Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware

RSS 2023

양현서

July 12, 2024

About

- Author: Tony Z. Zhao, Vikash Kumar, Sergey Levine, Chelsea Finn

- Conference: RSS 2023

Motivation

- Diffusion models well generate high-quality **rasterized** images

Motivation

- Diffusion models well generate high-quality **rasterized** images

- However, designers also vastly use **vectorized** images in practice like SVG (Scalable Vector Graphics)

Motivation

- Diffusion models well generate high-quality **rasterized** images

- However, designers also vastly use **vectorized** images in practice like SVG (Scalable Vector Graphics)

- Training diffusion model to generate vectorized images is theoritically possible but practically challenging

Challenges of training diffusion models for vectorized images

- Most labeled datasets are for rasterized images

Challenges of training diffusion models for vectorized images

- Most labeled datasets are for rasterized images

- Vectorized images’ data structure is more complex (hierarchical and variable-lengthed) than rasterized images’

Baselines

1. Generate a rasterized image with a pretrained diffusion model

Baselines

1. Generate a rasterized image with a pretrained diffusion model
2. Convert the rasterized image to a vectorized image using a vectorization algorithm: **LIVE algorithm**

Baselines

1. Generate a rasterized image with a pretrained diffusion model
2. Convert the rasterized image to a vectorized image using a vectorization algorithm: **LIVE algorithm**

Challenges

- Diffusion Models often generate **too complex** rasterized images to be vectorized
- Automated conversion **loses details**

VectorFusion

- Differentiable vector graphics renderer

VectorFusion

- Differentiable vector graphics renderer

- Score Distillation Sampling

VectorFusion

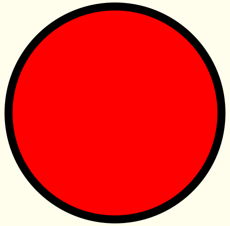
- Differentiable vector graphics renderer
- Score Distillation Sampling
- SVG-Specific regularization

Background: Vector representation and rendering pipeline

```
<svg width="200" height="200" xmlns="http://www.w3.org/2000/svg">
  <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />
</svg>
```

Background: Vector representation and rendering pipeline

```
  <svg width="200" height="200" xmlns="http://www.w3.org/2000/svg">
    <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />
  </svg>
```



Background: SVG Path

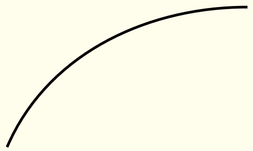
- **`M x,y`: Move to the point (x, y)**
- `L x,y`: Draw a line to the point (x, y)
- `H x`: Draw a horizontal line to x
- `V y`: Draw a vertical line to y
- **`C x1, y1, x2, y2, x, y`: Draw a cubic Bezier curve**
- `Q x1, y1, x, y`: Draw a quadratic Bezier curve
- **`Z`: Close the path**

Background: SVG Path

```
  <svg width="200" height="200" xmlns="http://www.w3.org/
2000/svg">
    <path d="M10 80 C 40 10, 160 10, 190 80" stroke="black"
fill="transparent"/>
  </svg>
```

Background: SVG Path

```
  <svg width="200" height="200" xmlns="http://www.w3.org/2000/svg">
    <path d="M10 80 C 40 10, 160 10, 190 80" stroke="black" fill="transparent"/>
  </svg>
```



Vector Grapchic Parameterization

- s: number of segments
- n: number of paths to add

Background: Diffusion Models Training

- $\mathcal{L}_{\text{DDPM}}(\phi, \mathtt{x}) = \mathbb{E}_{t,\epsilon}\left[w(t)\|\epsilon_\phi(\alpha_t \mathtt{x} + \sigma_t \epsilon) - \epsilon\|_2^2\right]$ [1]

[1] $\alpha_t = \sqrt{1-\beta_t}$, $\sigma_t = \sqrt{\beta_t}$

Background: Diffusion Models Training

- $\mathcal{L}_{\text{DDPM}}(\phi, \mathbf{x}) = \mathbb{E}_{t,\epsilon}\left[w(t)\|\epsilon_\phi(\alpha_t \mathbf{x} + \sigma_t \epsilon) - \epsilon\|_2^2\right]$[2]
- $w(t)$: Sampling weight (usually evenly set to 1)

---

[2] $\alpha_t = \sqrt{1 - \beta_t}$, $\sigma_t = \sqrt{\beta_t}$

Background: Diffusion Models Training

- $\mathcal{L}_{\mathrm{DDPM}}(\phi, \mathbf{x}) = \mathbb{E}_{t,\epsilon}\left[w(t)\|\epsilon_\phi(\alpha_t \mathbf{x} + \sigma_t \epsilon) - \epsilon\|_2^2\right]$ [3]
- $w(t)$: Sampling weight (usually evenly set to 1)
- ϵ_ϕ: Noise Prediction Model

---

[3] $\alpha_t = \sqrt{1-\beta_t}$, $\sigma_t = \sqrt{\beta_t}$

Background: Diffusion Models Training

- $\mathcal{L}_{\mathrm{DDPM}}(\phi, \mathbf{x}) = \mathbb{E}_{t,\epsilon}\left[w(t)\|\epsilon_\phi(\alpha_t \mathbf{x} + \sigma_t \epsilon) - \epsilon\|_2^2\right]$ [4]
- $w(t)$: Sampling weight (usually evenly set to 1)
- ϵ_ϕ: Noise Prediction Model
- $\mathbf{x}$: Original data sample

[4] $\alpha_t = \sqrt{1-\beta_t}$, $\sigma_t = \sqrt{\beta_t}$

Background: Diffusion Models Training

- $\mathcal{L}_{\text{DDPM}}(\phi, \mathbf{x}) = \mathbb{E}_{t,\epsilon}\left[w(t)\|\epsilon_\phi(\alpha_t \mathbf{x} + \sigma_t \epsilon) - \epsilon\|_2^2\right]$ [5]
- $w(t)$: Sampling weight (usually evenly set to 1)
- ϵ_ϕ: Noise Prediction Model
- $\mathbf{x}$: Original data sample
- ϵ: Random (Gaussian) noise

---

[5] $\alpha_t = \sqrt{1-\beta_t}$, $\sigma_t = \sqrt{\beta_t}$

Background: Diffusion Models Training

- $\mathcal{L}_{\text{DDPM}}(\phi, \mathbf{x}) = \mathbb{E}_{t,\epsilon}\left[w(t)\|\epsilon_\phi(\alpha_t\mathbf{x} + \sigma_t\epsilon) - \epsilon\|_2^2\right]$ [6]
- $w(t)$: Sampling weight (usually evenly set to 1)
- ϵ_ϕ: Noise Prediction Model
- $\mathbf{x}$: Original data sample
- ϵ: Random (Gaussian) noise
- α_t, σ_t: Propotion of $\mathbf{x}$ and ϵ

---

[6] $\alpha_t = \sqrt{1-\beta_t}$, $\sigma_t = \sqrt{\beta_t}$

Background: Diffusion Models Sampling

1. Sample $\mathtt{x}_t$ from a prior.
2. Predict noise $\epsilon_\phi(\mathtt{x}_t)$.
3. Compute $\hat{\mathtt{x}}$.

   $\hat{\mathtt{x}} = \frac{\mathtt{x}_t - \sigma_t \epsilon_\phi(\mathtt{x}_t)}{\alpha_t}$

4. Compute $\mathtt{x}_{t-1}$ and feed to the next step.

   $\mathtt{x}_{t-1} = \alpha_{t-1}\hat{\mathtt{x}} + \sigma_{t-1}\epsilon_\phi(\mathtt{x}_t)$

Background: Classifier Free Guidance

$$\hat{\epsilon}_\phi(\mathtt{x}, y) = (1 + \omega) * \epsilon_\phi(\mathtt{x}, y) - \omega * \epsilon_\phi(\mathtt{x})$$

- Generate both conditional and unconditional prediction for the noise. $\epsilon_\phi(\mathtt{x}, y)$ and $\epsilon_\phi(\mathtt{x})$

Background: Classifier Free Guidance

$$\hat{\epsilon}_\phi(\mathtt{x}, y) = (1 + \omega) * \epsilon_\phi(\mathtt{x}, y) - \omega * \epsilon_\phi(\mathtt{x})$$

- Generate both conditional and unconditional prediction for the noise. $\epsilon_\phi(\mathtt{x}, y)$ and $\epsilon_\phi(\mathtt{x})$

- Combine them with a weight ω

Background: Classifier Free Guidance

$$\hat{\epsilon}_{\phi}(\mathtt{x}, y) = (1 + \omega) * \epsilon_{\phi}(\mathtt{x}, y) - \omega * \epsilon_{\phi}(\mathtt{x})$$

- Generate both conditional and unconditional prediction for the noise. $\epsilon_{\phi}(\mathtt{x}, y)$ and $\epsilon_{\phi}(\mathtt{x})$

- Combine them with a weight ω

- This enhances the model's ability to generate images with the desired class.

Score Distillation Sampling

$$\mathcal{L}_{\mathrm{SDS}} = \mathbb{E}_{t,\epsilon}\left[\frac{\sigma_t}{\alpha_t} w(t)\,\mathrm{KL}\Big(q(\mathtt{x}_t|g(\theta);y,t)\|\; p_\phi(\mathtt{x}_t;y,t)\Big)\right]$$

Score Distillation Sampling

$$\mathcal{L}_{\text{SDS}} = \mathbb{E}_{t,\epsilon}\left[\frac{\sigma_t}{\alpha_t} w(t)\, \text{KL}\big(q(\mathtt{x}_t | g(\theta); y, t) \| \, p_\phi(\mathtt{x}_t; y, t)\big)\right]$$

- p_ϕ is a **frozen** noise prediction model
- q is a unimodal gaussian centered at a learned image $g(\theta)$
- The larger $\frac{\sigma_t}{\alpha_t}$, the more the model prediction is important

Score Distillation Sampling

$$\mathcal{L}_{\mathrm{SDS}} = \mathbb{E}_{t,\epsilon}\left[\frac{\sigma_t}{\alpha_t} w(t)\,\mathrm{KL}\big(q(\mathtt{x}_t | g(\theta); y, t) \|\, p_\phi(\mathtt{x}_t; y, t)\big)\right]$$

- p_ϕ is a **frozen** noise prediction model
- q is a unimodal gaussian centered at a learned image $g(\theta)$
- The larger $\frac{\sigma_t}{\alpha_t}$, the more the model prediction is important

$$\nabla_\theta \mathcal{L}_{\mathrm{SDS}} = \mathbb{E}_{t,\epsilon}\left[w(t)(\hat{\epsilon}_t(\mathtt{x}_t; y, t) - \epsilon)\frac{\partial \mathtt{x}}{\partial \theta}\right]$$

Method

Baseline

1. Sample an image using a pretrained diffusion model.

Baseline

1. Sample an image using a pretrained diffusion model.
2. Automatically convert the image to an SVG using the LIVE algorithm.

Baseline

1. Sample an image using a pretrained diffusion model.
2. Automatically convert the image to an SVG using the LIVE algorithm.

Provide "minimal flat 2d vector icon. lineal color. on a white background. trending on artstation" as a prompt.

Baseline

1. Sample an image using a pretrained diffusion model.
2. Automatically convert the image to an SVG using the LIVE algorithm.

Provide “minimal flat 2d vector icon. lineal color. on a white background. trending on artstation” as a prompt.

Rejection Sampling

Sample $K = 4$ images, select best by CLIP ViT-B/16 score

VectorFusion

1. Initialize a parameter $\{p_1, p_2, ..., p_k\}$

VectorFusion

1. Initialize a parameter $\{p_1, p_2, ..., p_k\}$
2. Repeat the following:

VectorFusion

1. Initialize a parameter $\{p_1, p_2, ..., p_k\}$
2. Repeat the following:
   1. DiffVg renders a 600 × 600 image using the parameters

VectorFusion

1. Initialize a parameter $\{p_1, p_2, ..., p_k\}$
2. Repeat the following:
   1. DiffVg renders a 600 × 600 image using the parameters
   2. Augment the image (perspective transformation, 512 × 512 crop)

VectorFusion

1. Initialize a parameter $\{p_1, p_2, ..., p_k\}$
2. Repeat the following:
   1. DiffVg renders a 600 × 600 image using the parameters
   2. Augment the image (perspective transformation, 512 × 512 crop)
   3. Map the image to the latent space

VectorFusion

1. Initialize a parameter $\{p_1, p_2, ..., p_k\}$
2. Repeat the following:
    1. DiffVg renders a 600 × 600 image using the parameters
    2. Augment the image (perspective transformation, 512 × 512 crop)
    3. Map the image to the latent space
    4. Compute $\mathbf{z}_t = \alpha_t \mathbf{z} + \sigma_t \epsilon$

VectorFusion

1. Initialize a parameter $\{p_1, p_2, ..., p_k\}$
2. Repeat the following:
   1. DiffVg renders a 600 × 600 image using the parameters
   2. Augment the image (perspective transformation, 512 × 512 crop)
   3. Map the image to the latent space
   4. Compute $\mathbf{z}_t = \alpha_t \mathbf{z} + \sigma_t \epsilon$
   5. Remove the noise using the noise prediction model

VectorFusion

1. Initialize a parameter $\{p_1, p_2, ..., p_k\}$
2. Repeat the following:
   1. DiffVg renders a 600 × 600 image using the parameters
   2. Augment the image (perspective transformation, 512 × 512 crop)
   3. Map the image to the latent space
   4. Compute $\mathbf{z}_t = \alpha_t \mathbf{z} + \sigma_t \epsilon$
   5. Remove the noise using the noise prediction model
   6. Update the parameters using SDS loss

VectorFusion: Details

- Sample t from $t \sim \mathrm{Uniform}(50, 950)$
- Use fp16 precision
  - Use fp32 precision for $\frac{\partial \mathtt{z}}{\partial \mathtt{x}_{\mathrm{aug}}}$ for stability
- Apply self intersection regularizer loss also[7]

$$\mathcal{L}_{\mathrm{Xing}} = D_1(\mathrm{ReLU}(-D_2) + (1 - D_1)(\mathrm{ReLU}(D_2)))$$

- Reinitialize low opacity or shrinked paths

[7] $D_1 = \mathbb{I}\left(\overrightarrow{AB} \cdot \overrightarrow{BC}\right)$, $D_2 = \frac{\overrightarrow{AB} \cdot \overrightarrow{CD}}{\|\overrightarrow{AB}\|\|\overrightarrow{CD}\|}$

VectorFusion: Architecture

Evaluation

No test dataset available for vector graphics

Therefore, use CLIP as an evaluation metric

Evaluation

No test dataset available for vector graphics

Therefore, use CLIP as an evaluation metric

R-Precision

For 128 SVGs, assign the captions by the CLIP score and check if it was the correct caption. Get the fraction of correct captions.

Evaluation

No test dataset available for vector graphics

Therefore, use CLIP as an evaluation metric

R-Precision

For 128 SVGs, assign the captions by the CLIP score and check if it was the correct caption. Get the fraction of correct captions.

- Used rejection sampling (K in the table)

Evaluation

- CLIPDraw performed best, but **qualitative results are poor**
- Therefore, also use OpenCLIP score
- Effect of rejection sampling $\sim$ caption consistency

Qualitative Results

Discussion

- Utilizes pretrained diffusion models **without captioned SVG datasets**
- Effectively shows the **distillation of generative models** compared to contrastive models

Discussion

- Utilizes pretrained diffusion models **without captioned SVG datasets**
- Effectively shows the **distillation of generative models** compared to contrastive models

- Computationally more **expensive** than CLIP-based approaches
- **Limited** by the quality and biases of the pretrained diffusion model

References

- DiffVG
- LIVE
- VectorFusion