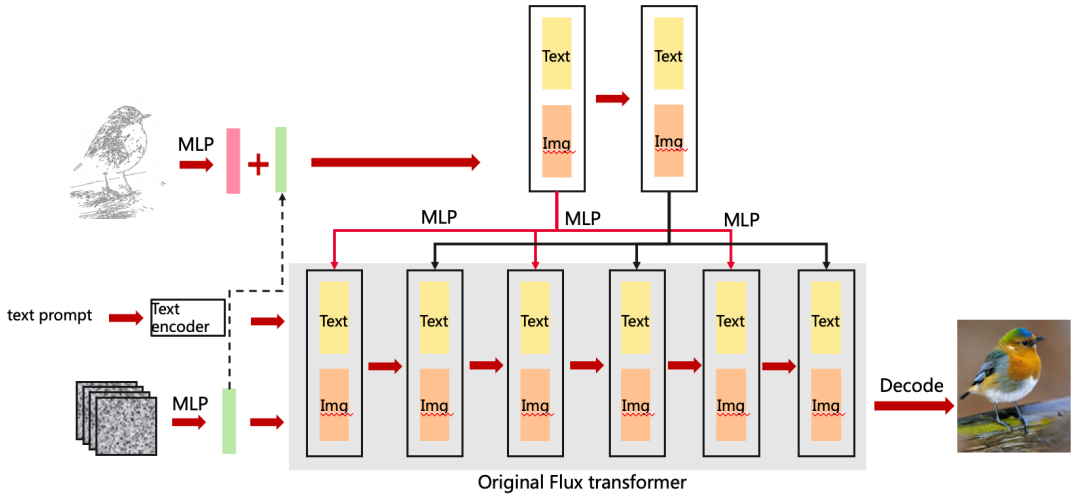
CannyEdit: Training-Free Image Editing through Canny Control in Pretrained Diffusion Models
Overleaf: https://www.overleaf.com/8332121776jghbzrynwfkw#a473dc

Memo 2503

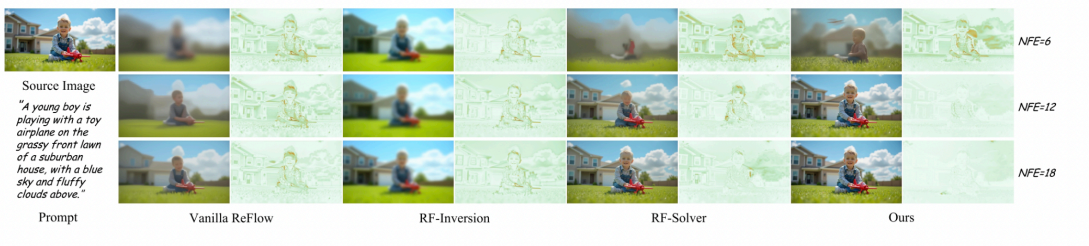
A. Background:

1. Flux + Canny ControlNet provides a tool to generate high-quality images with the desired general layout.



More examples related to layout maintenance.

2. FireFlow provides a tool to inverse a real-world image into latent space.



B. Our key intuition

To address two trade-offs:
1. Local editability vs. Background consistency
2. Local controllability vs. Global affordance

It has been observed in KV-edit that editing on the local region would also bring changes to the background. This affects trade-off 1. At the same time, when the background is changed a lot during the denoising with editing, the global affordance of the added object could also be affected. This affects trade-off 2.

To make a good balance on the two trade-offs, we propose an editing method based on the canny layout control via ControlNet. To maintain the background consistency, we apply the canny control on the background to maintain the layout of the background unchanged. To allow the local editability, the related region in the canny control is revised to be zero. In the later denoising steps, blending operation is applied to ensure the visual details of the background is unchanged on top of the maintained layout preserved by the ControlNet. Note that the unchanged background layout also makes sure that the generated object is semantically fitted with the original environment well if the objects can be generated.

To enable a good balance of local controllability and global affordance, we apply regional control with attention mask where local editing text prompts is cross-attended with the edited image region, and a context prompt which describe the image after editing is cross-attended with the whole image.

C. Our methods

Given a series of N discrete timesteps t = {t_n , ..., t_i, … t_0},
$Z_{i-1} = Z_i + (t_i - t_{i-1})X_i$.

1. **Apply FireFlow to obtain the inverted latent Z_i at each denoising step.**

Different from the original FireFlow, we add the Canny ControlNet during the inversion.

(Note that we denote the inversed latent as Z_i, and the latent during the editing denoising process as Z'_i.)

2. **Starting point of the editing:**

$$t_{start} = t_{n-l}$$

$Z'_{start}[BG] = Z_{n-l}[BG]$, for pixels in the background;
$Z'_{start}[Edit] = \epsilon$, for pixels in the edited region.

We have observed that the random noise ϵ would affect the editing results. To stabilize the effect, we sample the noise 20000 times and pick the one that results in the highest cosine similarity $Cos\left(Z'_{start}[Edit],\ Z_{n-l}[Edit]\right)$.

The starting point to allow the local editability while also preserving some information in the edited region.

3. **Canny control via ControlNet**

The ControlNet does:

$$Control_t = ControlNet(Z_{t-1},\ txt,\ canny)$$

The $Control_t$ is added into the denoising process of Z_t:

To maintain the background consistency and allow the local editability, we modify the $Control_t$ to $Control'_t$ via:
$Control'_t[BG] = Control_t[BG]$, and
$Control'_t[Edit] = \mathbf{0}$.

The $Control'_t$ is added into the denoising process of Z'_t.

4. **Regional Control via Attention Mask**

Context prompt: txt_{ctxt}
Environment prompt: txt_{env}
Editing prompt: txt_{edit}

Attention mask:

	txt_{ctxt}	txt_{env}	txt_{edit}	img_{env}	img_{edit}
txt_{ctxt}	1	0	0	1	1
txt_{env}	0	1	0	1	0
txt_{edit}	0	0	1	0	1
img_{env}	1	1	0	1	1
img_{edit}	1	0	1	1	1

Besides the hard attention masks, we can also apply re-weighting to amplify some attentions. For instance, we can amplify the attention: $\alpha \bullet Norm(Q_{img_{edit}} \bullet K_{txt_{edit}}) \bullet V_{img_{edit}}$, where $\alpha > 1$.

5. **Classifier-Free Guidance w/ Regional Control**

The CFG in regional control is little to be discussed.
Here, we do:
With negative prompt applied to the editing region: txt_{nedit},
We obtain $X_i^{'neg}$

	txt_{nedit}	img_{env}	img_{edit}
txt_{nedit}	1	0	1

img_{env}	0	1	1
img_{edit}	1	1	1

The CFG is only applied to the region:
$X'_i[Edit] = (1+w) \bullet X'_i[Edit] - w \bullet X_i^{'neg}[Edit]$, w is the CFG strength.
$Z_{i-1} = Z_i + (t_i - t_{i-1})X_i$.

Two good things of CFG:
1. Apply stronger guidance to the edited region; 2. Avoid the undesired features
One bad thing: may enforce more potential cartoon style.

6. Preservation of original information in the edited region

Although the starting point $Z'_{start}[Edit] = (1-t_{syn})*Z_0[Edit] + t_{syn}*\epsilon[Edit]$ contains information of the edited region in the original image, the information could be discarded. This information could be critical to ensure the affordance of the added object in some cases.

In order to better preserve the original information in the edited region, we may add the canny control information of the original edited region in some layers at some denoising steps. However, the question is that which layers and steps to add?
Intuitively, the insertion step should not be too late. If it is too late, it may result in artifacts.
For the layer, should we do the insertion in the layers that were identified by StableFlow as most related to the image layout?

Step 7 and Step 12
index_block<=1 or index_block>17
img = img + (original controlnet info + new controlnet info)
Original controlnet info

$$Control_t = ControlNet(Z_{t-1}, txt, canny)$$

New controlnet info

$$Control_t = ControlNet(Z'_{t-1}, txt, canny)$$

(Style normalization)
We may check which layer has strong effect to the style.
Apply ControlNet in the inversion. During denoising, don't add control to specific layer, check which layer causes the significant style change.
Check style similarities: https://github.com/learn2phoenix/CSD/

(Blending to further maintain background consistency)
(Incomplete/extra body parts➔ 6.)`
(Shape issue)

D. Extension to other editing tasks

E. Combination with other tools to provide different local control tools

==========

Mask acquisition of the generated objects

To obtain the mask of generated object for blending, we first apply the similar method as in Add-it.

- The extraction of five-point prompts

We extract the attention weights ($Q_{img_{edit}} \bullet K_{txt_{edit}}$) from the layers ["transformer blocks.13","transformer blocks.14", "transformer blocks.18", "single transformer blocks.23", "single transformer blocks.33"] of the middle and three-fourths denoising steps.

The attention weights averaged over the layers and the denoising steps are used to extract the top-5 salient point on the img_{edit}.
The top-5 points are used as prompts fitted to the SAM-2 model to get a mask m_{sam}.

However, we found the SAM-2 with the identified top-5 salient points are not always able to locate the generated objects first. In order to improve stability of the mask acquisition, we apply the language SAM (w/ the label of the generated object) to get a candidate set of masks $\mathbb{M}_{lang-sam}$. The one in $\mathbb{M}_{lang-sam}$ with the highest IoU with m_{sam} is identified as the final mask m.

==========

Stage 2 Technical points

Input:

- the canny map of the composite image $canny'$
- the canny map of the original image $canny$
- The refined mask of the added object m obtained from stage 1
- the inversion results Z_i at each denoising step i of the original image
(- the predicted latent Z'_i from stage 1 at each denoising step i)

1. Find the best fitted ellipse mask m_e induced from the given mask m

To make sure the good affordance when the mask is not perfect, we did not use the given mask directly. Instead we use the best fitted ellipse mask induced from the given mask.

```
Input:
  m ∈ {0,1}^{H×W}  ⟶ Binary mask (foreground=1, background=0)

Output:
  m_e ∈ {0,1}^{H×W}  ⟶ Ellipse-filled binary mask

1. Binarize the mask:
  m ← (m > 0) ∈ {0,1}^{H×W}

2. Morphological closing to merge small gaps:
  K ← Elliptical kernel of size (10,10)
  m_closed ← MorphClose(m, kernel=K)

3. Extract contours:
  C ← FindContours(m_closed)
  if |C| = 0:
    raise Error("No contours found")

4. Combine all contours into a point set:
  P ← ∪ C_i ∈ C

5. Check if sufficient points:
  if |P| < 5:
    raise Error("Too few points to fit ellipse")

6. Fit ellipse to point set:
  ⟨c, (a,b), θ⟩ ← FitEllipse(P)

  where:
    - c = (x₀, y₀): ellipse center
    - a, b: major and minor axes lengths
    - θ: rotation angle in degrees
    - FitEllipse(P) solves for best (least-squares) fit of general conic:
        Ax² + Bxy + Cy² + Dx + Ey + F = 0
      under constraint B² - 4AC < 0 (ellipse condition)

7. Optional shape adjustment (to reduce sharpness):
  if min(a,b) / max(a,b) < 0.2:
    if a < b:
      a ← 0.2·b
    else:
      b ← 0.2·a

8. Draw filled ellipse on empty mask:
  m_e ← zeros_like(m)
  m_e ← DrawEllipse(m_e, center=c, axes=(a,b), angle=θ, color=1, thickness=−1)

  where:
    - DrawEllipse(...) rasterizes the parametric ellipse:
        (x(t), y(t)) = (x₀ + a·cos(t)·cos(θ) − b·sin(t)·sin(θ),
                        y₀ + a·cos(t)·sin(θ) + b·sin(t)·cos(θ))
      for t ∈ [0, 2π]

Return:
  m_e
```

2. Starting point of the editing:

$t_{start} = t_{n-l'}$, l' depends on how much stage-1 information would like to preserve in the edited region
$Z''_{start}[BG] = Z_{start}[BG]$ for pixels in the background;
$Z''_{start}[Edit] = Z_{start}[Edit]$, for pixels in the edited region.

$Z''_{start}[Edit] = Z'_{start}[Edit]$

OR

$Z''_{start}[Edit] = \epsilon$ if do not want information for the stage 1

Need to consider how to preserve information from the stage 1 or from the given human image during the denoising to maintain the IP.

3. Canny Control:

The ControlNet does:

$$Control_t = ControlNet(Z_{t-1}, txt, canny)$$

The $Control_t$ is added into the denoising process of Z_t.

We now do:

$$Control'_t = ControlNet(Z_{t-1}, txt, canny'),$$

in the beginning k denoising steps.

To allow better affordance, in the later denoising steps, we do not apply canny control on the edited region and make it be able to blend in better with the context, where
$Control'_t[BG] = Control_t[BG]$, and
$Control'_t[Edit] = \mathbf{0}$.

Also, the weaker control strength is applied to allow the generation of collateral objects, like shadows, the racket held by the player…

4. The cyclical blending

To allow the generation of collateral objects but maintain the background mainly unchanged, we do the cyclical blending with partial blending on the background region with the inversion latent.

For every 5 denoising steps, we do the blending as below:
$Z''_t[BG] = 0.5 * Z_t[BG] + 0.5 * Z''_t[BG]$

5. Regional Control via Attention Mask

To maintain the style consistency between the edit region and the background region, we amplify the attention:
$\alpha \cdot Norm(Q_{img_{edit}} \cdot K_{img_{edit}}) \cdot V_{img_{edit}}$, where $\alpha > 1$.

Attention mask:

	txt_{ctxt}	txt_{env}	txt_{edit}	img_{env}	img_{edit}
txt_{ctxt}	1	0	0	1	1
txt_{env}	0	1	0	1	0
txt_{edit}	0	0	1	0	1
img_{env}	1	1	0	1	1
img_{edit}	1	0	1	1	1

Could reconsider this part of attention mask in Stage 2

==========

Evaluation and Ablation Study

Ablation Study:

- Regional control
(Stage 1 and) Stage 2: only local prompt or only context prompt
Attention weight amplification: Stage 2 the $\alpha > 1$ to maintain style consistency

- Canny control
Stage 2 - Alternatives: use the canny' throughout the whole process

- Blending
Full blending throughout the whole process

- The mask segmentation
Use only points from attention weights+SAM

(- The regional CFG in stage 1)

Evaluation:

Dataset:

Cover the tasks: add, replace, remove
Better to include the existing datasets from previous works (some label annotations may need);
To build a dataset with complex interactions w/ context and is difficult to maintain good affordance;
Real-world cases (to showcase previous method could fail to maintain style consistency).
Test cases of multiple adding at once.
(Test cases of IP preservation)

Qn metrics:
text faithfulness,
background consistency
Some metrics designed for deletion, refer to those in https://arxiv.org/pdf/2503.08677

Ql- human study

Ask if human can identify any objects in the image are created by inpainting OR
Ask if human can identify the difference between a GT image and an inpainting image

Baselines:
Compare w/ Open-source: Rectified Flow, Fireflow, KVEdit, Add-it, PowerPaint-SD, PowerPaint-Flux, Flux-inpaint, https://github.com/alimama-creative/FLUX-Controlnet-Inpainting

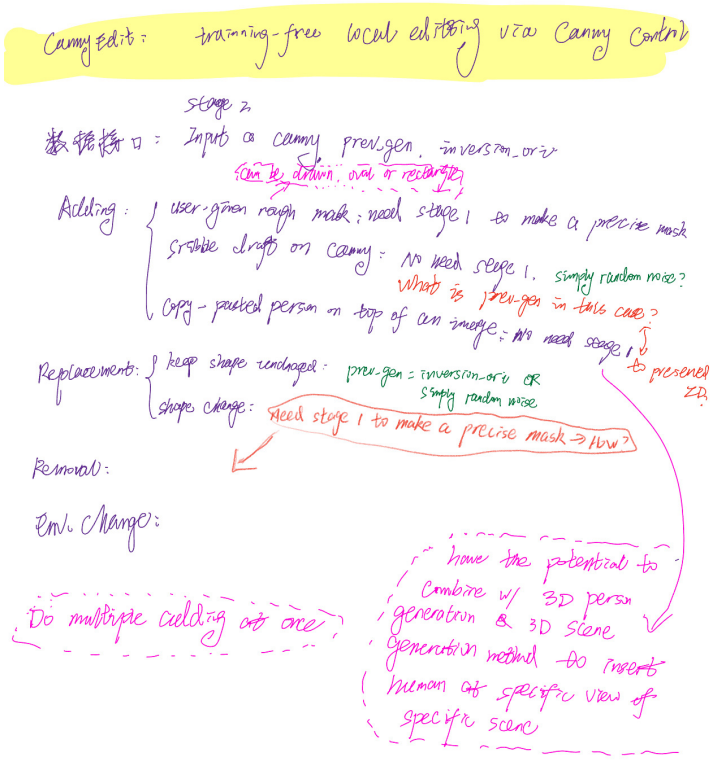
Some trained methods: OmniPaint, BrushEdit

Compare w/ Close-source: Gemini2-experimental (it is especially bad when doing two or more editings at one round), GPT-4

Update 0411

Next to do (0410):
1. Organize the current code and save a backup. [DONE]
2. Summary the technical points (ablation study). [DONE]
3. Design how to extend to different tasks, multiple edits, IP control. [DONE]
4. Start a writing on the overleaf.

==========



Draw a canny draft,
Give a region --> need stage 1
Copy-paste

==>

Replacement (no shape change)
Replacement (shape change) --> need stage 1
Removal

Blending:

When we have a Canny map
+ Region control to ensure the specific object is generated at specific location

We can generate the images with desired objects generated at specifically given location when making

+ Inversion and Blending

We can make sure the background part of the image is unchanged.

Question is how to get the Canny map: human drawing is one source, but how could we do it automatically?

==========

Our Technical Contributions:

1. Regional control; 2. Canny control; 3. Synchronized denoising; 4. Diff extraction.

==========

Extension:

0. +Add, Replace, Remove (task in different frequencies?), Environment Change

1. Multiple instances: adding multiple at the same time;

2. Gemini, GPT-4o cannot do, our method can. Let Gemini, GPT-4o generate some images first, and show that they cannot edit specific regions (when there are multiple humans in the image), but our method can

侧面“小”人物的修改，替换，一次性编辑多处，多次添加，指定区域添加“大”物体?

** Support different masks, like user-drawn mask, oval mask, rectangle mask...

** Support user-drawn canny, user-given person copied from other images (3D person generation)

** +IP Adapter (face replacement?)/Pose Control (incorporate it into the Canny?)/Depth control;

** + ConsiStory/ 3D scene generation model;

** + Image to video generation/editing

** Extend to an adapter w/ generation subject ratio?

** Try different DiT models

** Try AR+ControlNet models

==========