양태 또한 다양할 수 있다. 본 논문에서는 4가지의 드로잉 시나리오를 제시하고 각 시나리오에 맞춰 강화학습을 적용한다. 제시되는 SBR 시나리오는 크게 펜 드로잉과 고급 드로잉 기법으로 나누어진다. 펜 드로잉에서는 단색의 펜을 활용하여 그리는 기법을 다루며, 행동의 단순성을 이용해 실제 로봇을 이용한 드로잉으로 확장시켰다. 고급 드로잉 기법에서는 여러 스트로크에 대하여 일반화된 스트로크-조건부 페인팅 및 여러 재료를 모아 그림을 완성하는 콜라주 기법을 다룬다. 제시된 방법을 참고하면 드로잉과 관련한 인공지능 에이전트를 설계하고자 할 때 도움이 될 수 있다. 이러한 기술은 드로잉 소프트웨어, 영화, 애니메이션, 장난감, 공연 및 전시 등의 다양한 예술 산업에 활용될 수 있다.

주요어: 강화학습, SBR, 스트로크, 펜 드로잉, 페인팅, 콜라주

학번: 2020-27294

목차

그림 목차

표 목차

제 1 장 서론

Stroke-based rendering (이하 SBR)은 빈 도화지로부터 시작해 한 단계씩 스트로크(Stroke)의 위치, 모양, 색 등을 순차적으로 결정 및 배치해가면서 사용자로부터 주어진 목표 이미지를 그려내는 작업이다 [5]. 스트로크는 점, 선, 도형, 붓터치 등 아무 형태나 가능한 드로잉의 최소 단위로, 충분한 양의 스트로크가 배치되면 한 폭의 그림을 완성할 수 있다. 그림의 스타일은 스트로크의 종류에 따라 다르게 표현된다. 가령 점 스트로크를 사용하여 그림을 그릴 경우 오밀조밀하게 표현되는 점묘화 기법의 그림이 된다. 검은색 선 스트로크를 사용한다면 펜으로 그려진 낙서를 표현하거나 소묘와 같은 표현이 가능하다. 색상 변경이 가능한 붓터치 혹은 다각형처럼 보다 자유로운 스트로크를 사용한다면 페인팅이나 픽셀 아트 등의 더욱 다채로운 그림을 표현할 수 있다.

사람이 그림을 그리듯, SBR은 매 순간마다 스트로크를 배치하면서 도화지를 점점 완성된 그림에 가깝게 만든다. 이때 스트로크의 배치는 매 순간의 도화지 상태와 미래의 도화지 상태, 그리고 목표하는 그림을 고려해 심사숙고하여 결정되어야 한다. 따라서 SBR은 고도의 의사결정 능력을 요구하는 방법이다. 더욱이 스트로의 형식이나 목표하는 그림의 성격에 따라 적절한 스트로크의 배치 전략이 상이하기 때문에 단순 규칙만으로는 SBR을 자동적으로 수행하는 에이전트를 설계하기 어렵다. 이러한 특성 때문에 직접 문제에 걸맞는 그리기 규칙을 설계하기보다는 학습 방법을 도입하여 에이전트가 스스로 규칙을 형성할 수 있도록 하는 것이 매력적이다. 특히 강화학습은 순차적으로 행동을 결정하는 문제를 다루므로 SBR 학습에 적합하다. 본 논문에서는 페인팅 작업에 강화학습이 응용된 사례를 종합하여 일반화된 알고리즘을 먼저 제시한다. 이후 해당 알고리즘을 기반으로 문

제를 확장하여 펜 드로잉과 로봇 드로잉, 그리고 스트로크-조건부 페인팅, 콜라주 기법의 고급 드로잉 기법을 해결하는 방법과 그에 대한 결과를 제시한다.

논문의 구성은 다음과 같다. **2장 배경연구**에서는 SBR 문제를 해결하기 위해 시도되었던 고전적 접근 방법과 딥러닝이 도입된 방법을 간략히 소개 및 비교한 후 이후 제시할 방법의 이론적 기초가 되는 강화학습을 소개한다. **3장 방법**에서는 강화학습을 이용한 SBR 방법을 일반화하여 제시한다. **4장** 및 **5장**에서는 3장에서 제시한 알고리즘이 다양한 드로잉 분야에 적용될 수 있도록 세분화된 방법론을 각각 제시하고 실험 결과를 나타낸다. **6장 결론 및 논의**에서는 방법론과 실험 결과를 종합하고 의의와 한계 및 향후 연구 방향에 대하여 논의한다.

제 2 장 배경연구

2.1 고전적 방법

자동으로 SBR을 수행하는 에이전트를 만들기 위해 고전적으로는 주로 규칙 기반의 방법들이 사용되었다 [5]. 고전적 방법은 스트로크를 다루는 방법에 따라 크게 탐욕적 방법과 최적화 방법으로 나눌 수 있다. 탐욕적 방법에서는 매 순간 다음 스트로크의 위치와 모양 등이 휴리스틱으로 정의된 방식에 따라 계산되어 결정되고, 한 번 놓고 난 스트로크는 변경되지 않는다. 적절한 스트로크를 계산하는 내부 루프와 스트로크를 순차적으로 놓아 나가는 외부 루프를 거쳐 그림은 점차 구체화된다. 반면 최적화 방법에서는 스트로크가 이미 빼곡하게 놓여진 도화지에서 출발하여 휴리스틱으로 정의된 에너지 함수를 최소화하는 방향으로 각 스트로크들을 변경해 나간다. SBR은 목표하는 이미지를 재건하는 작업이므로, 이미지 스타일 변경에 고전적 방법들이 유용하게 사용되어 왔다. 탐욕적 방법은 페인팅을 하는 과정과 유사하기 때문에 사진을 페인팅 스타일로 변경하는데 적합하고, 최적화 방법은 점묘법이나 타일 기법 등의 수학적으로 계산되기 수월한 스타일로 변경하는데 적합하다. 그림 2.1은 휴리스틱을 사용하여 수행된 SBR의 예시이다.

2.2 딥러닝을 이용한 방법

고전적 방법은 명확한 규칙에 기반하므로 동작 과정의 파악과 알고리즘의 변경이 손쉽다는 장점이 있다. 하지만 그러한 규칙을 정의하는 것은 까다로운 과정이다. 특히 목표하는 그림 도메인에는 특화되어 있는 동시에 도메인 내에서는 일반화된 형태의 적절한 규칙들을 일일이 찾아내는 것은 어려운 일이며, 그 규칙은 휴리스

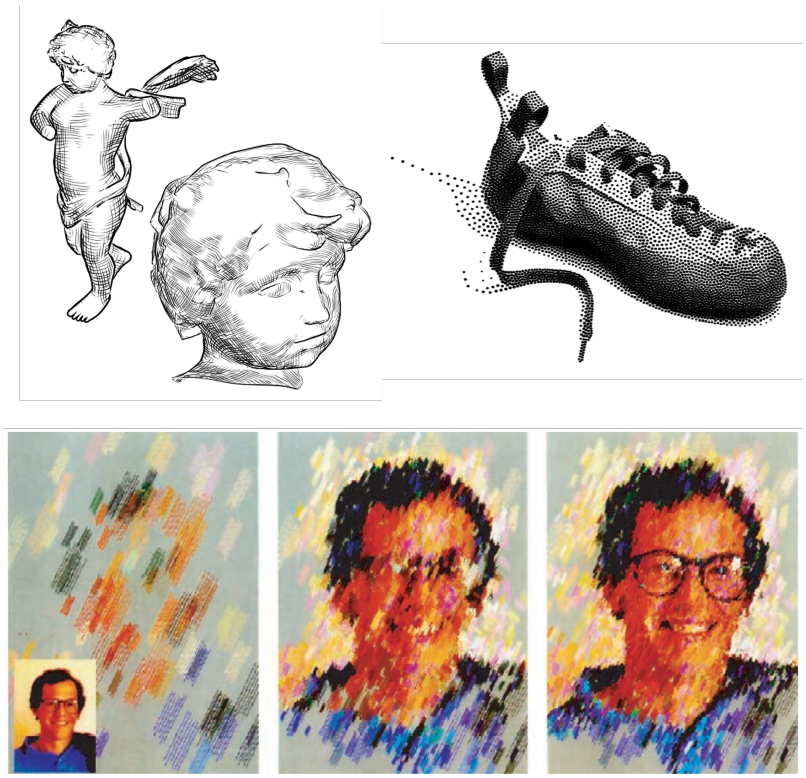


그림 2.1: 휴리스틱을 사용하여 그려진 그림들 [1, 2, 3]

틱에 기반하므로 최적에 가깝다고 보장할 수 없다. 그러나 이미지 처리 분야에서 딥러닝이 비약적으로 발전하면서 규칙에 기반한 고전적 방법을 학습에 기반한 방법으로 대체할 여지가 생겼다. 에이전트가 드로잉 규칙을 문제에 알맞게 스스로 발견하고 발전시킬 수 있다면 정의하기 어려운 규칙을 직접 만들어줄 필요가 없다. 또한, 잘 정의된 학습 방법은 수렴성을 보장하기 때문에 최적에 가깝게 학습된 규칙이라는 설명이 가능해진다.

SBR의 수행을 목표로, 또는 그와 관련하여 다양한 딥러닝 방법이 시도되었다. SketchRNN [6]은 지도학습을 이용한 사례로, 사람이 마우스 펜으로 그림을 그리는 과정을 수집한 데이터셋을 활용하여 순환신경망을 학습하였다. 해당 신경망은

하나의 제시어에 대하여 드로잉할 수 있도록 학습되는데, 가령 자동차 그림 과정 데이터로 학습된 신경망은 자동차 펜 그림을 다양한 형태로 그릴수 있다. 그러나 드로잉 과정에 대한 데이터를 다량으로 얻는 것은 비용이 많이 든다. 이때 강화학습을 활용하면 드로잉 과정 데이터 없이 목표 이미지 데이터만으로도 드로잉 에이전트를 학습할 수 있다. [7, 8, 4]에서는 SBR 문제를 강화학습에서 요구하는 문제의 형태로 재정의하여 에이전트가 시행착오를 통해 스스로 그리기 규칙을 형성해나갈 수 있도록 하였다. 최근에는 트랜스포머를 적용한 방법 [9]이나 그리는 과정을 모두 미분가능하게 하여 경사하강법을 통해 목표 이미지에 대해 최적화하는 방법 [10]이 시도되었다. 강화학습에 비해 트랜스포머를 사용한 방법은 학습 및 추론 시간이 적고, 최적화 방법은 긴 추론시간 대신 학습이 필요하지 않다는 장점이 있다. 그러나 [9]에서 에이전트가 동일조건에서 그린 최종 그림의 품질은 강화학습을 사용한 방법에서 가장 좋다는 것이 입증되었다.

2.3 강화학습

강화학습은 순차적으로 결정되어야 하는 문제에 적합한 학습 방법이다. 강화학습은 Markov Decision Process (MDP)로 주어지는 튜플 $M \equiv (\mathcal{S}, \mathcal{A}, p, r, \gamma)$로 환경을 정의한다. $\mathcal{S}$와 $\mathcal{A}$는 각각 상태공간과 행동공간이다. 매 순간 t에서 에이전트는 상태 $s \in \mathcal{S}$에 처하고 행동 $a \in \mathcal{A}$을 결정한다. 다음 상태 s'는 상태전이함수 $p(\cdot|s,a)$에 의해 결정된다. 에이전트는 매 순간 전이에 따른 보상 $r(s,a,s')$을 받는다. 에이전트의 목적은 각 순간 t에서의 행동과 상태에 따라 미래에 받게 될 보상의 총합을 나타내는 행동가치함수 $Q_r^\pi \equiv \mathbb{E}^\pi \left[\sum_{i=0}^{\infty} \gamma^i r\left(S_{t+i}, A_{t+i}, S_{t+i+1}\right) | S_t = s, A_t = a\right]$를 최대화하도록 행동을 결정하는 최적 정책 $\pi^*(a_t|s_t)$를 찾는 것이다. 단, 행동가치함수는 보상의 총합이 발산하지 않도록 매 순간마다 감가율 $\gamma \in [0,1)$이 복리적으로 곱해진 형태이다.

상술한 내용을 바탕으로 다양한 종류의 강화학습 알고리즘이 개발되었다. 강

화학습 알고리즘은 일반적으로 행동의 종류에 따라 이산행동을 위한 알고리즘과 연속행동을 위한 알고리즘의 두 종류로 나눌 수 있다. 이산행동은 키보드 자판의 q, w, e, r 과 같이 각 행동 간의 관계가 독립적인 경우를 말하고, 연속행동은 20km/h, 30km/h, 40km/h 와 같이 각 행동간의 관계가 연속적인 경우를 말한다. SBR 작업의 경우 스트로크의 위치나 형태, 색 등을 결정해야 하는데, 각 요소에서 결정될 수 있는 행동들은 연속적인 관계를 가지므로 SBR은 연속행동의 문제로 보는 것이 적절하다.

연속행동을 위한 대표적인 강화학습 알고리즘으로는 행동 결정 성격이 다른 두 알고리즘 Deep Deterministic Policy Gradient (DDPG) [11]와 Soft Actor-Critic (SAC)을 꼽을 수 있다. DDPG는 신경망의 출력이 곧바로 행동을 결정하는 결정적 정책을 가진다. 그러나 정책이 완전히 결정적이면 현행 정책에서 더 나은 정책을 발견하기 어렵다. 따라서 DDPG에서는 행동 출력에 잡음을 추가하여 에이전트가 학습 중에 다양한 경험을 얻을 수 있도록 한다. 학습 중이 아닐 때에는 잡음 없이 정책이 출력된 행동을 곧바로 이용한다. 반면 SAC에서는 확률적으로 행동을 결정하는 확률적 정책을 가진다. SAC에서는 신경망이 곧바로 결정된 행동을 출력하지 않고, 행동 확률분포에 대한 모수를 출력한다. 학습 중에는 해당 확률분포로부터 샘플링하여 행동을 결정하고, 학습 중이 아닐 때에는 모수 중 평균값을 행동으로 결정한다. 추가로, SAC에서는 기본적인 행동가치함수 식의 보상에 엔트로피를 더해준 형태인 Soft Q 함수

$$Q^*_{soft}(s_t, a_t) = r_t + \mathbb{E}_{(s_{t+1},\ldots)\sim\rho_\pi}\left[\sum_{l=1}^{\infty}\gamma^l(r_{t+1} + \alpha\mathcal{H}(\pi^*_{\text{MaxEnt}}(\cdot|s_{t+1})))\right] \quad (2.1)$$

을 채택하여 학습하는 정책의 행동 선택 다양성을 향상시킨다 [12, 13]. 이때 π^*_{MaxEnt}는 Soft Q 함수를 최대화하는 최적 정책이다. 본 논문에서는 SAC를 주요 강화학습 알고리즘으로 활용한다.

제 3 장 강화학습 기반 드로잉

3.1 드로잉 환경

2.3절에서 언급한 강화학습에서 요구하는 MDP 형식을 만족하기 위해 최소한의 드로잉 환경이 구축되어야 한다. SBR은 그림이 그려질 도화지와 참고할 목표 이미지가 필요하다. 여기서 도화지를 V, 목표 이미지를 G로 정의한다. 목표 이미지는 시간과 관계없이 항상 일정하다. 하지만 도화지는 매 순간 스트로크가 놓여짐에 따라 변화한다. 따라서 도화지는 시간에 따라 변화하는 값으로 순간 t에서의 도화지를 V_t로 나타낼 수 있다. 마찬가지로 스트로크를 A라 한다면 순간 t에서의 스트로크를 A_t로 나타낼 수 있다. 도화지의 변이는 같은 순간의 스트로크가 도화지에 놓여지면서 이루어진다. 도화지 변이함수를 $draw$라 한다면 도화지의 변이는 $V_{t+1} = draw(V_t, A_t)$와 같이 나타낼 수 있다. 이러한 과정은 렌더링이라고도 한다. 이미지 렌더링은 이산적인 픽셀 공간에서의 연산으로 일반적으로 미분 불가능하다. 하지만 [4, 10, 9]에서와 같이 신경망을 이용해 렌더링 기능을 근사하면 미분가능한 렌더링을 구현할 수 있다. 만약 렌더링이 미분가능하다면 종단 학습의 구성요소로 활용될 수 있고, gpu 가속을 통해 연산을 병렬화하여 학습 속도를 증가시킬 수 있다.

3.2 상태 및 행동

SBR을 위한 작업에서 행동을 결정하기 위해 에이전트가 순간 t에서 필수적으로 관찰하는 정보는 V_t, G이다. 문제에 따라 추가로 관찰하는 정보를 α_t라 하면 순간 t에서의 상태는 $s_t = (V_t, G, \alpha_t)$로 정의된다. 에이전트는 초기 도화지 V_0부터 시작하여 매 순간 상태 s_t를 관찰하고 행동 a_t를 결정한다. a_t는 스트로크

이미지 A_t를 결정할 일종의 파라미터 집합으로 스트로크 이미지가 배치될 위치나 크기, 색상 등과 같은 요소를 포함할 수 있다. 가령 [7, 4]에서는 Bézier curve라고 불리우는 곡선형 스트로크를 결정하는 파라미터를 행동에 포함한다. 행동 a_t를 스트로크 이미지 A_t로 만드는 함수를 $render$라 하면 스트로크 이미지는 $A_t = render(a_t)$와 같이 나타낼 수 있다. 정리하면, 결정된 행동에 따라 다음 순간에서의 도화지는 $V_{t+1} = draw(V_t, render(a_t))$와 같이 결정되므로 다음 상태는 $s_{t+1} = (draw(V_t, render(a_t)), G, p_\alpha(\alpha_t))$로 나타낼 수 있으며 이때 p_α는 V와 G를 제외한 상태요소에 대한 전이함수로 문제에 따라 다르게 정의될 수 있다.

3.3 보상

강화학습에서 보상은 에이전트의 유일한 학습 신호이므로 적절한 보상을 정의하는 것이 성공적인 학습을 좌우한다. 최종 목표는 에이전트가 도화지 V에 변화를 가하여 목표 이미지 G와 최대한 유사하게 만드는 것이므로, 이미지 간의 거리를 나타내는 어떠한 함수 d가 있다면 V와 G의 거리는 $d(V, G)$로 표현할 수 있고 이를 최소화하는 것이 에이전트의 적절한 학습 방향이다. 따라서 보상 설계시 거리 d를 고려하는 것이 직관적이다. [7]에서는 완성된 그림과 목표 이미지와의 거리를 계산하여 보상으로 사용한다. 그러나 최종 순간에서만 주어지는 희소 보상 환경보다는 매 순간 보상이 주어지는 밀집 보상 환경이 하나의 에피소드가 제공하는 정보량이 더 많기 때문에, 각자 약간의 장단점은 있지만 강화학습은 일반적으로 밀집 보상 환경에서 더 잘 학습한다. [4, 8]에서는 밀집 보상을 사용하는데, 매 순간마다 이전 도화지 그림과 목표 이미지 사이의 거리와, 현재 도화지 그림과 목표 이미지 사이의 거리를 계산하여 이전 순간으로부터 개선된 정도 만큼의 보상을 부여한다. 가령 보상 $r_t = d(V_t, G) - d(V_{t+1}, G)$와 같은 식이다.

이미지 간의 거리 d를 나타내기 위한 척도로는 다양한 방식을 사용할 수 있다.

가장 단순한 척도는 l_2 거리로 참고적인 수준에서 자주 사용된다.

$$l_2(\mathbf{x}, \mathbf{y}) = \frac{1}{K}\sum_{i=1}^{K}(\mathbf{x}_i - \mathbf{y}_i)^2 \tag{3.1}$$

$\mathbf{x}$, $\mathbf{y}$는 거리를 측정할 이미지 벡터이다. l_2거리는 어떠한 벡터이든 객관적으로 적용된다는 장점이 있지만, 이미지 벡터에는 편향이 존재하기 때문에 사람의 눈이 느끼는 픽셀 수준 이상의 시각적 차이를 나타내기에 아주 적절한 방법은 아니다. 이에 [7, 4, 8]에서는 적대적 생성망 [14] 학습시 사용되는 판별자 신경망을 도입하여 거리(또는 유사도) 척도로 활용한다. 판별자 신경망을 D, 에이전트가 최종적으로 그린 그림을 V_T라고 하면 판별자 신경망은 (V_T, G) 쌍에 대해서는 위조로, (G, G) 쌍에 대해서는 진품으로 판별하도록 학습된다. 판별자 학습을 위한 알고리즘으로 [7, 4]에서는 WGAN [15]을 사용하지만, [8]에서는 SNGAN [16]을 사용하는 것이 WGAN보다 더 좋은 척도임을 실험적으로 입증하였다.

3.4 일반화

3.1, 3.2, 3.3절을 종합하면 SBR을 위한 강화학습 방법을 알고리즘 1과 같이 일반화할 수 있다. 환경적으로는 스트로크 이미지A_t를 구성하기 위해 어떠한 파라미터를 행동 a_t에 포함할 지와 그에 대한 렌더링 방식 $render$가 정의되어야 한다. 해당 정의 방식은 드로잉의 기초 단위로서 완성될 그림의 스타일을 결정하는 중요 요소이다. 스트로크 이미지가 도화지에 놓여지는 $draw$의 구현 방법은 다양할 수 있으나 어떤 문제이든 동일한 기능을 수행한다.

에이전트의 학습을 설계할 때는 먼저 어떤 강화학습 알고리즘을 사용할 지 선택한다. 강화학습 알고리즘에 따라 행동을 선택하는 방식도 다양해진다. 문제에 따라 에이전트가 현재 도화지와 목표 이미지 외에 관찰해야 하는 새로운 요소가 있을 수 있고, 이는 모두 상태 s_t에 포함된다. 보상 신호는 밀집 보상의 형식을 따라 목표 이미지와 도화지 간의 거리를 바탕으로 문제에 맞게 정의한다.

Algorithm 1 Reinforcement Learning Framework for SBR

Initialize agent π

Define reinforcement learning algorithm RL

for episode=1,2,...,N **do**

    Initialize canvas V_0

    Get goal image G

    for t=0,1,...,T **do**

        Observe $s_t = (V_t, G, \alpha_t)$ for agent π

        Take action a_t from agent π

        Render stroke $A_t = render(a_t)$

        Make next canvas $V_{t+1} = draw(A_t)$

        Apply state transition $s_{t+1} = p(s_t, a_t) = (V_{t+1}, G, p_\alpha(\alpha_t))$

        Calculate reward from distances $d(V_t, G)$, $d(V_{t+1}, G)$

    end for

    Step RL for agent π

end for

제 4 장 펜 드로잉으로의 응용

본 장에서는 3.4절에서 제시한 SBR을 위해 일반화된 강화학습 방법을 바탕으로 다양한 문제에 맞게 응용 및 확장된 방법론을 제안하고 그에 대한 적용 예시를 보인다.

4.1 펜 드로잉

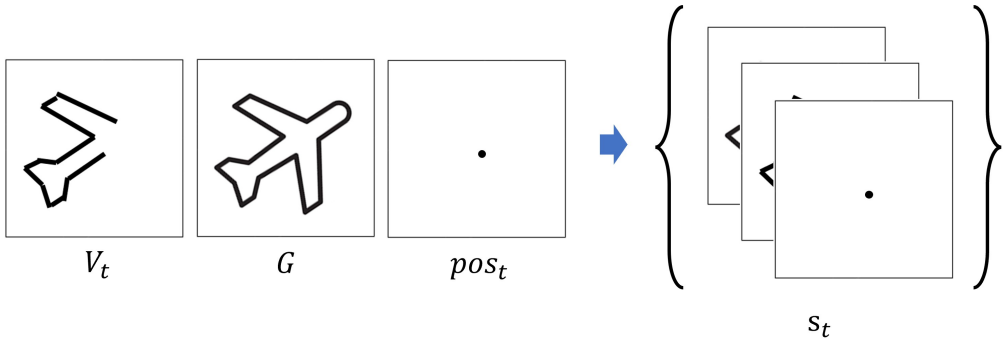
4.1.1 문제

펜 드로잉은 펜이나 연필과 같은 선분 표현에 용이한 도구를 이용하여 그림을 그리는 방식이다. 본 절에서 정의하는 펜 드로잉에서는 일정 굵기의 검은색 펜만을 이용하고, 펜을 뗄 수 없이 이어그려야 하는 특수한 조건을 부여한다.

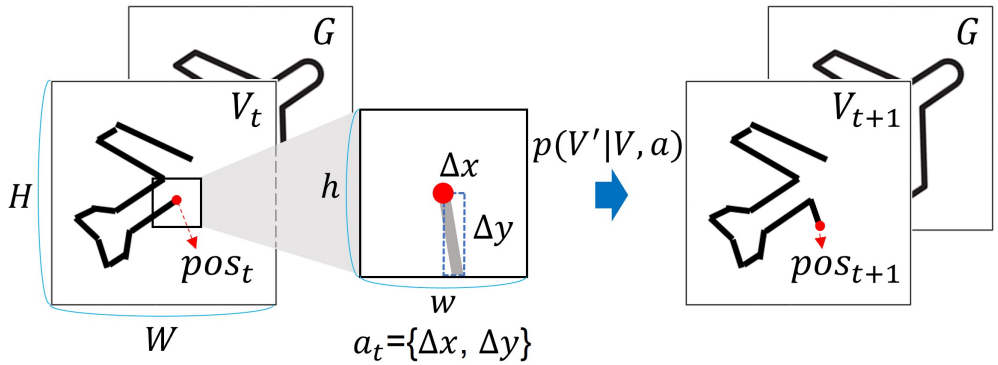
4.1.2 방법

펜 드로잉 문제에 필요한 정보를 표현하기 위해 상태 s_t를 그림 4.1(a)와 같이 도화지 V_t, 목표 이미지 G 및 현재 펜촉의 위치를 나타낸 이미지 pos_t로 구성하였다. 행동은 그림 4.1(b)와 같이 현재 펜촉의 위치를 얼마만큼 이동할 것인지에 대한 2차원 좌표 변화량 정보 $a_t = (\Delta x, \Delta y)$로 나타내었다. 이때 펜촉이 너무 멀리 이동할 수 없도록 길이 h의 정사각형 구간 내에서만 펜촉을 이동할 수 있도록 제한하였다. 본 논문에서는 $h = 20$로, 도화지의 크기는 $W = 84$, $H = 84$로 설정하였다. $render$는 현재 펜촉의 위치에서 시작하여 행동이 나타내는 위치 변화량이 적용된 좌표에서 끝나는 선분을 스트로크 이미지로 표현하도록 구성하였다.

강화학습 알고리즘은 SAC를 사용하였다. 학습을 위해 제공되는 목표 이미지 G로는 *Quick, Draw!* [6] 데이터셋(그림 4.2) 중 34,500장의 이미지를 가져온 후



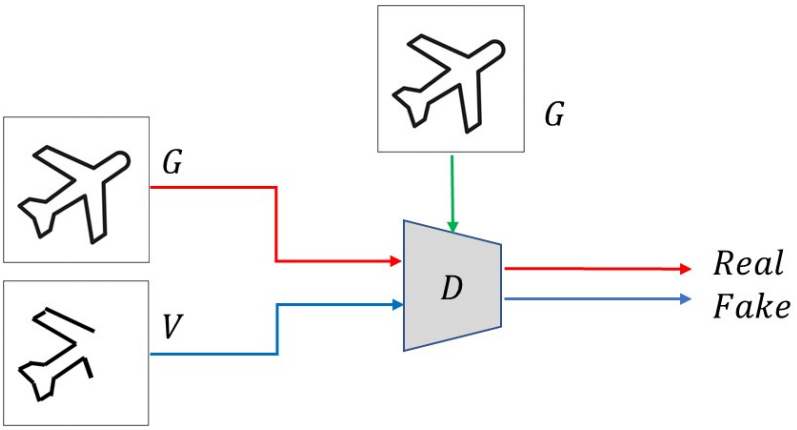
(a) 펜 드로잉을 위한 상태정의



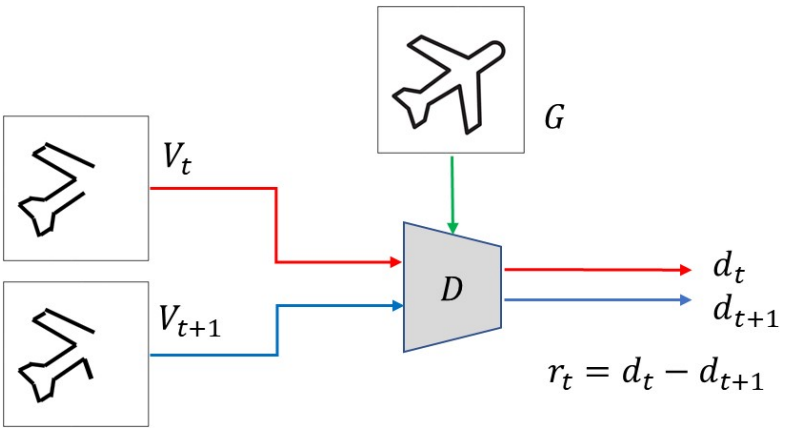
(b) 펜 드로잉을 위한 행동 및 상태전이

그림 4.1: 펜 드로잉을 위한 환경 구성

그림 4.2: *Quick, Draw!* 데이터셋



(a) 판별자 신경망의 학습



(b) 판별자 신경망을 이용한 보상의 계산

그림 4.3: 펜 드로잉을 위한 보상함수

균일 샘플링하여 활용하였다. 보상을 제공하기 위해 SNGAN의 판별자 신경망을 사용하였다. 판별자 신경망 D의 학습은 그림 4.3(a)와 같이 (G,G)쌍에 대해서는 참, (V,G)쌍에 대해서는 위조로 판별하도록 하여 도화지와 목표 이미지 간의 거리 d를 나타낼 수 있도록 학습하였다. 이렇게 학습된 판별자 신경망을 이용하여 그림 4.3(b)와 같이 목표 이미지와 도화지 간의 거리가 줄어든 정도를 보상 $r_t = d_t - d_{t+1}$로 나타내었다. 판별자 신경망의 학습은 에이전트의 강화학습과 동시에 온라인으로 진행하도록 하였다.

4.1.3 실험 및 결과

드로잉 에이전트의 수행 결과를 살펴보기 위해 하나의 드로잉에 대해 거치는 행동 수 $T = 50$으로, 보상 할인율 $\gamma = 0.95$로 설정하고 32개의 병렬 환경에서 40,000회 학습하였다. 그림 4.4는 학습된 에이전트에 대한 실험 결과를 나타낸 그림이다. 그림 4.4(a)는 호박이 표현된 한 목표 이미지에 대해 에이전트가 드로잉을 수행하는 모습을 시간 순서대로 나타낸 것이다. 학습된 에이전트는 시작점으로부터 펜촉을 천천히 이동시키며 성공적으로 목표 이미지를 그려내었다. 선분이 교차하는 지점에서는 어느 쪽으로 펜촉을 옮겨도 그림을 완성하는 것이 가능한데, 그러한 순간마다 에이전트는 학습 경험을 통해 형성한 자신만의 드로잉 기법을 토대로 보간 없이 한 쪽을 선택하여 그림을 완성한다. 그림 4.4(b)는 학습 시에 사용하지 않은 10가지 숫자 및 그림에 대한 드로잉 예시로, 다양한 이미지에 대해서 성공적으로 목표 이미지를 그려내었다. 그림을 그리기 시작하는 펜촉의 위치는 매 그림마다 임의로 결정되었다. 숫자 5나 의자를 그린 경우에는 불리한 위치에서 펜촉이 시작된 선분이 그림을 약간 벗어나 있는 것으로 표현되었다. 그러나 그러한 경우에도 적절히 펜촉을 이동하여 그림을 완성할 수 있도록 학습되었다. 완성된 그림과 목표 이미지와의 오차는 평균 약 6.24%로 계산되었다.

(a) 학습된 에이전트의 드로잉 과정

(b) 학습된 에이전트의 드로잉 결과

그림 4.4: 펜 드로잉 실험 결과

4.2 로봇 드로잉

4.2.1 문제

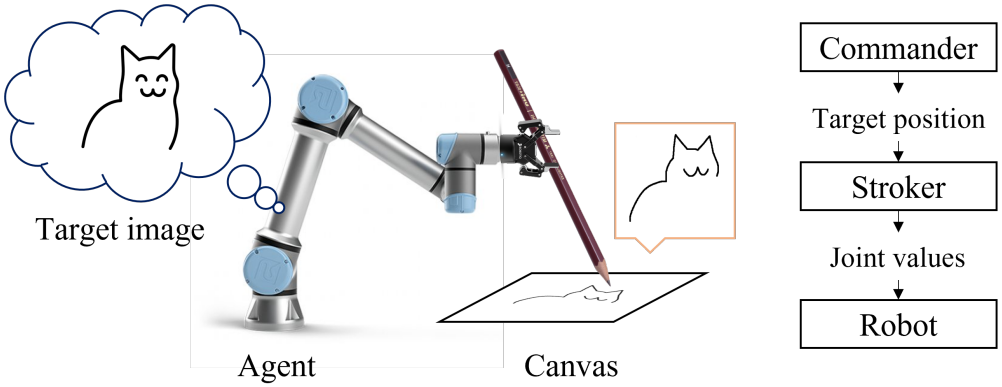


그림 4.5: 로봇 펜 드로잉 문제

4.1절에서는 가상의 도화지에서 그림이 그려졌지만, 로봇을 사용하면 현실세계의 도화지에 그림이 그려지도록 할 수 있다. 펜 드로잉은 그림 도구 하나만 사용하므로 페인팅이나 콜라주 기법에 비해 비교적 단순해 로봇에 적용하기 좋은 문제이다. 본 절에서는 그림 4.5와 같이 로봇 팔을 이용해서 실제 도화지에 펜 드로잉을 수행하는 문제를 다룬다.

4.2.2 방법

로봇을 이용한 펜 드로잉 문제는 펜 이동 위치 결정문제와 해당 위치를 달성하기 위한 로봇 팔 관절값 결정문제의 두 가지 계층적 문제로 나눌 수 있다. 먼저 지시자가 목표 이미지를 보고 도화지 위에서 펜을 어떻게 옮길 지를 결정하면, 수행자는 지시자의 결정에 따라 움직이기 위한 로봇 관절값을 결정하는 구조이다. 지시자는 4.1절에 제시된 펜 드로잉 문제와 거의 동일한 구조로 학습된다. 로봇 팔 관절값 결정문제는 고전적으로는 역기구학(inverse kinematics)을 사용하면 간

편히 해결할 수 있으나, 펜이 수직으로 세워진 상태에서 움직이는 등의 인위적인 움직임을 보이는 것이 단점이다. 이에 본 절에서 제안하는 방법에서는 강화학습을 사용하여 수행자가 여러 번의 시행착오를 통해 자연스럽게 펜을 움직이는 방법을 스스로 터득하도록 학습한다.

그림 4.6는 로봇을 이용한 펜 드로잉 방법 학습과 학습 이후 수행 시를 구분하여 나타낸 모식도이다. 학습 시에는 지시자(Commander)와 수행자(Stroker) 모두 강화학습하지만, 이들은 각자의 환경에서 학습한다. 지시자는 4.1절에서 펜 드로잉을 학습하는 방법과 같이 SNGAN의 discriminator를 보상함수로 이용하여 학습한다. 지시자는 상태 s_t^C를 관찰하고 행동 $a_t^C = (down, \Delta x, \Delta y)$를 결정한다. $down$은 이동 시 펜을 들 것인지 아니면 종이에 붙인 채로 그리면서 이동할 것인지를 결정하는 요소이다. 목표 이미지 G는 지시자의 목표로, 수행자 목표 g^S와의 구분을 위하여 g^C로 표기하였다. 수행자는 임의로 생성되는 목표 $g^S = (down, \Delta x, \Delta y)$를 상태 s_t^S와 함께 받아 해당 목표를 달성하기 위한 행동 a_t^S를 결정한다. 로봇은 6-자유도 관절로 이루어진 UR5를 사용하였다. 상태는 로봇의 각 관절값인 $s_t^S = (j_1, j_2, j_3, j_4, j_5, j_6)$이며, 행동은 각 관절값에 가할 변화량 $a_t^S = (\Delta j_1, \Delta j_2, \Delta j_3, \Delta j_4, \Delta j_5, \Delta j_6)$이다. 학습을 위한 강화학습 알고리즘은 지시자와 수행자 모두 SAC를 사용하였다. 학습 이후 수행 시에는 지시자의 행동이 곧 수행자의 목표 ($a_t^C = g^S$)가 되어 지시자와 수행자가 함께 동작한다.

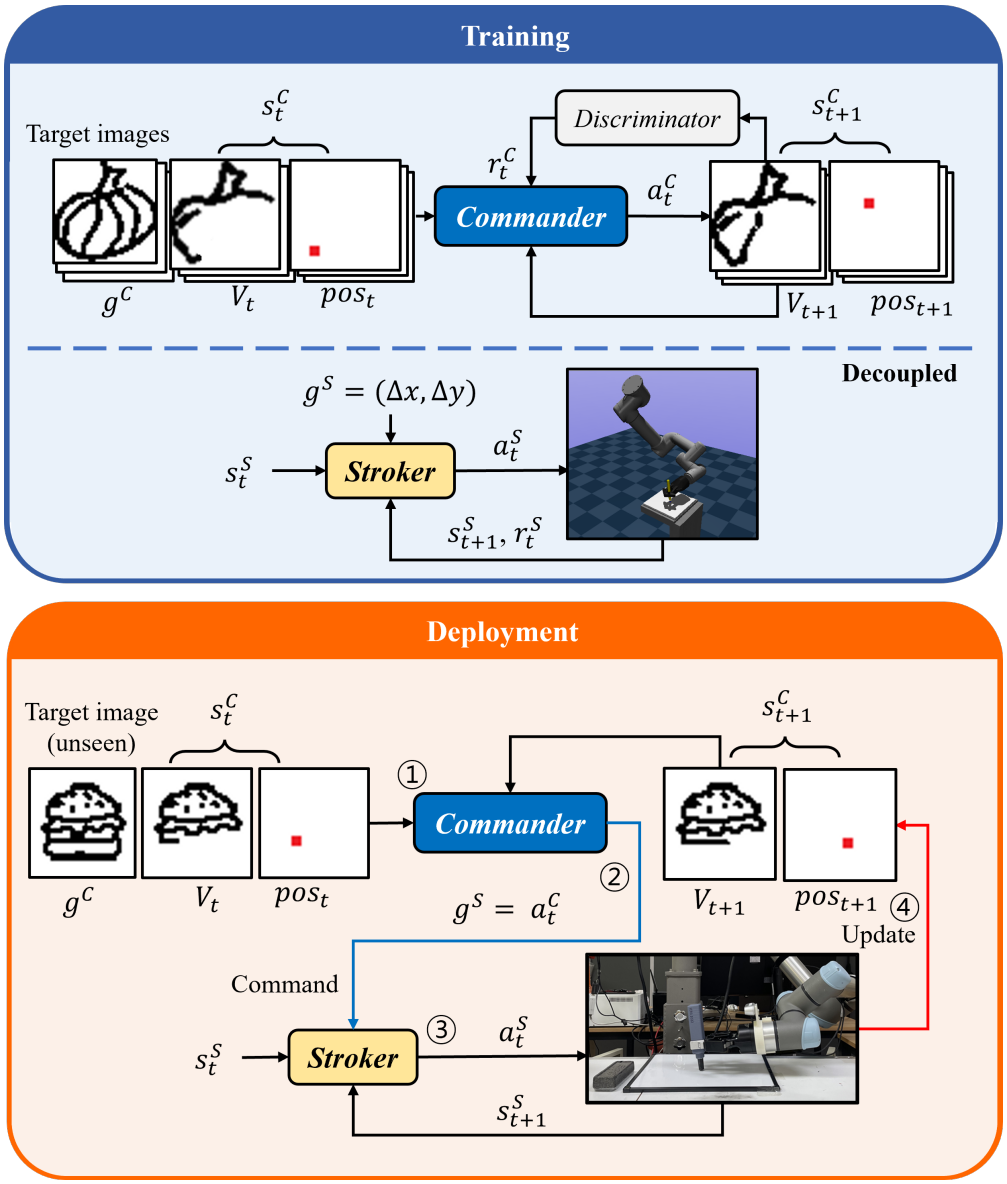


그림 4.6: 로봇 펜 드로잉 학습 및 수행 방법

지시자는 가상의 픽셀로 이루어진 도화지 공간에 그림을 그리면서 진행한다. 수행자는 많은 양의 시행착오를 필요로 하므로 시뮬레이션 환경 MuJoCo [17]에

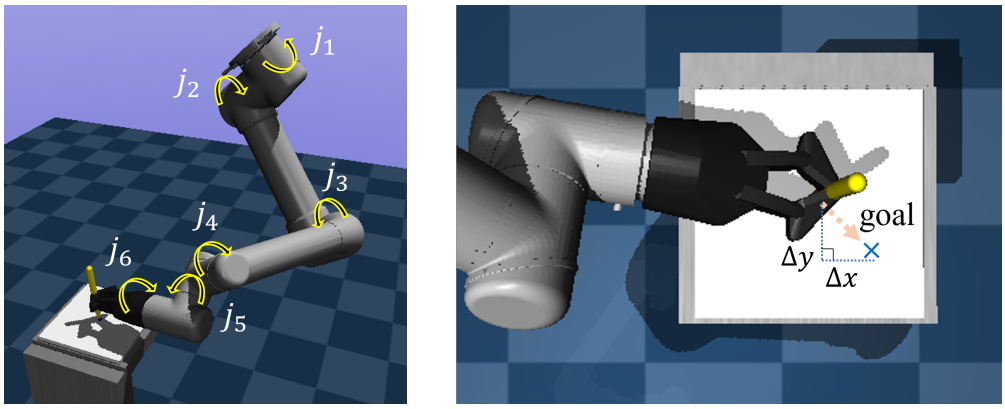


그림 4.7: 수행자 학습을 위한 시뮬레이션 환경

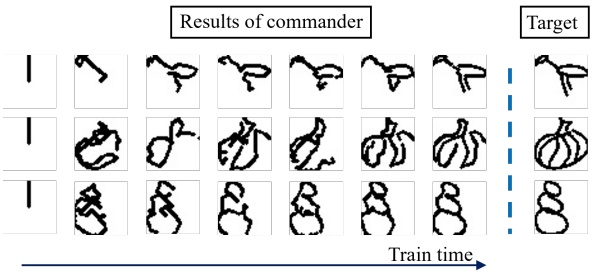
서 학습한다(그림 4.7). 그러나 실제 시나리오 수행 시에 수행자는 현실의 로봇에 적용하여 현실의 도화지에 그림을 그린다. 이때 지시자의 가상 도화지와 수행자가 그려야 하는 현실 도화지 간의 오차가 발생할 수 있다. 따라서 매 순간마다 수행자의 위치를 정기구학(forward kinematics)를 이용하여 측정한 후 지시자의 가상 도화지를 현실의 도화지와 일치하도록 교정하는 작업을 거친다. 수행자는 펜이 2-핑거 그리퍼에 의해 고정되어 있다고 가정하고 학습한다. 수행자를 학습하기 위한 보상함수는 아래와 같이 3개 항으로 구성된다.

$$\begin{aligned} r_t^S = &-10\sqrt{(g_x - p_{t_x})^2 + (g_y - p_{t_y})^2 + 5(g_z - p_{t_z})^2} \\ &+ 5\frac{[g_x\ g_y] \cdot [p_{t_x}\ p_{t_y}]}{\|[g_x\ g_y]\|\|[p_{t_x}\ p_{t_y}]\|} - \sqrt{(\theta_g - \theta_t)^2 + (\psi_g - \psi_t)^2} \end{aligned} \tag{4.1}$$

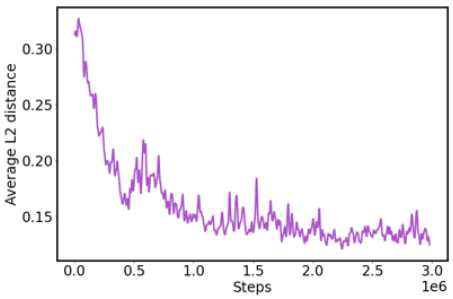
첫번째 항은 g^S가 나타내는 목표 위치 (g_x, g_y, g_z)와 행동에 의해 실제로 도달한 위치 $(p_{t_x}, p_{t_y}, p_{t_z})$와의 차이에 대한 유클리드 거리이다. z축은 $down$을 고려한 것으로 다른 축에 비해 5배 강조하였다. 두 번째 항은 목표 위치와 실제 도달 위치의 x, y축에 한한 코사인 유사도이다. 이는 그리려는 선분이 올바른 방향으로 위치하도록 도와준다. 세 번째 항은 펜의 기울기를 안정적으로 유지하기 위한 것으로, 바람직한 펜의 오일러각 성분 중 두 성분 θ, ψ에 대한 모범 각도 (θ_g, ψ_g)와 실제

각도 (θ_t, ψ_t) 사이의 오차를 나타낸다. 다른 하나의 성분 ϕ는 펜의 머리에서부터 펜촉을 관통하는 긴 축에 대한 성분으로, 이 성분이 바뀌어도 펜촉 위치는 변함이 없어 보상항에 반영하지 않았다.

4.2.3 실험 및 결과



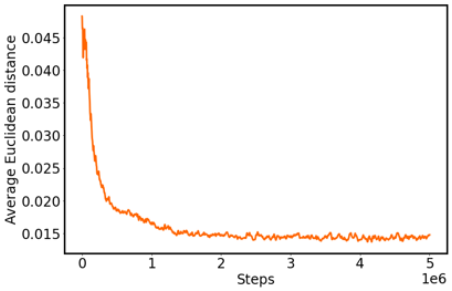
(a) 학습 경과에 따른 최종 그림 변화



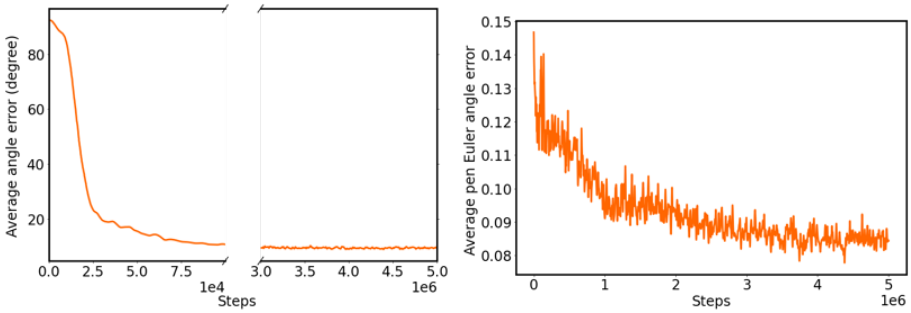
(b) 학습 경과에 따른 L2 거리 변화

그림 4.8: 지시자의 학습 경과

지시자의 학습을 위해 4.1절에서도 사용했던 *Quick, Draw!* 데이터셋을 사용하여 목표 이미지를 제공하였다. 지시자는 총 6만회의 그림을 그리며 학습했으며, 한 그림당 50번의 행동을 선택하였다. 수행자는 500만개의 임의로 생성되는 목표에 대하여 학습했으며, 하나의 목표 당 1번의 행동을 선택하였다. 지시자가 보는 도화지 크기는 42×42 픽셀로, 현실에서 사용하는 도화지의 크기는 21×21 cm



(a) 학습 경과에 따른 거리 오차 변화



(b) 학습 경과에 따른 각도 오차 변화 (c) 학습 경과에 따른 기울기 오차 변화

그림 4.9: 수행자의 학습 경과

으로 설정하였다.

그림 4.8(a)는 지시자의 학습 경과에 따른 완성된 그림의 변화를 나타낸 것이다. 지시자는 가상 도화지 상에서 점차 목표하는 그림과 유사하게 그림을 그려내었다. 실제로 그림 4.8(b)에 학습 경과에 따라 지시자의 최종 그림과 목표 이미지 사이의 l_2 거리가 감소하는 것이 나타나 있다. 수행자의 학습 경과에 따른 그래프는 그림 4.9의 (a), (b), (c)에 각 보상항과 관련하여 나타나 있다. 학습이 진행됨에 따라 수행자의 보상 설계 의도와 일치하도록 각각의 오차 요소가 감소하였다.

학습을 모두 마친 뒤 실제 로봇을 이용하여 진행한 실험 환경은 그림 4.10와 같다. 실험을 위한 목표 이미지는 학습시 사용하지 않은 이미지들로 구성하였다.

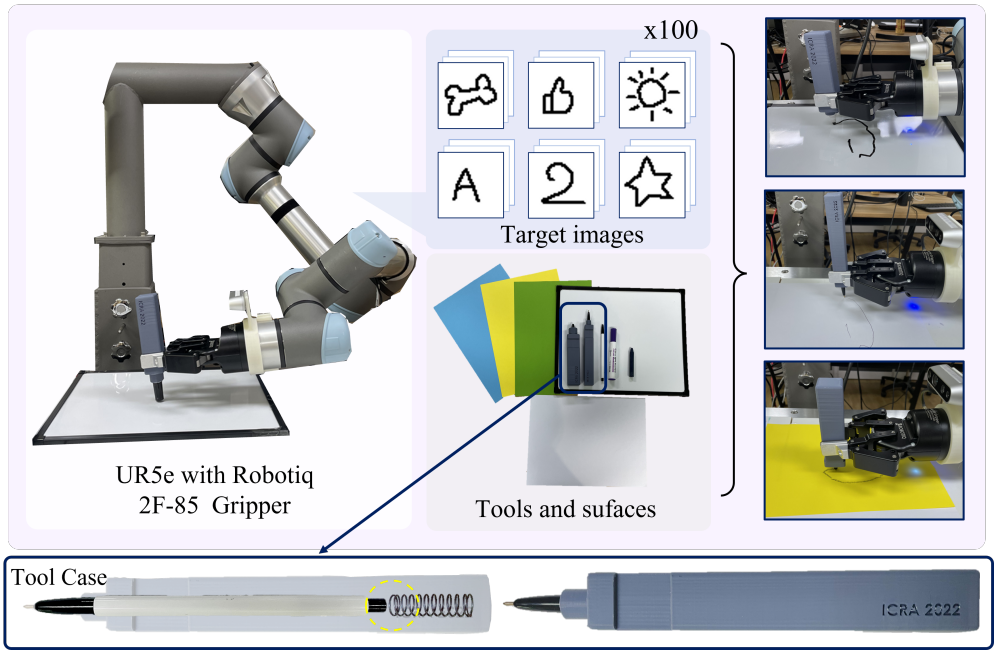
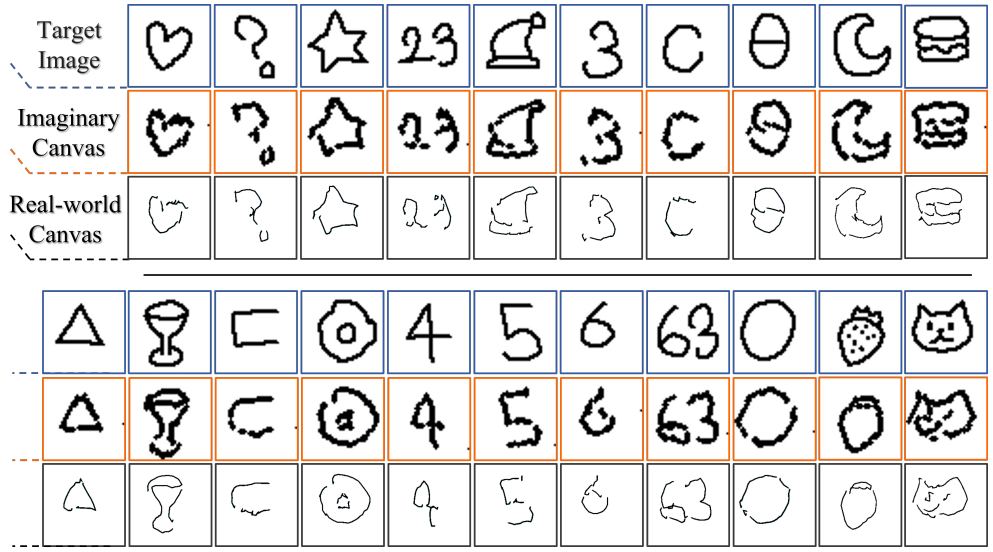
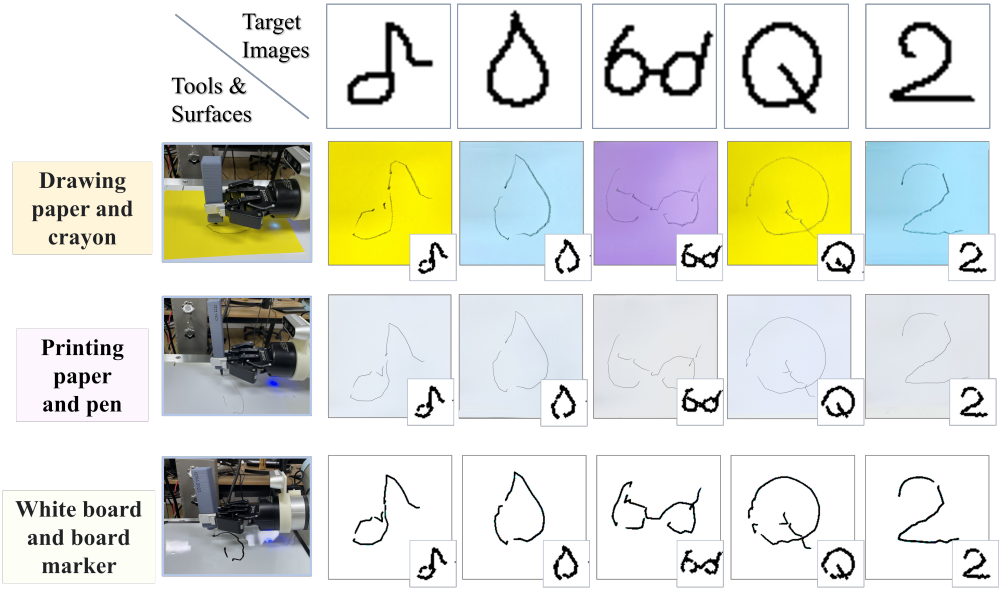


그림 4.10: 실험 환경

펜과 책상과의 충돌로 인한 로봇의 비상 작동중지를 방지하기 위해 뒤쪽에 스프링을 장착한 케이스를 그림 도구에 장착하였다. 그림 도구는 마커&보드, 볼펜&A4 용지, 크레용&색지의 세 가지 세트로 구성하였다. 그림 4.11(a)는 마커&보드로 다양한 목표 그림에 대하여 수행된 결과로, 첫 번째 행은 목표 이미지, 두 번째 행은 가상 도화지, 세 번째 행은 실제 도화지이다. 학습된 에이전트는 대부분 목표 그림과 유사하게 그림을 그려내었다. 그러나 하단 우측에서 두 번째 그림(딸기)에 대해서는 딸기씨를 그리지 못했는데, 이는 지시자의 한계로 *Quick, Draw!* 데이터셋의 낮은 복잡도에 기인한다. 그림 4.11(b)는 세 가지 다른 그림 도구 세트에 대한 수행 결과이다. 학습된 에이전트는 다른 여러 그림 조건들에 대해서도 성공적으로 그림을 그려낼 수 있었다.



(a) 다양한 그림에 대한 실험 결과



(b) 다양한 그림 조건에서의 실험 결과

그림 4.11: 실험 결과

제 5 장 고급 드로잉 기법으로의 응용

5.1 스트로크-조건부 페인팅

5.1.1 문제

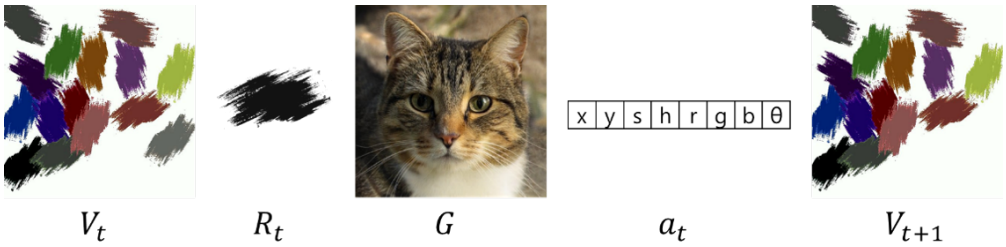


그림 5.1: 스트로크-조건부 페인팅

4.1절의 펜 드로잉에서는 펜촉의 위치만을 조정하며 그림을 그렸지만, 페인팅에서는 매 순간 새로운 위치에 다양한 색상의 붓터치를 적용한다. 강화학습을 이용한 페인팅은 [7, 8, 4, 10, 9]에서 이미 다루었으므로 본 절에서는 스트로크에 대해서 보다 일반화된 페인팅 문제인 스트로크-조건부 페인팅 문제를 새로 제시하고 그에 대한 방법에 대하여 다룬다. 기존 페인팅 방법들에서는 스트로크의 형식이 고정되어 학습된다. 예를 들면, 어떤 견본 붓터치 방식이 있다고 할 때 해당 방식만을 사용하여 학습된 에이전트는 연필이나 원형 스티커 등의 다른 스트로크 형식을 이용하여 그림을 그릴 수 없다. 제안하는 스트로크-조건부 페인팅 문제에서는 특정 스트로크 형식에 맞추어 학습하는 대신, 스트로크 형식까지 입력으로 받아 행동을 결정하도록 한다. 이렇게 학습된 에이전트는 매 순간마다 사용자가 임의로 제공하는 스트로크 형식에 따라 드로잉할 수 있다.

5.1.2 방법

그림 5.1는 스트로크-조건부 페인팅에서 그림이 그려지는 방식에 대한 모식도이다. 환경을 표현하기 위한 상태는 $s_t = (V_t, G, R_t, l_t, c)$로 정의한다. R_t는 순간 t에서 주어진 스트로크의 형식 이미지이다. T를 하나의 그림에서 사용할 총 스트로크 수라고 할 때, $l_t = tanh((T-t)/T)$는 그림이 완성되기까지 남은 가용 스트로크 수에 대한 정보로 에이전트가 전체 드로잉 과정에서 속도 조절을 할 수 있도록 돕는다. c는 도화지의 좌표 정보를 담고 있는 상수벡터로, 페인팅과 같이 위치 정보가 중요한 작업에서 이미지와 함께 합성곱 신경망에 입력되면 성능에 도움을 주는 것으로 알려져 있다 [18]. 에이전트는 매 순간 형식 R_t를 기반으로 한 스트로크를 도화지 V_t에 배치하여 목표 이미지 G와 가까워지도록 해야 한다. 이때의 행동을 $a_t = (x_{coord}, y_{coord}, s_{size}, r_{color}, g_{color}, b_{color}, h_{bright}, \theta_{angle})$로 정의한다. x_{coord}, y_{coord}는 도화지에 스트로크를 배치할 위치, s_{size}는 스트로크의 크기, r_{color}, g_{color}, b_{color}는 RGB로 표현되는 색조, h_{bright}는 밝기, 그리고 θ_{angle}은 각도를 나타낸다. 이 행동에 의해 스트로크 형식 R_t이 변형되고 도화지에 배치된다.

강화학습 알고리즘은 SAC를 사용하며, 보상함수의 설계는 4.1.2에서 제시한 방법과 동일하다. 단, 행동이 스트로크의 크기를 조정할 수 있어 스트로크의 크기가 매우 큰 경우와 작은 경우의 보상 차이가 커질 수 있다. 이에 따른 보상 편차를 완화하기 위해 보상에 역탄젠트 함수를 취하여 보상의 규모를 제한한다.

매순간 참고해야 할 스트로크 형식이 다르게 제시되면 그만큼 미래를 예측하기 어렵고 문제의 복잡도가 높아진다. 이에 본 문제에 대한 방법에서는 SAC의 특성을 활용하여 행동을 보다 신중히 선택하도록 하였다. SAC에서는 행동을 확률적으로 결정하는데, 학습 시에는 정책 신경망이 가우시안 분포의 모수인 평균 μ와 표준편차 σ를 출력하면 해당 모수로 형성된 확률분포에서 샘플링을 통해 행동을 선택한다. 학습 이후에는 샘플링 대신 확률분포의 평균값으로 행동을 결정하는 것이 일반적이다. 그러나 본 방법에서는 학습 이후 예측할 수 없는 스트로크에

대한 대응력과 그림의 품질을 높이기 위해 행동 확률분포에서 N개의 행동 후보 $A_t^{cand} = \{a_t^{(1)}, a_t^{(2)}, ..., a_t^{(N)}\}$를 샘플링한 후 행동 평가를 통해 그 중 가장 적절한 행동을 선택한다. 행동 평가는 V_t와 G 간의 l_2 거리를 활용하여 다음과 같이 행동 a_t을 결정한다.

$$a_t = \arg\min_{a_t^{(i)}} l_2(V_{t+1}(V_t, R_t, a_t^{(i)}), G),\ a_t^{(i)} \in A_t^{cand} \tag{5.1}$$

$$a_t^{(i)} \sim \mathcal{N}(\mu, \sigma^2) \tag{5.2}$$

$$l_2(\mathbf{x}, \mathbf{y}) = \frac{1}{K}\sum_{i=1}^{K}(\mathbf{x}_i - \mathbf{y}_i)^2 \tag{5.3}$$

5.1.3 실험 및 결과

실험에 사용할 에이전트를 위해 한 그림에 소요되는 행동 수 $T = 200$으로 총 100만 번의 그림을 그리면서 학습하였으며, 목표 이미지 G와 초기 도화지 V_0는 학습 중 임의로 생성되는 이미지를 사용하였다. 학습 중 임의의 이미지를 생성하기 위해 흰 도화지에서 임의의 중심좌표와 색상을 가지는 직사각형을 100회 배치한 후 최종 이미지를 G로, 중간 단계의 이미지를 임의로 하나를 골라 V_0로 사용하였다. 배치 순서에 따라 직사각형의 크기는 점차 작아지도록 하였다. 스트로크 형식 이미지 R_t는 Describable Texture Dataset (DTD)[19]의 다양한 질감 이미지를 매 순간마다 임의의 모양으로 잘라 사용하였다. 모든 이미지의 크기는 신경망에 입력되기 전에 64×64으로 변형되었다.

학습된 에이전트에 대한 실험 결과를 나타내기 위해 사용한 목표 이미지는 512×512 크기의 이미지를 사용하였다. 고화질의 그림을 페인팅하기 위해서 [4, 9]에서 사용된 방법과 같이 처음에는 전체 이미지를 목표로 그렸다가 점차 목표를 4분할하여 병렬적으로 그리는 방법을 채택하였다. 4분할은 총 4번 진행되며(최종 256분할), 분할되기 전 단계를 포함해 매 분할 단계에서는 분할된 각 영역에 대해서

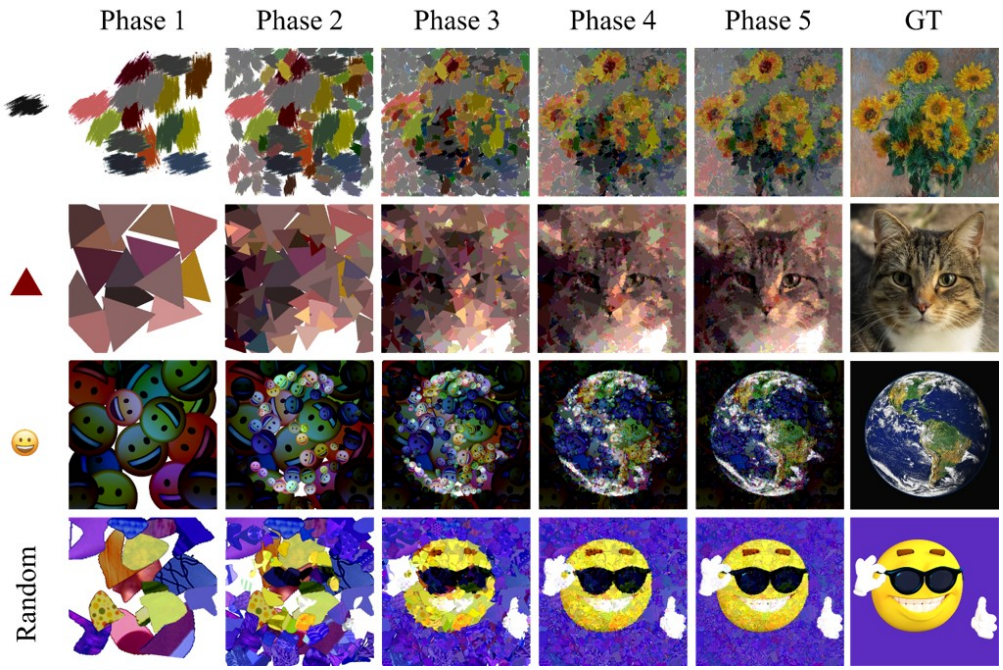


그림 5.2: 학습된 에이전트의 스트로크-조건부 페인팅 수행 과정

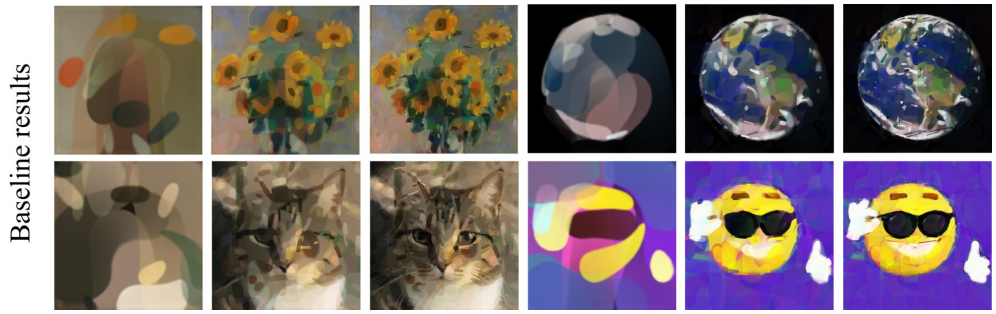


그림 5.3: 기존 방법 [4]으로 페인팅이 수행되는 과정

15개의 스트로크가 놓여진다. 매 순간마다 샘플링하는 행동 후보 개수는 $N = 30$으로 설정하였다. 감가율은 $\gamma = 0.95$를 사용하였다.

그림5.2은 스트로크의 종류에 따라 학습된 에이전트에 의해 페인팅이 그려지는 과정을 분할 단계에 따라 도시한 것으로, GT는 목표 이미지이다. Random은 매 순

표 5.1: 스트로크에 따른 목표 이미지 및 제안된 방법으로 그려진 그림 간 단계별 평균 $l2$ 거리

	Brush	Triangle	Smile	Random
Phase 0	.3566	.3566	.3566	.3566
Phase 1	.2728	.2059	.2138	.2275
Phase 2	.1864	.1574	.1842	.1790
Phase 3	.1589	.1541	.1775	.1662
Phase 4	.1555	.1541	.1746	.1626
Phase 5	.1546	.1544	.1722	.1611

간마다 DTD 이미지를 임의로 선택하고 잘라 스트로크 이미지로 사용한 경우이다. 모두 각자의 스트로크 조건 하에서 목표하는 그림과 유사한 그림을 그려내었다. 그림 5.3은 기존 방법[4]으로 수행된 결과이다. 제안된 방법과 달리 학습에 사용한 단 하나의 스트로크 형식만을 사용할 수 있다. 기존 방법에 비해 스트로크가 바뀌는 상황에 대응하여 페인팅을 수행할 수 있는 이유는 에이전트가 실제로 무작위로 변이하는 스트로크를 받으면서 학습하였기 때문이다. 또한, SAC 특성을 활용한 행동 샘플링 방법이 한 스트로크에 대하여 여러 번 시도해본 후 행동을 결정하도록 하므로 변이하는 스트로크에 더 면밀히 대응할 수 있다. 이는 화가가 실제로 붓터치를 하기 전 상상 속에서 다양한 붓터치를 시뮬레이션하는 과정에 비유할 수 있다.

표 5.1은 각 단계에서 스트로크 종류에 따른 목표 그림과 그려지는 이미지 간의 평균 $l2$ 거리를 나타낸 표이다. Phase 0은 완전한 백지 상태이므로 모든 그림에서 거리가 동일하다. 단계가 진행되면서 각 그림에 대한 거리는 점차 감소한다. 완성된 그림에서 삼각형 스트로크에서 가장 가까웠고, 이모티콘 스트로크에서 가장

멀었다. 삼각형 스트로크는 단색이고 기하적으로 단순하여 그림 표현이 비교적 쉬운 반면, 이모티콘 스트로크는 삼각형에 비해 복잡한 형태라는 속성이 반영되었기 때문이다.

5.2 콜라주

5.2.1 문제

(a) 콜라주 그림[1]

(b) 원본 그림

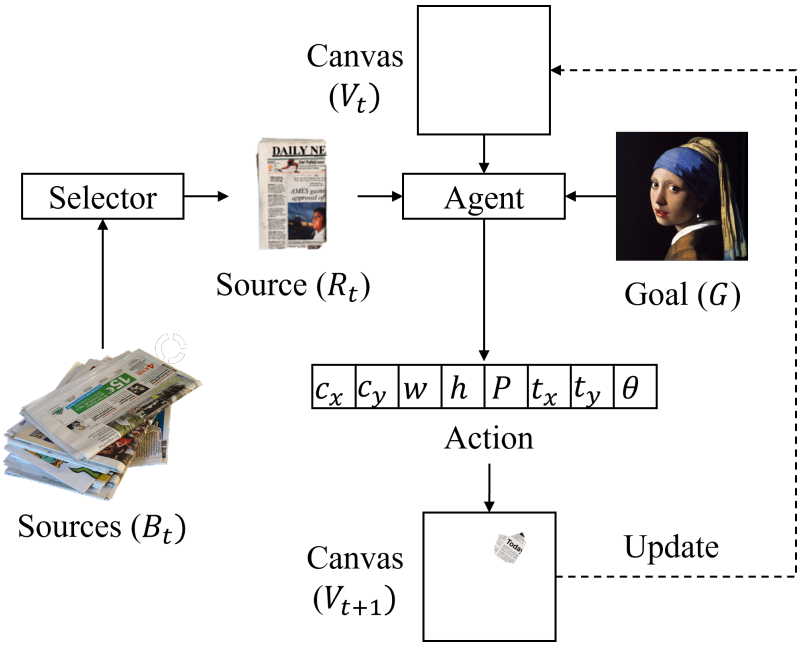
그림 5.4: 콜라주 기법

강화학습을 이용한 SBR 방법은 드로잉 영역을 넘어서 더 넓은 예술적 기법에 활용할 수 있다. 콜라주 기법이란 시각 예술에서 주로 쓰이는 기법으로 질이 다른 여러가지 헝겊, 비닐, 타일, 나뭇조각, 종이 상표 등을 붙여 화면을 구성하는 기법이다. 어원은 "접착하다"는 의미의 프랑스어 coller에서 파생되었다. 그림 5.4은 콜라주 기법으로 그려진 그림으로 신문지, 잡지 등을 원본 그림의 형태에 맞게 도화지에 오려붙여 재구성하였다.

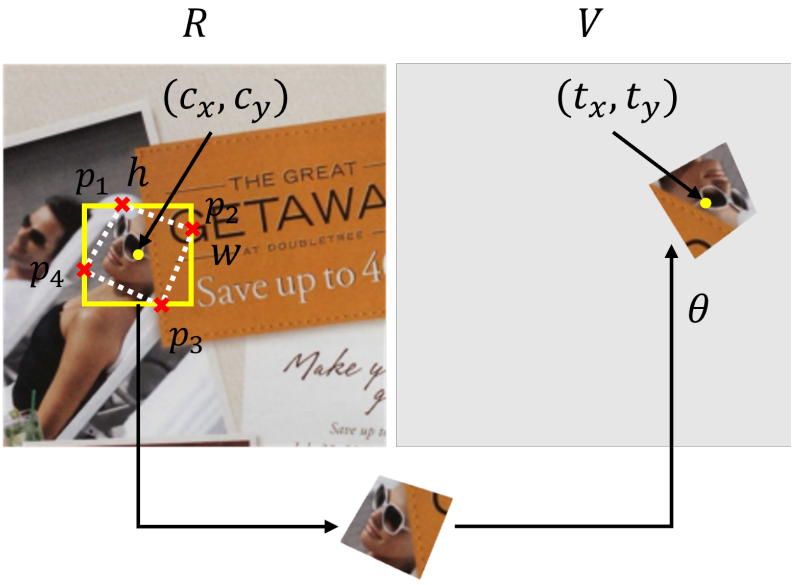
5.2.2 방법

그림 5.9(a)는 콜라주 환경에 대한 모식도이다. 콜라주 환경을 표현하기 위한 상태는 $s_t = (V_t, G, R_t, l_t, c)$로 5.1절의 스트로크-조건부 페인팅에서와 유사하다. 대신 콜라주에서 R_t는 스트로크 형식이 아닌 하나의 재료에 해당한다. 재료 선택자(Selector)는 여러 재료 모음(B_t) 중 하나의 재료 R_t를 선택하고, 에이전트는 해당

[1] https://www.pinterest.co.kr/pin/667095763568114483/



(a) 콜라주 기법의 적용 과정



(b) 콜라주에서의 행동벡터

그림 5.5: 콜라주 문제 환경

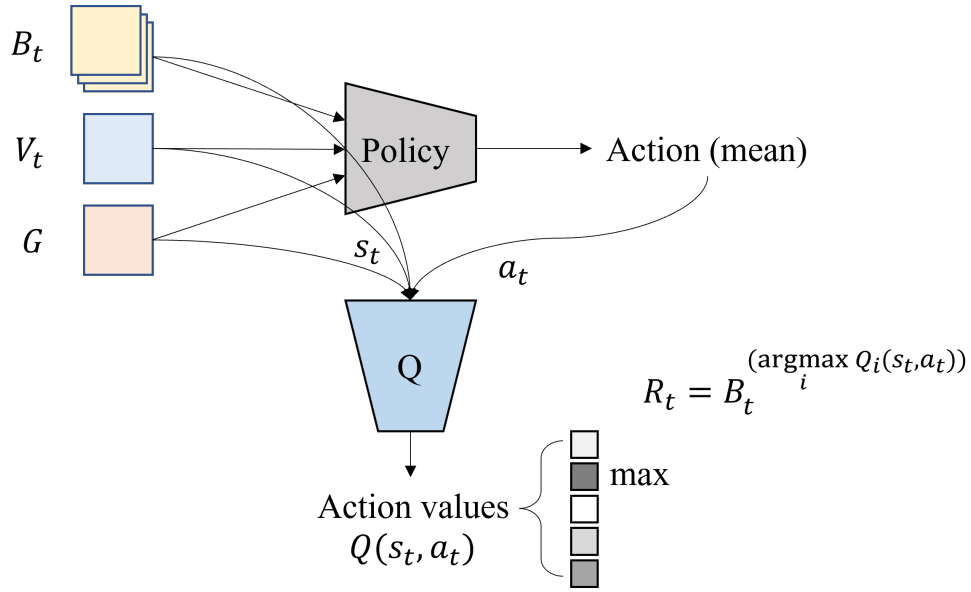


그림 5.6: 콜라주 에이전트의 최적 재료 선택 과정

재료를 받아 행동을 결정한다. 에이전트의 행동은 $a_t = (c_x, c_y, w, h, P, t_x, t_y, \theta)$로 표현된다. c_x, c_y는 재료를 잘라낼 사각형의 프레임을 결정할 중심좌표이며, w, h는 해당 프레임의 가로 및 세로 길이이다. $P = (p_1, p_2, p_3, p_4)$는 프레임 각 변의 점을 결정할 비율이다. P에 의해 최종적으로 오려낼 모양이 결정되며, 오려낸 조각은 t_x, t_y가 가리키는 좌표와 회전값 θ 따라 도화지에 붙여진다. 그림 5.9(b)는 에이전트의 행동이 어떻게 도화지에 적용되는 지를 시각화한 것이다.

강화학습 알고리즘은 SAC를 사용하며, 보상함수는 5.1.2항에서와 동일한 방식을 사용한다. 행동 선택 또한 5.1.2항에서 제시한 대로 SAC의 특성을 활용한 샘플링 방법을 사용한다. 매순간 샘플링하는 행동 후보 수 또한 동일하게 $N = 100$으로 설정하였다. 재료 모음 B_t에서 재료 R_t를 선택하기 위해서도 SAC가 행동가치함수 $Q(s_t, a_t)$를 학습한다는 특성을 활용하였다. 먼저 재료 모음을 SAC의 정책 신경망에 대입하여 각 재료들에 대한 행동 모음을 얻는다. 이후 상태와 행동 모음들을 행동가치함수에 대입하면 현재 상태에서 각 재료들에 대해 추론된 행동가치

모음을 얻을 수 있다. 이 행동가치 모음 중 가장 큰 행동가치를 찾고, 해당 행동가치를 결정했던 재료를 최적 재료로 선택한다. 이에 대한 과정은 그림 5.6에 나타나 있다. 재료 모음에 대한 신경망의 출력은 병렬적으로 이루어지므로 최적 재료 선택 과정에서 반복에 의한 시간 소요는 발생하지 않는다.

5.2.3 실험 및 결과

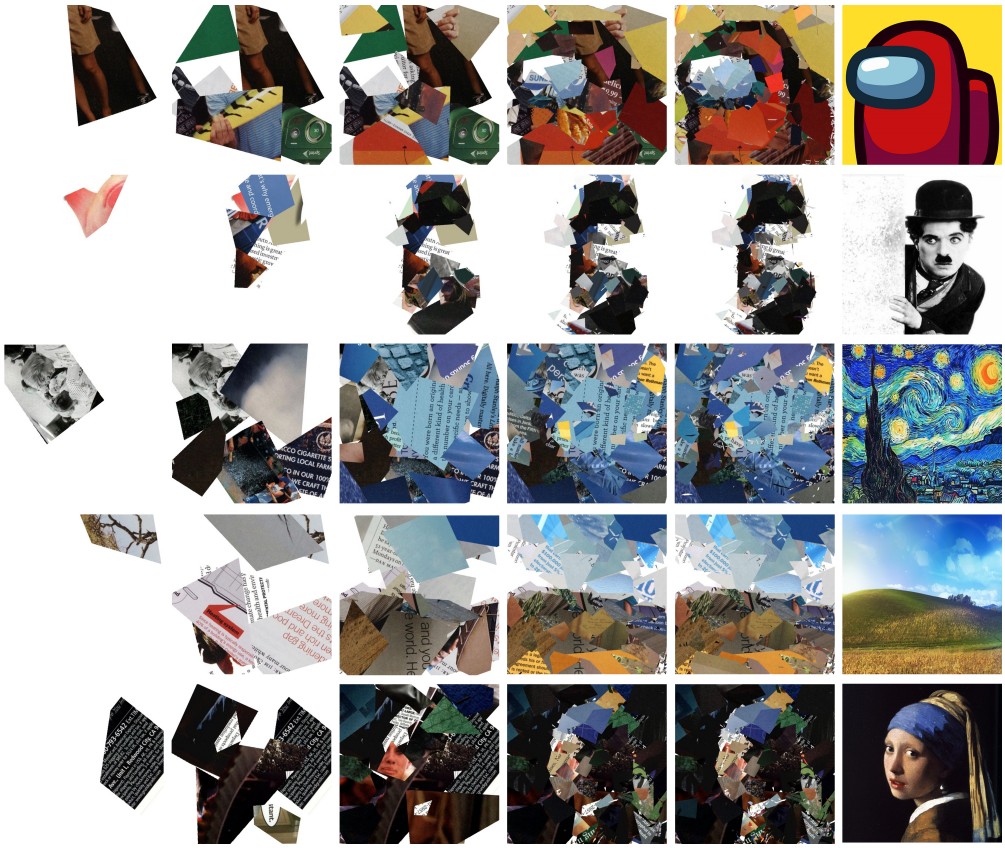


그림 5.7: 학습된 에이전트의 콜라주 수행 과정

그림 5.8: 50개 스트로크(좌), 200개 스트로크(중), 1000개 스트로크(우)

(a) G (b) V_T (128×128) (c) V_T (512×512)

그림 5.9: 고화질 그림에 대해 발생한 픽셀 밀도 차이

콜라주를 수행할 에이전트를 얻기 위해 한 그림 당 소요되는 행동의 수는 $T = 200$으로, 16개의 병렬 환경에서 각 10만회의 그림을 그리며 학습하였다. 학습을 위해 제공되는 목표 이미지와 초기 도화지는 5.1.3항에서 제시한 바와 같은 방법으로 임의로 생성되는 이미지를 사용하였다. 재료 데이터셋 또한 5.1.3항에서 사용한 데이터셋인 Describable Texture Dataset (DTD)를 사용하였다. 에이전트는 학습 중 매순간 DTD 데이터셋에서 임의로 추출된 하나의 이미지를 재료로 사용하였다. 학습 및 추론에 사용된 이미지의 크기는 모두 신경망 입력 전에 128×128 픽셀로 변형되었다. 학습된 에이전트의 콜라주 수행에 사용된 재료 모음은 2001년부터 2015년까지의 타임(Time) 잡지 데이터[2]를 사용하였으며, 목표 이미지는 512×512 크기의 이미지를 사용하였다. 에이전트는 128×128 픽셀 수준에서 행동을 결정하지만, 모든 행동을 녹화한 후 각 스트로크의 스케일을 늘려 512×512 픽셀 수준에서 재구성하여 고화질의 결과를 얻었다.

그림 5.7는 학습된 에이전트가 다양한 목적 이미지에 대해 200회의 행동 결정을 걸쳐 콜라주로 재구성한 결과이다. 첫 번째 열은 각 목적 이미지에 대해 처음으로 놓여진 스트로크로, 잡지에서 일정한 모양으로 오려져 붙여진 것을 확인할 수 있

[2]https://www.kaggle.com/datasets/thegupta/time-magazine-part-9-2001-to-2015

다. 그림이 진행됨에 따라 점차 개별 잡지 조각들이 모여 목표 이미지에 가깝게 표현되었다. 처음에는 잡지가 크게 오려져 전체 구도를 잡지만, 나중에는 점점 오리는 크기가 작아져 상세한 부분이 표현되는 것을 볼 수 있다. 이는 에이전트에게 저절로 발현된 행동으로, 그림 초기에는 큰 잡지 조각을 붙이는 것이 많은 보상을 가져오지만, 이후 그림이 많이 진행된 상태에서 큰 잡지 조각을 붙이는 것은 오히려 그려놓은 것을 가려 음의 보상을 가져올 가능성이 높기 때문이다. 그림 5.8는 왼쪽에서부터 각각 50, 200, 1000여개의 스트로크를 사용하여 그린 결과로, 스트로크 수가 많아질 수록 표현의 섬세함은 증가하지만 추상화 정도는 감소한다. 따라서 스트로크의 수를 조절하여 원하는 스타일의 그림을 얻을 수 있다. 단, 1000개 스트로크의 경우 잡지 조각이 매우 작아지면서 이미지에 약간의 잡음이 발생하는데, 이는 128×128 픽셀 수준에서 결정된 행동을 512×512 픽셀 수준으로 재구성하면서 발생하는 문제이다. 그림 5.9에서 나타난 바와 같이 재구성하지 않은 128×128 그림에서는 발생하지 않았던 잡음이 128×128 그림에서 발생한다. 이는 보다 고화질의 이미지를 사용해 에이전트를 학습하면 해결할 수 있다.

제 6 장 결론 및 향후 연구

6.1 결론

본 학위논문에서는 강화학습을 이용하여 SBR을 수행하는 에이전트의 학습법을 일반화하여 제시한 후 이를 펜 드로잉이나 콜라주 등의 다양한 기법으로 확장하여 적용하고 각 기법에 해당하는 방법 및 실험 결과를 상세히 기술하였다. 각 실험 결과 에이전트의 학습 의도에 알맞게 드로잉을 수행하는 것을 확인하였다. 펜 드로잉 문제에서 제한된 스트로크 수로 그림을 완성하기 위해 펜촉은 효율적으로 움직여야 한다. 그리고 펜촉의 끝점은 곧 다음 순간에서의 시작점이 되므로, 다음 순간에서의 움직임 또한 고려하여 펜촉을 움직여야 한다. 페인팅이나 콜라주의 경우에도 스트로크 수는 제한되었고, 한 번의 스트로크가 이후의 모든 상태에 영향을 미치므로, 드로잉은 순간만이 아닌 미래의 그림 상태들을 고려하여 진행되어야 한다. 강화학습은 상태의 미래 가치를 판단하며 학습하므로 이러한 작업요건을 달성하기에 매우 적절한 학습 방법이다.

제안된 방법과 결과는 심층 강화학습이 다양한 시나리오의 드로잉 작업에서 유용하게 활용될 수 있음을 보여준다. 각 작업의 특성에 맞추어 제안된 알고리즘은 일반화된 알고리즘을 다양하게 변형하는 예시를 보여준다. 본 학위논문의 내용은 드로잉과 관련한 인공지능 에이전트 설계를 위한 참조자료가 될 수 있으며, 이러한 기술은 드로잉 소프트웨어, 영화, 애니메이션, 장난감, 공연 및 전시 등의 다양한 예술 산업 분야에 활용 가능할 것으로 기대된다.

6.2 향후 연구

강화학습은 학습에 상당한 양의 시나리오 경험이 필요하며 그에 따라 많은 시간이 소요된다. 특히 고차원 비전 데이터를 매순간 처리해야 하는 SBR 작업은 더욱 고비용의 작업이다. 따라서 학습 효율을 높이는 것은 향후 연구의 중요한 방향이 된다. 작업 특성 면에서의 한계는 제시한 방법들이 모두 참조해야 할 목표 이미지를 필요로 하고, 에이전트 스스로 그림을 생성하지는 못한다는 점이다. 향후 연구에서는 대형 언어모델 등과 융합하거나, [8]에서와 같이 그림의 생성을 위한 문제를 설계하는 것이 의미가 있을 것이다. 표현 학습 기술을 활용하여 목표 이미지에 대한 잠재 변수를 얻은 후 그림 생성을 다양화하는 방식도 고려해볼만 하다. 보상 함수는 수식을 통해 명확하게 제시되었지만, 인간이 개입하여 보상을 제공하는 인간-컴퓨터 상호작용을 활용하면 보다 인간에 가까운 예술감각을 지닌 에이전트를 설계할 수 있다. 한편, 본 논문에서 제시한 방법들 또한 일종의 탐욕적 방법으로 한 번 놓은 스트로크는 이후 변경하지 않는다. 그러나 사람이 지우개, 화이트 등을 이용하는 것처럼 그림을 수정하고 교정하는 작업 또한 문제에 포함된다면 더 고품질의 드로잉을 수행하는 에이전트를 만들 수 있을 것이다. 로봇으로 확장한 예시에서는 펜촉의 위치 정보를 이용해 가상 도화지를 갱신하면서 드로잉이 진행되었지만, 향후 연구에서는 카메라를 통한 시각 정보로 실제 도화지를 의사결정에 직접 반영한다면 더 다양한 드로잉 도구와 환경 변이에 대응할 수 있는 드로잉 로봇 에이전트를 학습할 수 있을 것이다.

참고문헌

[1] A. Hertzmann and D. Zorin, "Illustrating smooth surfaces," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 517–526, 2000.

[2] A. Secord, "Random marks on paper: Non-photorealistic rendering with small primitives," in *MASTER'S THESIS*, Citeseer, 2002.

[3] P. Haeberli and P. B. Numbers, "Abstract image representations," *Computer Graphics*, vol. 24, no. 4, p. 207.

[4] Z. Huang, W. Heng, and S. Zhou, "Learning to paint with model-based deep reinforcement learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8709–8718, 2019.

[5] A. Hertzmann, "A survey of stroke-based rendering," Institute of Electrical and Electronics Engineers, 2003.

[6] D. Ha and D. Eck, "A neural representation of sketch drawings," *arXiv preprint arXiv:1704.03477*, 2017.

[7] Y. Ganin, T. Kulkarni, I. Babuschkin, S. A. Eslami, and O. Vinyals, "Synthesizing programs for images using reinforced adversarial learning," in *International Conference on Machine Learning*, pp. 1666–1675, PMLR, 2018.

[8] J. F. Mellor, E. Park, Y. Ganin, I. Babuschkin, T. Kulkarni, D. Rosenbaum, A. Ballard, T. Weber, O. Vinyals, and S. Eslami, "Unsupervised doodling and painting with improved spiral," *arXiv preprint arXiv:1910.01007*, 2019.

[9] S. Liu, T. Lin, D. He, F. Li, R. Deng, X. Li, E. Ding, and H. Wang, "Paint transformer: Feed forward neural painting with stroke prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6598–6607, 2021.

[10] Z. Zou, T. Shi, S. Qiu, Y. Yuan, and Z. Shi, "Stylized neural painting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15689–15698, 2021.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[12] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *International Conference on Machine Learning*, pp. 1352–1361, PMLR, 2017.

[13] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[15] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[16] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[17] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033, IEEE, 2012.

[18] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski, "An intriguing failing of convolutional neural networks and the coordconv solution," *Advances in neural information processing systems*, vol. 31, 2018.

[19] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

Abstract

Design of Artificial Drawing Artist Using Deep Reinforcement Learning

Lee Ganghun
Interdisciplinary Program in Cognitive Science
The Graduate School
Seoul National University

People easily draw to express their thoughts, while drawing requires some level of aesthetic sense or even pondering ability. We can find drawing is a high-intellectual behavior when we recall the fact that drawing is common in people but not in animals. For that reason, hand-crafted algorithms devised for automated drawing have limitations representing the subtle pattern of diverse drawing styles. However, several recent artworks have engaged deep learning techniques, filling the gap between computational automation and aesthetic sense. Stroke-based rendering (SBR), one of the computer graphics-based drawing tasks, is a task placing "stroke" such as brush touches and stipples on the digital canvas to complete target drawings. In this article, total 4 kinds of reinforcement learning algorithms for each style of SBR scenarios are presented. Those scenarios are divided into pen drawings and high-level drawings: pen drawing denotes doodle-like drawings drawn with single-colored pen, while

high-level drawing denotes painting-like drawings drawn with diverse realistic strokes or some parts of magazines. This article can be a useful reference to artificial drawing agents using deep reinforcement learning, and the proposed methods can be utilized to various art industries such as drawing software, movies, animations, toys, shows, exhibitions, and so on.

Keywords: Reinforcement Learning, SBR, Stroke, Pen Drawing, Painting, Collage

Student Number: 2020-27294