Beyond Flat Text: Dual Self-inherited Guidance for Visual Text Generation

Supplementary Material

7. Additional Background

Latent Diffusion Models. The Latent Diffusion Models (LDM) denoise a noisy vector z_t to z_0 that is mapped to an image x_0 through an autoencoder, based on a text prompt y. To sequentially remove the noise ε, a network ε_θ is trained to minimize the loss:

$$\mathcal{L} = \mathbb{E}_{z_0, y, \varepsilon \sim \mathcal{N}(0,1), t} ||\varepsilon - \varepsilon_\theta(z_t, t, c(y))||_2^2, \tag{8}$$

where the $c(y)$ is the conditioning embedding of the prompt y, z_t is a noisy vector obtained by adding noise to z_0 according to the timestep t. During inference, provided a random noise vector z_T, the trained network iteratively removes the predicted noise to produce a latent z_0 for T steps. Namely, we employ popular DDIM sampling [32] for inference:

$$\begin{aligned} z_{t-1} =& \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{z_t - \sqrt{1-\alpha_t}\epsilon_\theta(z_t)}{\sqrt{\alpha_t}}\right)}_{\text{predicted } z_0} \\ &+ \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta(z_t). \end{aligned} \tag{9}$$

The DDIM Sampling in general combines the predicted z_0 and z_t to obtain the z_{t-1}. We mainly analyze the predicted x_0 that is decoded from predicted z_0 in the rest of the section.

Adaptive Instance Normalization. The Adaptive Instance Normalization (AdaIN) [12] is previously adopted in the task of style transfer. It substitutes mean and standard deviation of the activations from each CNN [15] filter of the original image with those of the style image as follows:

$$\text{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y), \tag{10}$$

where x and y stand for the activation of a CNN filter from the content image and style image, $\mu(\cdot)$ and $\sigma(\cdot)$ are channel-wise mean and standard deviation. Masui *et al.* [21] further prove that the AdaIN can be directly applied to the LDM without any additional training for style transfer.

8. Full Algorithm

The complete algorithm is represented as Algorithm 1.

9. Efficiency Analysis

When generating a batch of four images with 20 inference steps, AnyText requires 13.183 seconds, whereas our method takes 28.338 seconds. Both approaches are configured consistently, as described in Sec. 4.1.

Algorithm 1: STGen

```
1  Input: A text prompt $y$, an input position image $l_p$,
     a flat reference position image $\tilde{l}_p$ and a set of
     iterations for refinement $\{t_1, \dots, t_k\}$ and a trained
     Visual Text Generation Model $\mathcal{M}$ .
2  Output: Denoised vector $z_0$ .
3  Render the glyph image $l_g$ and compute its latent $z_g$;
4  Compute the reference latent $z_0^f$;
5  Rotate and crop $z_0^f$ to get $z_f$ according to the
     transformation from $\tilde{l}_p$ to $l_p$;
6  for $t = T, T-1, \dots, 1$ do
7      if $t \in \{t_1, \dots, t_k\}$ then
8          $\tilde{z}_t \leftarrow \text{AdaIN}(z_f, z_t) \odot l_p + z_t \odot (1 - l_p)$;
9          $\hat{z}_t \leftarrow \rho\text{AdaIN}(z_g, z_t) + (1-\rho)\tilde{z}_t$;
10         $\ddot{z}_t \leftarrow (\kappa_t \lambda \hat{z}_t + (1-\kappa_t) z_t) \odot l_p + z_t \odot (1 - l_p)$;
11         Set $z_t \leftarrow \ddot{z}_t$;
12     end
13     Set $z_{t-1} \leftarrow \mathcal{M}(z_t, y, t, l_p, l_g)$;
14 end
15 Return $z_0$
```

10. Benchmark Details.

As shown in Fig. 8, the benchmark we created inherited the prompt and the lines of text as much as possible. Besides, our benchmark evaluates the generation ability of the model for generating visual texts in various angles, sizes, and languages.

11. Limitations

Our method still inherits the problems of the baseline AnyText, such as the unsatisfactory ability to generate texts on a small scale or to generate texts in a specific font. Restricted by the latent size, our method may not acquire accurate texts when the text region is too small.

12. Details of User Study

An illustration of our user study is provided in Fig. 9. Since the captions in the English (LAION) set are often hard for users to understand (*e.g.* "carousel of carousel of carousel of carousel of carousel of carousel of carousel of carousel of carousel of carousel of carousel"). To produce meaningful prompts that can help the user understand, we employ LLMs to generate image prompts for user study. Here we provide a simplified template of our instructions and examples:

1.Task instruction

I am working on a text-to-image generation task, where I need to generate images from given Text Prompts. Each Prompt should include a scene description and contain one or two sets of words that need to be present in the image, enclosed in double quotes.

2. Examples

- *A raccoon stands in front of the blackboard with the words "Deep Learning" written on it*
- *A crayon drawing by the child, a snowman with a Santa hat, pine trees, outdoors in heavy snowfall, titled "Snowman".*
- *......*

3.Trigger CoT reasoning ability of LLMs

Reasoning: This task is very important to me. Let's think step by step......

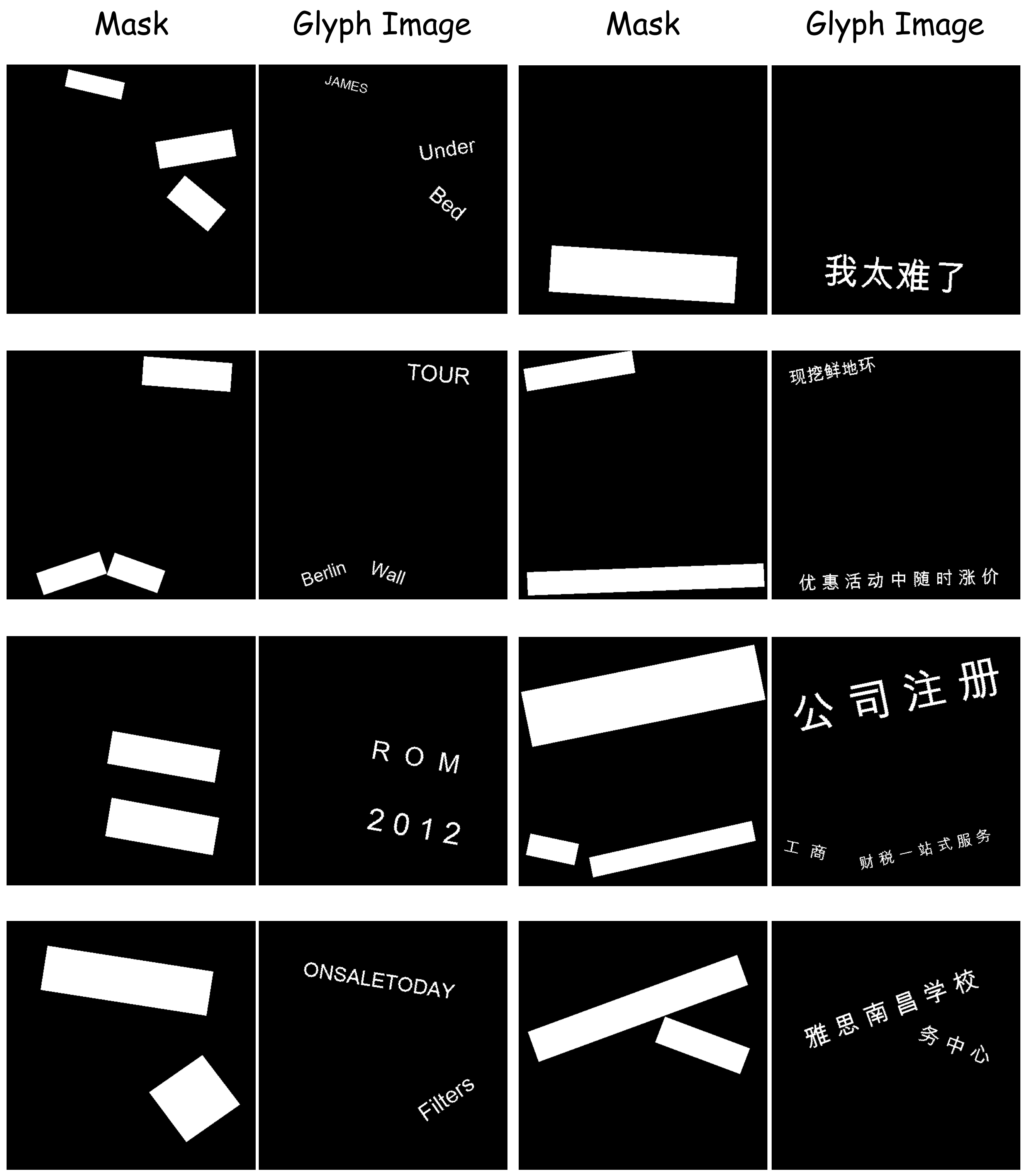
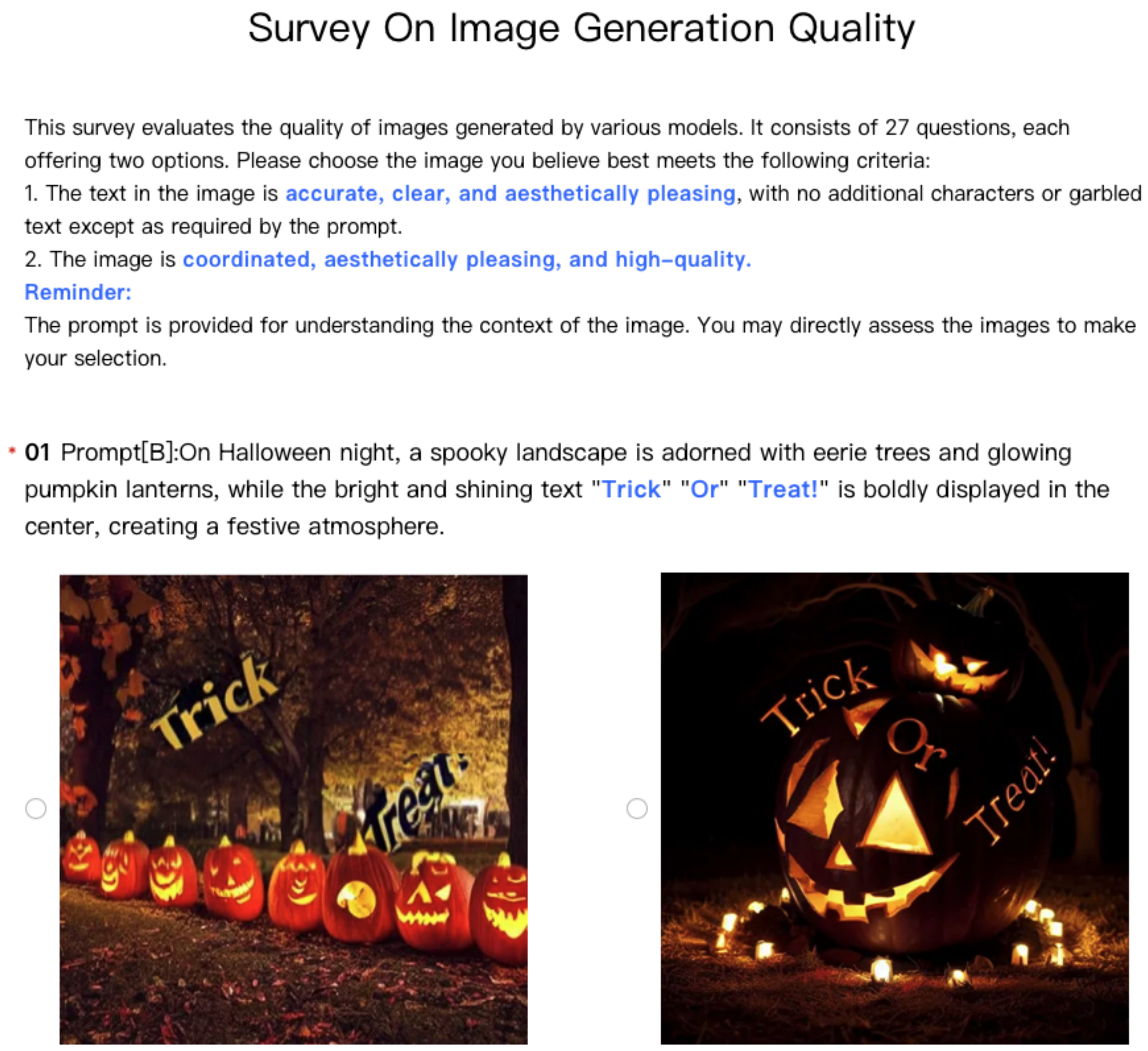


Figure 8. **Benchmark Detail.**



Survey On Image Generation Quality

This survey evaluates the quality of images generated by various models. It consists of 27 questions, each offering two options. Please choose the image you believe best meets the following criteria:
1. The text in the image is **accurate, clear, and aesthetically pleasing**, with no additional characters or garbled text except as required by the prompt.
2. The image is **coordinated, aesthetically pleasing, and high–quality.**
Reminder:
The prompt is provided for understanding the context of the image. You may directly assess the images to make your selection.

* **01** Prompt[B]:On Halloween night, a spooky landscape is adorned with eerie trees and glowing pumpkin lanterns, while the bright and shining text "**Trick**" "**Or**" "**Treat!**" is boldly displayed in the center, creating a festive atmosphere.

Figure 9. **User study print screen.**