Dragging Score Distillation

한상범[1], 이민식[1,2]

[1]한양대학교 전자공학과, [2]한양대학교 ERICA 전자공학부

gkstkdqja88@hanyang.ac.kr, mleepaper@hanyang.ac.kr

Dragging Score Distillation

Sangbeom Han[1], Minsik Lee[1,2]

[1]Department of Electrical and Electronic Engineering, Hanyang University

[2]School of Electrical Engineering, Hanyang University ERICA

요 약

최근 텍스트를 입력으로 3D 객체를 생성하는 Text-to-3D 연구 분야에서 Score Distillation Sampling 기법이 제안되었다. 해당 기법은 3D 학습 데이터 없이도 3D 객체를 생성할 수 있는 장점이 있으나, 다양한 시점에서 객체의 기하학적 일관성을 보장하지 못한다는 한계가 있다. 이에 본 논문에서는 Dragging 이미지 편집 기법을 도입해 다중 시점의 기하학적 일관성을 개선한 새로운 Dragging Score Distillation 기법을 제안한다.

Ⅰ. 서 론

최근 Diffusion Model(DM)의 이미지, 텍스트, 영상 등 다양한 모달리티에서 성공적으로 활용되었으며 활발히 연구되고 있다. 하지만 DM을 학습하기 위해서는 방대한 양의 데이터가 필요하다는 단점이 있다.

이러한 배경에서 Score Distillation Sampling(SDS)[1]는 사전 학습된 Text-to-Image DM과 학습 가능한 3D 모델을 활용하여 3D 학습 데이터 없이 3D 객체를 생성하는 효율적인 기법으로 제안되었다. 그러나 SDS는 단일 시점 정보를 기반해 3D 모델을 최적화하는 방식이기 때문에, 3D 객체의 서로 다른 두 시점에서 똑같은 형상이 생성되어 기하학적 일관성이 떨어지는 Janus 문제가 발생한다.

본 논문에서는 이러한 문제를 해결하고자 Dragging 이미지 편집 기법과 SDS를 결합한 Dragging Score Distillation(DSD)을 제안한다. Dragging 이미지 편집 기법은 임의의 두 점을 활용하여 이미지 내 객체의 자연스러운 이동, 회전 등 다양한 편집을 수행한다. 본 논문에서는 해당 편집 기법을 통한 공간적 정보를 활용하여 다중 시점 간의 정합성을 확보하고, 3D 객체의 다양한 시점의 기하학적 일관성을 개선할 수 있다.

Ⅱ. 본론

Ⅱ-1 Score Distillation Sampling

SDS는 사전 학습된 Text-to-Image 2D DM을 활용해서 3D 객체를 생성한다. SDS는 $\boldsymbol{\epsilon}_\phi$는 사전 학습된 DM, $\mathbf{x} := g(\theta, \pi)$는 임의의 시점 π에서 렌더링된 이미지, θ는 3D 모델의 파라미터, $g(\cdot)$은 미분 가능한 렌더링 함수일 때,

$$\nabla_\theta L_{\text{SDS}} := \mathbb{E}_t \left[w(t) (\hat{\boldsymbol{\epsilon}}_\phi(\mathbf{z}_t, t, y) - \boldsymbol{\epsilon}) \frac{\partial \mathbf{x}}{\partial \theta} \right] \quad (1)$$

으로 정의된다. t는 DM의 time step, y는 텍스트, $w(t)$는 가중치 함수를 의미한다. 여기에서 $\mathbf{z}_t$는 $\mathbf{z}_\text{t} = \alpha_\text{t}\mathbf{x} + \sigma_\text{t}\boldsymbol{\epsilon}$으로 DM[2, 9]의 노이즈를 더하는 과정이다. $\boldsymbol{\epsilon} \sim \text{N}(\mathbf{0}, \mathbf{I})$는 노이즈를 의미한다. $\hat{\boldsymbol{\epsilon}}_\phi$는 Classifier-free Guidance(CFG)[3]로

$$\hat{\boldsymbol{\epsilon}}_\phi(\mathbf{z}_t, t, y) = \boldsymbol{\epsilon}_\phi(\mathbf{z}_t, t) + \omega(\boldsymbol{\epsilon}_\phi(\mathbf{z}_t, t, y) - \boldsymbol{\epsilon}_\phi(\mathbf{z}_t, t)) \quad (2)$$

으로 정의된다. ω는 CFG의 Guidance 가중치이다.

Ⅱ-2 Dragging 이미지 편집 알고리즘

Dragging은 두 점을 기반으로 수행하는 이미지 편집 기법이다. EasyDrag[4]는 이미지를 인코더와 DDIM Inversion[5]을 통해 잠재적 벡터로 변환한 후 두 점을 중심으로 한 일정 크기의 패치 사이의 코사인 유사도 $\varepsilon(\cdot)$를 활용해 잠재적 벡터를 최적화한다. DDIM Inversion은

$$\mathbf{z}_{\text{s}+\delta_\text{T}} = \frac{\alpha_{\text{s}+\delta_\text{T}}}{\alpha_\text{s}} \mathbf{z}_\text{s} + \left(\sigma_{\text{s}+\delta_\text{T}} - \frac{\alpha_{\text{s}+\delta_\text{T}}}{\alpha_\text{s}} \sigma_\text{s} \right) \boldsymbol{\epsilon}_\phi(\mathbf{z}_s, s) \quad (3)$$

으로 정의된다. $s = 0, \delta_T, 2\delta_T, \cdots, t - \delta_T$, $\mathbf{z}_0 = \mathbf{x}$이다. $\left\{ \mathbf{z}_{\delta_\text{T}}, \mathbf{z}_{2\delta_\text{T}}, \cdots, \mathbf{z}_\text{t} \right\}$의 잠재적 공간의 궤적을 예측하여 이미지의 잠재적 벡터를 얻을 수 있다. Dragging의 최적화 과정은 다음과 같이 정의된다.

$$\mathbf{z}_t^{i+1} \leftarrow \mathbf{z}_t^i - \eta \nabla_{\mathbf{z}_t} \varepsilon \quad (4)$$

코사인 유사도 $\varepsilon(\cdot)$는 각 잠재적 벡터를 DM $\boldsymbol{\epsilon}_\phi$를 통과시킨 U-Net 디코더의 특징맵 $\mathbf{F}$의 패치를 활용하여 계산한다. 디코더의 두 번째 및 세 번째 특징맵을 사용하여 코사인 유사도를 계산한다.

$$\varepsilon(\mathbf{F}, \mathbf{F}_0) = \frac{1}{1 + \frac{1 + \cos(\mathbf{F}, sg(\mathbf{F}_0))}{2}} \quad (5)$$

$\mathbf{F}$는 현재 최적화 대상인 $\mathbf{z}_t^i$의 특징맵이며, $\mathbf{F}_0$는 초기 $\mathbf{z}_t^0$의 특징맵이다. $sg(\cdot)$은 stopgrad, η는 Dragging의 학습률을 의미한다. 해당 최적화 과정은 N번 수행한다.

Ⅱ-3 Dragging Score Distillation

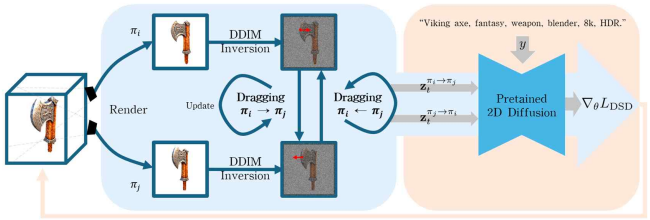

그림 1. Dragging Score Distillation 구조

본 논문에서 SDS와 Dragging을 결합한 Dragging Score Distillation(DSD)을 제안한다. 임의의 시점 π_i와 다른 시점 π_j를 샘플링 한다. 각 시점에서의 렌더링 이미지들은 DDIM Inversion을 통해 각각 $\mathbf{z}_t^{\pi_i}$와 $\mathbf{z}_t^{\pi_j}$를 얻는다. 해당 잠재적 벡터를 서로 다른 시점으로 Dragging 최적화 과정을 거친다. 해당 과정으로 최적화된 벡터를 각각 $\mathbf{z}_t^{\pi_i \to \pi_j}$와 $\mathbf{z}_t^{\pi_j \to \pi_i}$라고 하면, 해당 벡터들로 두 개의 SDS 식을 계산한다.

$$\nabla_\theta L_{\text{DSD}}^{\pi_i} = \mathbb{E}_t\left[w(t)(\hat{\boldsymbol{\epsilon}}_\phi(\mathbf{z}_t^{\pi_i \to \pi_j}, t, y_j) - \boldsymbol{\epsilon}_j)\frac{\partial \mathbf{x}^{\pi_i}}{\partial \theta}\right] \quad (6)$$

$$\nabla_\theta L_{\text{DSD}}^{\pi_j} = \mathbb{E}_t\left[w(t)(\hat{\boldsymbol{\epsilon}}_\phi(\mathbf{z}_t^{\pi_j \to \pi_i}, t, y_i) - \boldsymbol{\epsilon}_i)\frac{\partial \mathbf{x}^{\pi_j}}{\partial \theta}\right] \quad (7)$$

최종적인 DSD는 $\nabla_\theta L_{\text{DSD}} = \nabla_\theta\left(L_{\text{DSD}}^{\pi_i} + L_{\text{DSD}}^{\pi_j}\right)$으로 정의된다. $\left\{\mathbf{x}^{\pi_i}, y_i, \boldsymbol{\epsilon}_i\right\}$와 $\left\{\mathbf{x}^{\pi_j}, y_j, \boldsymbol{\epsilon}_j\right\}$는 각 샘플링 된 시점에 따른 렌더링된 이미지, 텍스트, 노이즈다. 각 업데이트된 잠재적 벡터들은 서로 다른 시점에서 샘플링된 노이즈를 매칭하는 식으로 구성된다.

두 시점의 위치는 큰 차이가 없도록 샘플링한다. 두 시점의 차이가 클수록 Dragging 최적화 과정의 부정확성이 커지기 때문이다. 특징점 매칭을 통해 두 점을 구하지 못한 경우는 SDS로 학습한다.

Ⅲ. 실험

DSD는 사전 학습된 Text-to-Image 2D DM은 Stable Diffusion v2.1[6], 3D 모델은 3D Gaussian Splatting(3D-GS)[7]을 사용했다. SDS와 DSD 두 모델의 CFG 가중치는 100으로 진행했다. DSD의 Dragging 최적화 과정 횟수는 총 5회, 학습률은 0.01로 설정했다. DSD의 두 이미지 시점은 azimuth [-30°, 30°] 및 polar [-15°, 15°] 구간의 랜덤한 값 차이를 갖도록 샘플링했다. Dragging의 패치 사이즈는 6x6 크기로 설정했다. RTX 3090 1개 환경에서 진행했다. SDS의 배치 크기는 4로 설정했으며 DSD의 배치 크기는 2로 설정했다.

실험은 총 2가지 텍스트에 대해서 진행했다. SDS는 한 객체 생성에 약 50분이 소요되었으며 DSD는 한 객체 생성에 약 6시간이 소요되었다. 그림 2에서 첫 줄은 SDS, 두 번째 줄은 DSD 결과다. DSD는 SDS에 비해 구조적으로 안정된 모습의 결과를 보여준다. 그림 2 상단 결과에서 DSD는 SDS에 비해 더 좋은 퀄리티의 생성 결과를 보여준다. 다만, 그림 2 하단 결과의 0°와 180°에서 생성된 모습은 SDS 및 DSD 모두 서로 일관성 없는 결과를 보여준다.

Ⅳ. 결론

Dragging 이미지 편집 기법을 SDS에 접목하여 여러 시점의 기하학적 일관성에 대해 좋은 영향을 미치고 양질의 결과를 얻을 수 있다. 그럼에도 DSD의 잠재적 벡터 최적화 과정, Dragging 기법의 부정확성 등 개선할 수 있는 여지가 존재한다. 추후 해당 기법을 고도화하여 보다 양질의 객체를 생성할 수 있을 것으로 기대된다.

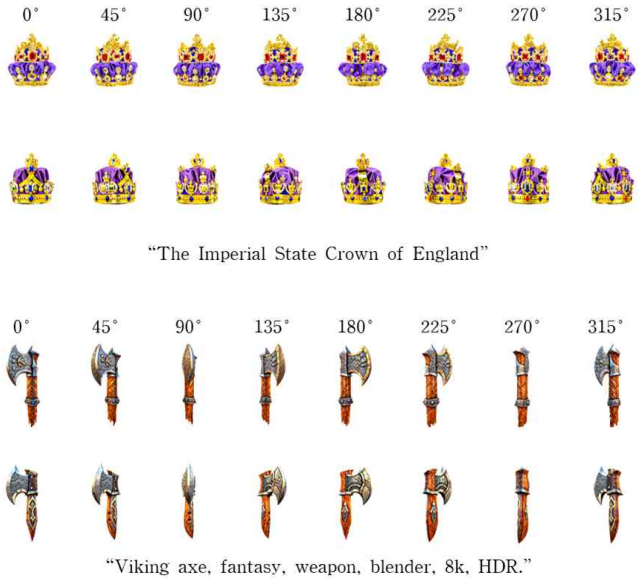

그림 2. 실험 결과

ACKNOWLEDGMENT

이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원-지역지능화혁신인재양성사업의 지원을 받아 수행된 연구임 (IITP-2026-RS-2020-II201741)

참 고 문 헌

[1] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhal. DreamFusion: Text-to-3D using 2D Diffusion. In International Conference on Learning Representations, 2023.

[3] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. In 35th Conference on Neural Information Processing Systems Workshop DGMs Applications, 2021.

[4] Xingzhong Hou, Boxiao Liu, Yi Zhang, Jihao Liu, Yu Liu, and Haihang You. EasyDrag: Efficient Point-based Manipulation on Diffusion Models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024.

[5] Prafulla Dhariwal and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. In Conference on Neural Information Processing Systems, 2021.

[6] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-Resolution Image Synthesis With Latent Diffusion Models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 10684-10695, 2022.

[7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuhler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. In ACM Transactions on Graphics, 2023.