

Expressive Text-to-Image Generation with Rich Text

Songwei Ge¹ Taesung Park² Jun-Yan Zhu³ Jia-Bin Huang¹

¹University of Maryland, College Park ²Adobe Research ³Carnegie Mellon University

<https://rich-text-to-image.github.io/>



A pizza with pineapples, pepperonis, and mushrooms on the top, 4k, photorealism.



A young woman sits at a table in a *beautiful, lush garden*, reading a book on the table.

Styles: Johannes Vermeer, Claude Monet



A mesmerizing sight that captures the beauty of a *rose* (71,117, 252) blooming, close up



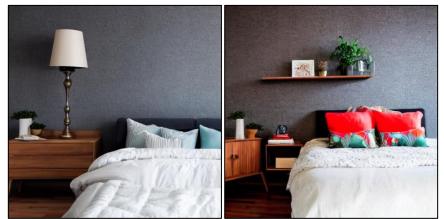
A close-up of a cat¹ riding a scooter. Tropical trees in the background.

¹A cat wearing glasses and has a bandana around its neck.



A vibrant field of *sunflowers* under the sunset. A towering mountain¹ rises up in the distance.

Styles: Hokusai, Van Gogh. ¹A single snow mountain.



A nightstand¹ next to a bed with *pillows* on it.

Grey wall² bedroom.

¹nightstand with some books. ²accent shelf with plants on the wall.

Figure 1. Plain text (left image) vs. Rich text (right image). Our method allows a user to describe an image using a rich text editor that supports various text attributes such as font family, size, color, and footnote. Given these text attributes extracted from rich text prompts, our method enables precise control of text-to-image synthesis regarding colors, styles, and object details compared to plain text.

Abstract

Plain text has become a prevalent interface for text-to-image synthesis. However, its limited customization options hinder users from accurately describing desired outputs. For example, plain text makes it hard to specify continuous quantities, such as the precise RGB color value or importance of each word. Furthermore, creating detailed text prompts for complex scenes is tedious for humans to write and challenging for text encoders to interpret. To address these challenges, we propose using a rich-text editor supporting formats such as font style, size, color, and footnote. We extract each word’s attributes from rich text to enable local style control, explicit token reweighting, precise color rendering, and detailed region synthesis. We achieve these capabilities through a region-based diffusion process. We first obtain each word’s region based on cross-attention maps of a vanilla diffusion process using plain text. For each region, we enforce its text attributes by creating region-specific detailed prompts and applying region-specific guidance. We present various examples of image generation from rich text and demonstrate that our method outperforms strong baselines with quantitative evaluations.

1. Introduction

The development of large-scale text-to-image generative models [46, 50, 48, 25] has propelled image generation to an unprecedented era. The great flexibility of these large-scale models further offers users powerful control of the generation through visual cues [4, 14, 67] and textual inputs [7, 16]. Without exception, existing studies use *plain text* encoded by a pretrained language model to guide the generation. However, in our daily life, it is rare to use only plain text when working on text-based tasks such as writing blogs or editing essays. Instead, a *rich text* editor [60, 62] is the more popular choice providing versatile formatting options for writing and editing text. In this paper, we seek to introduce accessible and precise textual control from rich text editors to text-to-image synthesis.

Rich text editors offer unique solutions for incorporating conditional information separate from the text. For example, using the font color, one can indicate an *arbitrary* color. In contrast, describing the precise color with plain text proves more challenging as general text encoders do not understand RGB or Hex triplets, and many color names, such as ‘salmon red’ and ‘orange’, have ambiguous mean-

ings. This font color information can be used to define the color of generated objects. For example, in Figure 1, a specific `blue` can be selected to instruct the generation of a rose with that exact color.

Beyond providing precise color information, various font formats make it simple to augment the word-level information. For example, reweighting token influence [16] can be implemented using the font size, a task that is difficult to achieve with existing visual or textual interfaces. Rich text editors offer more options than font size – similar to how font style distinguishes the styles of individual text elements, we propose using it to capture the artistic style of specific regions. Another option is using footnotes to provide supplementary descriptions for selected words, simplifying the process of creating complex scenes.

But how can we use rich text? A straightforward implementation is to convert a rich-text prompt with detailed attributes into lengthy plain text and feed it directly into existing methods [48, 16, 7]. Unfortunately, these methods struggle to synthesize images corresponding to lengthy text prompts involving multiple objects with distinct visual attributes, as noted in a recent study [10]. They often mix styles and colors, applying a uniform style to the entire image. Furthermore, the lengthy prompt introduces extra difficulty for text encoders to interpret accurate information, making generating intricate details more demanding.

To address these challenges, our insight is to decompose a rich-text prompt into two components (1) a short plain-text prompt (without formatting) and (2) multiple region-specific prompts that include text attributes, as shown in Figure 2. First, we obtain the cross-attention maps to associate each word with a specific region using a vanilla denoising process with the *plain-text* prompt. Second, we create a short prompt for each region using the attributes derived from *rich-text* prompt. For example, we use “mountain in the style of Ukiyo-e” as the prompt for the region corresponding to the word “mountain” with the attribute “font style: Ukiyo-e”. For RGB font colors that cannot be converted to the prompts, we iteratively update the region with region-based guidance to match the target color. We apply a separate denoising process for each region and fuse the predicted noises to get the final update.

We demonstrate qualitatively and quantitatively that our method generates more precise color, distinct styles, and accurate details compared to plain text-based methods. Our code and demo are available on the project page <https://rich-text-to-image.github.io/>.

2. Related Work

Text-to-image models. Text-to-image systems aim to synthesize realistic images according to scene descriptions [72, 37]. Fueled by the large-scale text-image datasets [54], various training and inference techniques [17, 55, 18, 19], and

scalability [45], significant progress has been made in photorealistic text-to-image generation using Diffusion models [4, 45, 40, 50, 14], autoregressive models [46, 66, 9, 13], GANs [53, 25], and their hybrids [48]. Our work focuses on making these models more accessible and providing precise controls. In contrast to existing work that uses *plain text*, we use a *rich text* editor with various formatting options.

Controllable image synthesis with diffusion models. A wide range of image generation and editing applications have been made possible through either fine-tuning pre-trained diffusion models [49, 28, 67, 3, 63, 27, 36] or modifying the denoising process [38, 11, 16, 41, 5, 10, 2, 4, 23, 6, 52, 68]. For example, SDEdit [38] first adds a small amount of noise to the input and then subsequently denoises it conditioned on the target prompt. Prompt-to-prompt [16] uses the attention maps from the original prompt to guide the spatial structure of the target prompt. InstructPix2Pix [7] trains an image-conditional diffusion model to follow user instructions. Although these methods can be applied to some rich-text-to-image applications, the results often fall short, as shown in Section 4.

Concurrent with our work, Mixture-of-diffusion [23] and MultiDiffusion [6] propose merging multiple diffusion-denoising processes in different image regions through linear blending. Our method shares similar spirits. However, instead of relying on user-provided regions, we automatically compute regions of selected tokens using cross-attention maps. Gradient [21] and Universal [5] guidance control the generation by optimizing the denoised generation at each time step. We apply them to precise color generation by designing an objective on the target region to be optimized.

Cross-attention in diffusion models. Cross attention mechanism has been used in various diffusion-based applications such as view synthesis [32, 59, 61], image editing [16, 10, 42, 41, 28], and video editing [33, 43, 8, 35]. For instance, Attend-and-Excite [10] alleviates the missing object issue by refining corresponding cross-attention maps. Pix2pix-zero [41] minimizes the difference between the original and target image’s cross-attention maps to preserve the spatial structure. We also leverage the alignment information between texts and spatial regions in cross-attention maps for rich-text-to-image generation.

Rich text modeling and application. Exploiting information beyond the intrinsic meanings of the texts has been previously studied [39, 56, 65, 30]. For example, Glyph information in logographic languages helps learn word embeddings [39, 56]. Moreover, visual information, such as underlining and bold type, have also been extracted for various document understanding tasks [65, 30]. To our knowledge, we are the first to leverage rich text information for text-to-image synthesis.

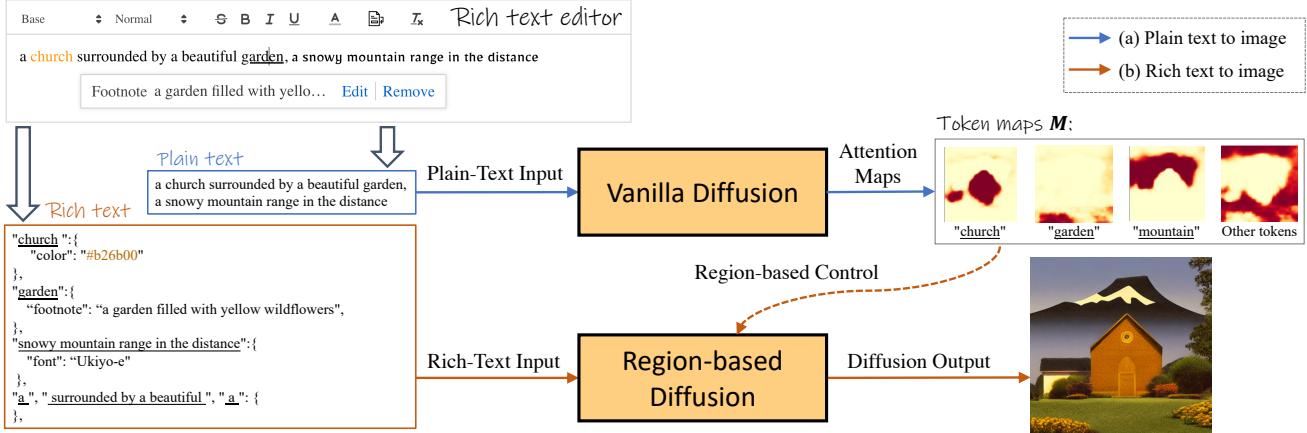


Figure 2. **Rich-text-to-image framework.** First, the token maps of the input prompt are constructed by aggregating the cross-attention maps of a diffusion model using a plain text representation of the input. Then the rich texts are processed as JSON to provide attributes for each token span. The resulting token maps and attributes are used to guide our region-based control.

Image stylization and colorization. Style transfer [15, 71, 34] and colorization [47, 57, 64, 29, 69, 70] for *editing real images* have also been extensively studied. In contrast, our work focuses on precise color and local style control for *generating images* from text-to-image models.

3. Rich Text to Image Generation

From writing messages on communication apps, designing websites [51], to collaboratively editing a document [31, 22], a rich text editor is often the primary interface to edit texts on digital devices. Nonetheless, only plain text has been used in text-to-image generation. To use formatting options in rich-text editors for more precise control over the black-box generation process [1], we first introduce a problem setting called *rich-text-to-image generation*. We then discuss our approach to this task.

3.1. Problem Setting

As shown in Figure 2, a rich text editor supports various formatting options, such as font styles, font size, color, and more. We leverage these text attributes as extra information to increase control of text-to-image generation. We interpret the rich-text prompt as JSON, where each text element consists of a span of tokens e_i (e.g., ‘church’) and attributes a_i describing the span (e.g., ‘color:#b26b00’). Using these annotated prompts, we explore four applications: 1) local style control using *font style*, 2) precise color control using *font color*, 3) detailed region description using *footnotes*, and 4) explicit token reweighting with *font sizes*.

Font style is used to apply a specific artistic style a_i^s , e.g., $a_i^s = \text{‘Ukiyo-e’}$, to the synthesis of the span of tokens e_i . For instance, in Figure 1, we apply the impressionism painting style to the landscape background and the style of Johannes Vermeer to the portrait, enabling the application of

localized artistic styles. This task presents a unique challenge for existing text-to-image models, as there are limited training images featuring multiple artistic styles. Consequently, existing models tend to generate a *uniform* mixed style across the entire image rather than distinct local styles.

Font color indicates a specific color of the modified text span. Given the prompt “a red toy”, the existing text-to-image models generate toys in various shades of red, such as light red, crimson, or maroon. The color attribute provides a way for specifying a *precise color* in the RGB color space, denoted as a_i^c . For example, to generate a toy in fire brick red, one can change the font color to “*a toy*”, where the word “toy” is associated with the attribute $a_i^c = [178, 34, 34]$. However, as shown in the experiment section, the pretrained text encoder cannot interpret the RGB values and have difficulty understanding obscure color names, such as salmon red and orange.

Footnote provides supplementary explanations of the target span without hindering readability with lengthy sentences. Writing detailed descriptions of complex scenes is a tedious work, and it inevitably creates lengthy prompts [26, 24]. Additionally, existing text-to-image models are prone to ignoring some objects when multiple objects are present [10], especially with long prompts. Moreover, excess tokens are discarded when the prompt’s length surpasses the text encoder’s maximum length, e.g., 77 tokens for CLIP models [44]. We aim to mitigate these issues using a footnote string a_i^f .

Font size can be employed to indicate the importance, quantity, or size of an object. We use a scalar a_i^w to denote the weight of each token.

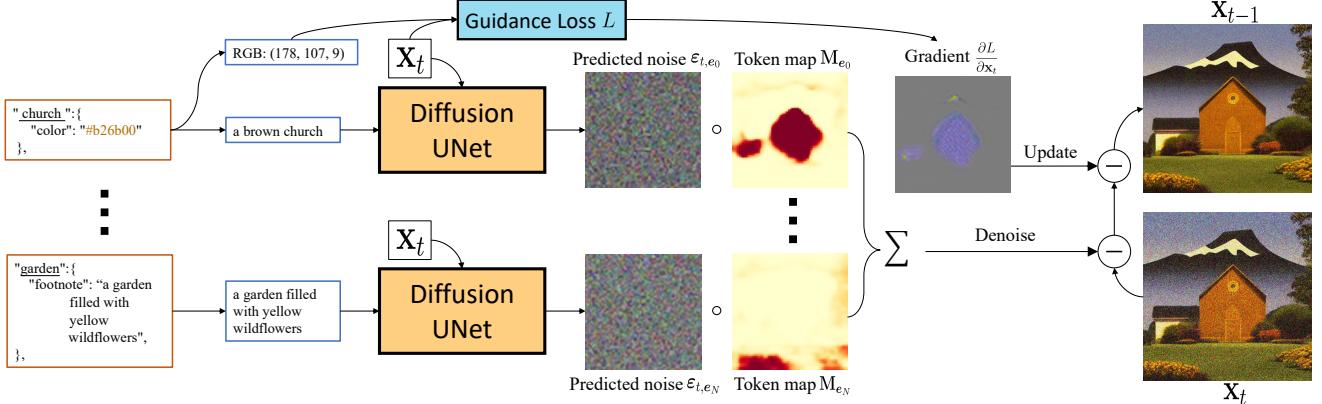


Figure 3. **Region-based diffusion.** For each element of the rich-text input, we apply a separate diffusion process to its region. The attributes are either decoded as a region-based guidance target (e.g. re-coloring the church), or as a textual input to the diffusion UNet (e.g. handling the footnote to the garden). The predicted noise ϵ_{t,e_i} , weighted by the token map M_{e_i} , and the guidance gradient $\frac{\partial L}{\partial \mathbf{x}_t}$ are used to denoise and update the previous generation \mathbf{x}_t to \mathbf{x}_{t-1} .

3.2. Method

To utilize rich text annotations, our method consists of two steps, as shown in Figure 2. First, we compute the spatial layouts of individual token spans. Second, we use a new region-based diffusion to render each region’s attributes into a globally coherent image.

Step 1. Token maps for spatial layout. Several works [58, 35, 4, 16, 10] have discovered that the attention maps in the cross-attention layers of the diffusion UNet indicate the spatial correspondence of individual tokens in the generation. This motivates us to use the plain text as the input to the diffusion model and compute cross-attention maps for each token w_j :

$$\mathbf{m}_j = \frac{\exp(\mathbf{s}_j)}{\sum_k \exp(\mathbf{s}_k)}, \quad (1)$$

where $\mathbf{s}_j = \mathbf{q}_j^\top \mathbf{k}_j$ is the attention score for query \mathbf{q}_j and key \mathbf{k}_j . Note that we can compute such an attention map for each token, head, layer, and timestep of the diffusion model. To aggregate them into a single spatial map per token span, we first interpolate each cross-attention map \mathbf{m}_j to the highest resolution. We then derive the preliminary spatial map $\widehat{\mathbf{M}}_{e_i}$ for each span e_i by first taking the average across heads, layers, and time steps, and then taking the maximum across tokens. Note that taking the element-wise maximal value instead of the average value across the tokens in the span e_i is important. Otherwise, unimportant tokens like stopwords can dilute the map and lead to an inaccurate spatial layout. Additionally, we discard all the “ $<|endoftext|>$ ” and “ $<|startoftext|>$ ” tokens. Finally, we obtain the *token map* in Figure 2 of span e_i as

$$\mathbf{M}_{e_i} = \frac{\exp(\widehat{\mathbf{M}}_{e_i}/\tau)}{\sum_j \exp(\widehat{\mathbf{M}}_{e_j}/\tau)}, \quad (2)$$

where $\tau = 10^{-3}$ is a softmax temperature hyperparameter.

Step 2. Region-based denoising and guidance. As shown in Figure 2, given the text attributes and *token maps*, we divide the overall image synthesis into several region-based denoising and guidance processes to incorporate each attribute, similar to an ensemble of diffusion models [28, 6]. More specifically, given the span e_i , the region defined by its *token map* \mathbf{M}_{e_i} , and the attribute a_i , the predicted noise ϵ_t for noised generation \mathbf{x}_t at time step t is

$$\epsilon_t = \sum_i \mathbf{M}_{e_i} \cdot \epsilon_{t,e_i} = \sum_i \mathbf{M}_{e_i} \cdot D(\mathbf{x}_t, f(e_i, a_i), t), \quad (3)$$

where D is the pretrained diffusion model, and $f(e_i, a_i)$ is a plain text representation derived from text span e_i and attributes a_i using the following process:

1. Initially, we set $f(e_i, a_i) = e_i$.
2. If footnote a_i^f is available, we set $f(e_i, a_i) = a_i^f$.
3. The style a_i^s is appended if it exists. $f(e_i, a_i) = f(e_i, a_i) + \text{'in the style of'} + a_i^s$.
4. The closest color name (string) of font color \hat{a}_i^c from a predefined set \mathcal{C} is prepended. $f(e_i, a_i) = \hat{a}_i^c + f(e_i, a_i)$. For example, $\hat{a}_i^c = \text{'brown'}$ for RGB color $a_i^c = [136, 68, 20]$.

We use $f(e_i, a_i)$ as the original plain text prompt of Step 1 for the spans that do not contain any text attributes. This helps us generate a coherent image, especially around region boundaries.

Guidance. By default, we use classifier-free guidance [20] for each region to better match the prompt $f(e_i, a_i)$. In addition, if the font color is specified, to exploit the RGB

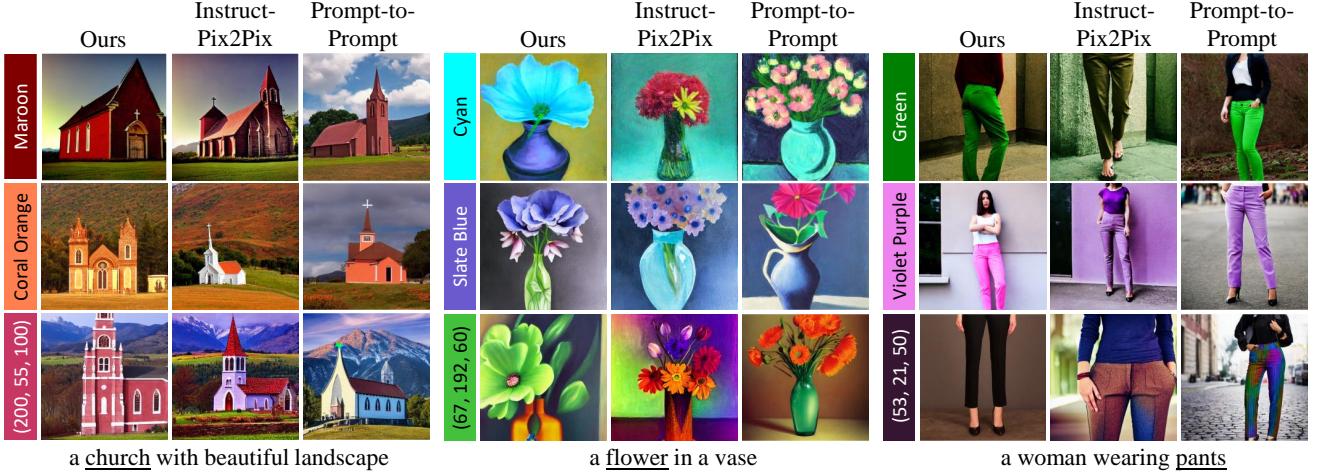


Figure 4. **Qualitative comparison on precise color generation.** We show images generated by Prompt-to-Prompt, InstructPix2Pix, and our method using prompts with font colors. Our method generates precise colors according to either color names or RGB values. Both baselines generate plausible but inaccurate colors given color names, while neither understands the color defined by RGB values. InstructPix2Pix tends to apply the color globally, even outside the target object.

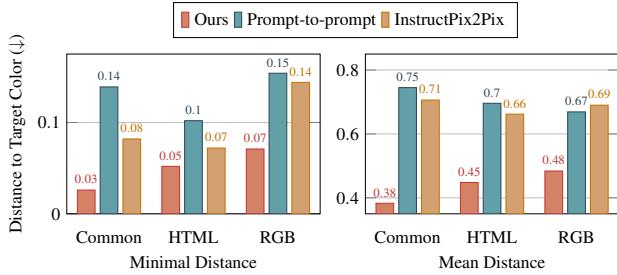


Figure 5. **Quantitative evaluation on precise color generation.** Distance against target color is reported (lower is better). Our method consistently outperforms baselines.

values information further, we apply gradient guidance [21, 12, 5] on the current clean image prediction:

$$\hat{\mathbf{x}}_0 = \frac{\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}_t}{\sqrt{\bar{\alpha}_t}}, \quad (4)$$

where \mathbf{x}_t is the noisy image at time step t , and $\bar{\alpha}_t$ is the coefficient defined by noise scheduling strategy [17]. Here, we compute an MSE loss \mathcal{L} between the average color of $\hat{\mathbf{x}}$ weighted by the *token map* \mathbf{M}_{e_i} and the RGB triplet \mathbf{a}_i^c . The gradient is calculated below,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_t} = \lambda \frac{\partial \| \sum_p (\mathbf{M}_{e_i} \cdot \hat{\mathbf{x}}_0) / \sum_p \mathbf{M}_{e_i} - \mathbf{a}_i^c \|_2^2}{\sqrt{\bar{\alpha}_t} \partial \hat{\mathbf{x}}_0}, \quad (5)$$

where the summation is over all pixels p and λ is a hyper-parameter to control the strength of the guidance. We use $\lambda = 1$ unless denoted otherwise.

Token reweighting with font size. Last, to re-weight the impact of the token w_j according to the font size a_j^w , we modify its cross-attention maps \mathbf{m}_j . However, instead

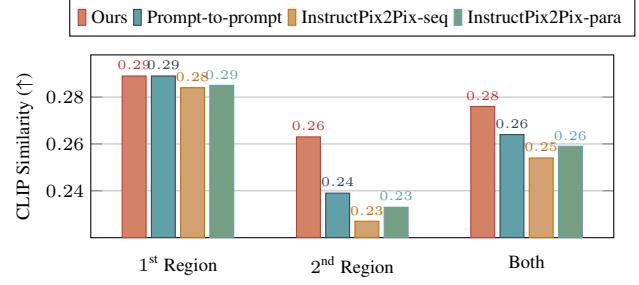


Figure 6. **Quantitative evaluation of local style control.** We report the CLIP similarity between each stylized region and its region prompt. Our method achieves the best stylization.

of direct multiplication as in Prompt-to-Prompt [16] where $\sum_j a_j^w \mathbf{m}_j \neq 1$, we find that it is critical to preserve the probability property of \mathbf{m}_j . We thus propose the following reweighting approach:

$$\hat{\mathbf{m}}_j = \frac{a_j^w \exp(\mathbf{s}_j)}{\sum_k a_k^w \exp(\mathbf{s}_k)}. \quad (6)$$

We can compute the token map (Equation 2) and predict the noise (Equation 3) with the reweighted attention map.

4. Experimental Results

Implementation details. We use Stable Diffusion V1-5 [48] for our experiments. To create the token maps, we use the cross-attention layers in all blocks, excluding the first encoder and last decoder blocks, as the attention maps in these high-resolution layers are often noisy. We discard the maps at the initial denoising steps with $T > 750$. We report the results averaged from three random seeds for all quantitative experiments. More details, such as the running time,



The figure shows two rows of images. The top row compares 'A night sky filled with stars (1st Region: Van Gogh) above a turbulent sea with giant waves (2nd Region: Ukiyo-e)'. The bottom row compares 'The awe-inspiring sky and sea (1st Region: J.M.W. Turner) by a coast with flowers and grasses in spring (2nd Region: Monet)'. Four methods are compared: Ours, Prompt-to-Prompt, InstructPix2Pix-para, and InstructPix2Pix-seq. 'Ours' shows distinct styles for both regions. 'Prompt-to-Prompt' and 'InstructPix2Pix-para' show similar or identical styles across both regions. 'InstructPix2Pix-seq' shows a mix of styles.



The figure shows four pizzas. The first is labeled 'Ours: A pizza with pineapples, pepperonis, and mushrooms'. The second is labeled 'Prompt-to-Prompt'. The third is labeled 'A pizza with pineapples, pepperonis, and mushrooms, mushrooms, mushrooms, mushrooms, mushrooms'. The fourth is labeled 'A pizza with pineapples, pepperonis, and (((((mushrooms))))). The Prompt-to-Prompt version has artifacts due to large weight and unbounded probability. Heuristic methods like repeating and parenthesis do not work well.

can be found in Appendix B. **Font style evaluation.** We compute CLIP scores [44] for each local region to evaluate the stylization quality. Specifically, we create prompts of two objects and styles. We create combinations using seven popular styles and ten objects, resulting in 420 prompts. For each generated image, we mask it by the token maps of each object and attach the masked output to a black background. Then, we compute the CLIP score using the region-specific prompt. For example, for the prompt “a lighthouse (Cyberpunk) among the turbulent waves (Ukiyo-e)”, the local

CLIP score of the lighthouse region is measured by comparing its similarity with the prompt “lighthouse in the style of cyberpunk.”

Font color evaluation. To evaluate a method’s capacity to understand and generate a specific color, we divide colors into three categories. The *Common color* category contains 17 standard names, such as “red”, “yellow”, and “pink”. The *HTML color* names are selected from the web color names ¹ used for website design, such as “sky blue”,

¹https://simple.wikipedia.org/wiki/Web_color

A coffee table¹ sits in front of a sofa² on a cozy carpet. A painting³ on the wall. cinematic lighting, trending on artstation, 4k, hyperrealistic, focused, extreme details.

¹A rustic wooden coffee table adorned with scented candles and many books.

²A plush sofa with a soft blanket and colorful pillows on it.

³A painting of wheat field with a cottage in the distance, close up shot, trending on artstation, hd, calm, complimentary color, realistic lighting, by Albert Bierstadt, Frederic Church.



Figure 9. **Qualitative comparison on detailed description generation.** We show images generated by Attend-and-Excite, Prompt-to-Prompt, InstructPix2Pix, and our method using complex prompts. Our method is the only one that can generate all the details faithfully.

“lime green”, and “violet purple”. The *RGB color* category contains 50 randomly sampled RGB triplets to be used as “color of RGB values [128, 128, 128]”. To create a complete prompt, we use 12 objects exhibiting different colors, such as “flower”, “gem”, and “house”. This gives us a total of 1,200 prompts. We evaluate color accuracy by computing the mean L2 distance between the region and target RGB values. We also compute the minimal L2 distance as sometimes the object should contain other colors for fidelity, e.g., the “black tires” of a “yellow car”.

Baselines. We compare our method quantitatively with two strong baselines, Prompt-to-Prompt [16] and InstructPix2Pix [7]. When there are two instructions for each image in our font style experiments, we apply them in both parallel (InstructPix2Pix-para) and sequential manners (InstructPix2Pix-seq). More details are discussed in Appendix B. For re-weighting token importance, we visu-

ally compare with Prompt-to-Prompt [16] and two common heuristic methods, repeating and adding parentheses. For complex scene generation with footnotes, we also compare with Attend-and-Excite [10].

4.1. Quantitative Comparison

We report the local CLIP scores computed by a ViT-B/32 model in Figure 6. Our method achieves the best overall CLIP score compared to the two baselines. This demonstrates the advantage of our region-based diffusion method for localized stylization. To further understand the capacity of each model to generate multiple styles, we report the metric on each region. Prompt-to-Prompt and InstructPix2Pix-para achieve a decent score on the 1st Region, i.e., the region first occurs in the sentence. However, they often fail to fulfill the style in the 2nd Region. We conjecture that the Stable Diffusion model tends to generate a uniform style for the

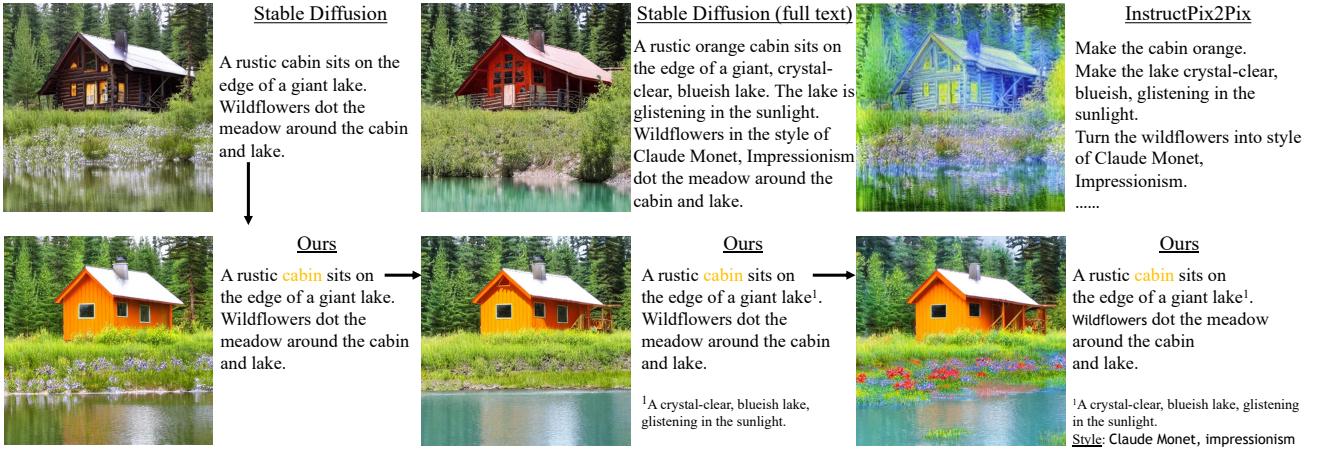


Figure 10. **Our workflow.** (top left) A user begins with an initial plain-text prompt and wishes to refine the scene by specifying the color, details, and styles. (top center) Naively inputting the whole description in plain text does not work. (top right) InstructPix2Pix [7] fails to make accurate editing. (bottom) Our method supports precise refinement with region-constrained diffusion processes. Moreover, our framework can naturally be integrated into a rich text editor, enabling a tight, streamlined UI.

entire image, which can be attributed to single-style training images. Furthermore, InstructPix2Pix-seq performs the worst in 2nd Region. This is because the first instruction contains no information about the second region, and the second region’s content could be compromised when we apply the first instruction.

We show quantitative results of precise color generation in Figure 5. The distance of *HTML color* is generally the lowest for baseline methods, as they provide the most interpretable textual information for text encoders. This aligns with our expectation that the diffusion model can handle simple color names, whereas they struggle to handle the RGB triplet. Our rich-text-to-image generation method consistently improves on the three categories and two metrics over the baselines.

4.2. Visual Comparison

Precise color generation. We show qualitative comparison on precise color generation in Figure 4. InstructPix2Pix [7] is prone to create global color effects rather than accurate local control. For example, in the flower results, both the vase and background are changed to the target colors. Prompt-to-Prompt [16] provides more precise control over the target region. However, both Prompt-to-Prompt and InstructPix2Pix fail to generate precise colors. In contrast, our method can generate precise colors for all categories and prompts.

Local style generation. Figure 7 shows a visual comparison of local style generation. When applying InstructPix2Pix-seq, the style in the first instruction dominates the entire image and undermines the second region. Figure 22 in Appendix shows that this cannot be fully

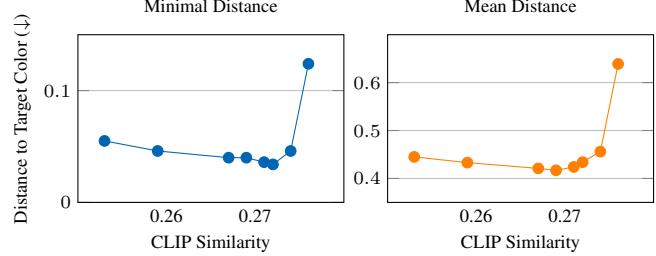


Figure 11. **Ablation on the hyperparameter λ in Equation 5.** We report the trade-off of CLIP similarity and color distance achieved by sweeping the strength of color optimization λ .

resolved using different hyperparameters of classifier-free guidance. Similar to our observation in the quantitative evaluation, our baselines tend to generate the image in a globally uniform style instead of distinct local styles for each region. In contrast, our method synthesizes the correct styles for both regions. One may suggest applying baselines with two stylization processes independently and composing the results using token maps. However, as shown in Figure 13 and Figure 27 (Appendix), such methods generate artifacts on the region boundaries.

Complex scene generation. Figure 9 shows comparisons on complex scene generation. Attend-and-Excite [10] uses the tokens missing in the full-text generation result as input to fix the missing objects, like the coffee table and carpet in the living room example. However, it still fails to generate all the details correctly, e.g., the books, the painting, and the blanket. Prompt-to-Prompt [16] and InstructPix2Pix [7] can edit the painting accordingly, but many objects, like the colorful pillows and stuff on the table, are still missing. In

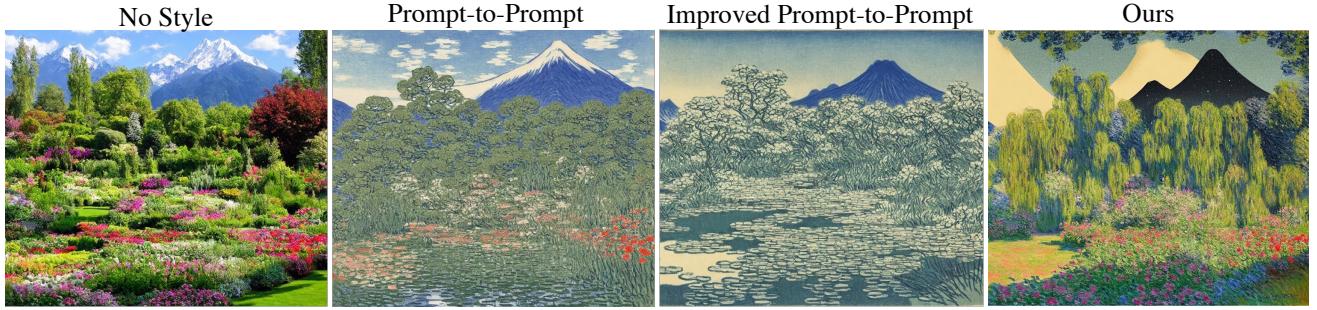


Figure 12. **Improved Prompt-to-Prompt.** Further constraining the attention maps for styles does not resolve the mixed style issue.

contrast, our method faithfully synthesizes all these details described in the target region.

Token importance control. Figure 8 shows the qualitative comparison on token reweighting. Both our method and Prompt-to-Prompt [16] use a large weight for ‘mushroom’. Prompt-to-Prompt generates clear artifacts as it modifies the attention probabilities to be unbounded and creates out-of-distribution intermediate features. Heuristic methods fail with adding more mushrooms, while our method generates more mushrooms and preserves the quality. The results of different font sizes are shown in Figures 23 - 25 in Appendix.

Interactive editing. In Figure 10, we showcase a sample workflow to illustrate our method’s interactive strength and editing capacity over InstructPix2Pix [7].

4.3. Ablation Study

Ablation of the color guidance weight. Changing the guidance strength λ allows us to control the trade-off between *fidelity* and *color precision*. To evaluate the fidelity of the image, we compute the CLIP score between the generation and the plain text prompt. We plot the CLIP similarity vs. color distance in Figure 11 by sweeping λ from 0 to 20. Increasing the strength always reduces the CLIP similarity as details are removed to satisfy the color objective. We find that larger λ first reduces and then increases the distances due to the optimization divergence.

Constrained Prompt-to-Prompt. The original Attention Refinement proposed in Prompt-to-Prompt [16] does not apply any constraint to newly added tokens’ attention maps, which may be the reason that it fails with generating distinct styles. Therefore, we attempt to improve Prompt-to-Prompt by injecting the cross-attention maps for the newly added style tokens. For example, in Figure 12, we use the cross attention map of “garden” for the style “Claude Monet”. However, the method still produces a uniform style.



Figure 13. **Composed-based method.** We apply InstructPix2Pix [7] on two regions separately and compose the results using token maps. The generation contains sharp changes and artifacts at the boundary areas. We show three random seeds.

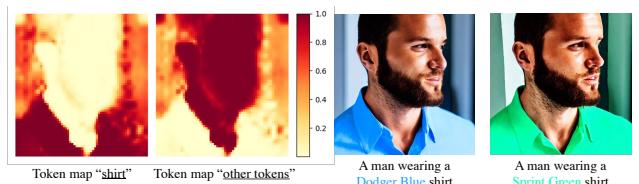


Figure 14. **Failure cases.** The color leaks to the background due to the inaccurate “shirt” token map.

5. Discussion and Limitations

In this paper, we have expanded the controllability of text-to-image models by incorporating rich-text attributes as the input. We have demonstrated the potential for generating images with local styles, precise colors, different token importance, and complex descriptions.

Nevertheless, numerous formatting options remain unexplored, such as bold/italic, hyperlinks, spacing, and bullets/numbering. Moreover, there are multiple ways to use

the same formatting options. For example, one can use font style to characterize the shape of the objects. We hope this paper encourages further exploration of integrating accessible daily interfaces into text-based generation tasks, even beyond images.

Limitations. Our approach relies on cross-attention maps to associate text tokens with spatial regions. However, these attention maps could be sometimes inaccurate [10]. For example, in Figure 14, since the prompt has no information about the background, the token map of “shirt” also covers partial background regions. As a result, the target color *bleeds* into the background. Some preprocessing steps, such as thresholding or erosion, can mitigate the issue but often require case-by-case tuning. Also, as we use multiple diffusion processes and two-stage methods, our method can be multiple times slower than the original process.

Acknowledgment. We thank Mia Tang, Aaron Hertzmann, Nupur Kumari, Gaurav Parmar, Ruihan Gao, and Aniruddha Mahapatra for their helpful discussion and paper reading. This work is partly supported by NSF grant No. IIS-239076, as well as NSF grants No. IIS-1910132 and IIS-2213335.

References

- [1] Maneesh Agrawala. Unpredictable black boxes are terrible interfaces, March 2023. <https://magrawala.substack.com/p/unpredictable-black-boxes-are-terrible>. 3
- [2] Omri Avrahami, Ohad Fried, and Dani Lischinski. Blended latent diffusion. *arXiv preprint arXiv:2206.02779*, 2022. 2
- [3] Omri Avrahami, Thomas Hayes, Oran Gafni, Sonal Gupta, Yaniv Taigman, Devi Parikh, Dani Lischinski, Ohad Fried, and Xi Yin. Spatext: Spatio-textual representation for controllable image generation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2
- [4] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022. 1, 2, 4
- [5] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models. *arXiv preprint arXiv:2302.07121*, 2023. 2, 5
- [6] Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. Multidiffusion: Fusing diffusion paths for controlled image generation. *arXiv preprint arXiv:2302.08113*, 2023. 2, 4
- [7] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2, 7, 8, 9, 13, 28
- [8] Duygu Ceylan, Chun-Hao Huang, and Niloy J. Mitra. Pix2video: Video editing using image diffusion. *arXiv:2303.12688*, 2023. 2
- [9] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023. 2
- [10] Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. *arXiv preprint arXiv:2301.13826*, 2023. 2, 3, 4, 7, 8, 10, 13
- [11] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. Ilvr: Conditioning method for denoising diffusion probabilistic models. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 2
- [12] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Conference on Neural Information Processing Systems (NeurIPS)*, volume 34, pages 8780–8794. Curran Associates, Inc., 2021. 5
- [13] Ming Ding, Wendi Zheng, Wenyi Hong, and Jie Tang. Cogview2: Faster and better text-to-image generation via hierarchical transformers. *arXiv preprint arXiv:2204.14217*, 2022. 2
- [14] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors. In *European Conference on Computer Vision (ECCV)*, pages 89–106. Springer, 2022. 1, 2
- [15] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3
- [16] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022. 1, 2, 4, 5, 7, 8, 9, 13, 23, 28
- [17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Neural Information Processing Systems (NeurIPS)*, 2020. 2, 5
- [18] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022. 2
- [19] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 2
- [20] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 4
- [21] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 2, 5
- [22] Claudia-Lavinia Ignat, Luc André, and Gérald Oster. Enhancing rich content wikis with real-time collaboration. *Concurrency and Computation: Practice and Experience*, 33(8):e4110, 2021. 3

- [23] Álvaro Barbero Jiménez. Mixture of diffusers for scene composition and high resolution image generation. *arXiv preprint arXiv:2302.02412*, 2023. 2
- [24] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4565–4574, 2016. 3
- [25] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2
- [26] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015. 3
- [27] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2
- [28] Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. *arXiv preprint arXiv:2212.04488*, 2022. 2, 4
- [29] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM SIGGRAPH*, 2004. 3
- [30] Junlong Li, Yiheng Xu, Tengchao Lv, Lei Cui, Cha Zhang, and Furu Wei. Dit: Self-supervised pre-training for document image transformer. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 3530–3539, 2022. 2
- [31] Geoffrey Litt, Sarah Lim, Martin Kleppmann, and Peter van Hardenberg. Peritext: A crdt for collaborative rich text editing. *Proceedings of the ACM on Human-Computer Interaction (PACMHCI)*, 2022. 3
- [32] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. *arXiv preprint arXiv:2303.11328*, 2023. 2
- [33] Shaoteng Liu, Yuechen Zhang, Wenbo Li, Zhe Lin, and Jiaya Jia. Video-p2p: Video editing with cross-attention control. *arXiv:2303.04761*, 2023. 2
- [34] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3
- [35] Wan-Duo Kurt Ma, JP Lewis, W Bastiaan Kleijn, and Thomas Leung. Directed diffusion: Direct control of object placement through attention guidance. *arXiv preprint arXiv:2302.13153*, 2023. 2, 4
- [36] Yue Ma, Yingqing He, Xiaodong Cun, Xintao Wang, Ying Shan, Xiu Li, and Qifeng Chen. Follow your pose: Pose-guided text-to-video generation using pose-free videos, 2023. 2
- [37] Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Generating images from captions with attention. In *International Conference on Learning Representations (ICLR)*, 2016. 2
- [38] Chenlin Meng, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Image synthesis and editing with stochastic differential equations. *International Conference on Learning Representations (ICLR)*, 2022. 2
- [39] Yuxian Meng, Wei Wu, Fei Wang, Xiaoya Li, Ping Nie, Fan Yin, Muyu Li, Qinghong Han, Xiaofei Sun, and Jiwei Li. Glyce: Glyph-vectors for chinese character representations. *Neural Information Processing Systems (NeurIPS)*, 32, 2019. 2
- [40] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *International Conference on Machine Learning (ICML)*, pages 16784–16804, 2022. 2
- [41] Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. *arXiv preprint arXiv:2302.03027*, 2023. 2
- [42] Or Patashnik, Daniel Garabi, Idan Azuri, Hadar Averbuch-Elor, and Daniel Cohen-Or. Localizing object-level shape variations with text-to-image diffusion models. *arXiv preprint arXiv:2303.11306*, 2023. 2
- [43] Chenyang Qi, Xiaodong Cun, Yong Zhang, Chenyang Lei, Xintao Wang, Ying Shan, and Qifeng Chen. Fatezero: Fusing attentions for zero-shot text-based video editing, 2023. 2
- [44] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021. 3, 6
- [45] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 2
- [46] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning (ICML)*, pages 8821–8831, 2021. 1, 2
- [47] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):34–41, 2001. 3
- [48] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 5, 25
- [49] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2
- [50] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Bureu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J

- Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. 1, 2, 25
- [51] Arnaud Sahuguet and Fabien Azavant. Wysiwyg web wrapper factory (w4f), 1999. 3
- [52] Vishnu Sarukkai, Linden Li, Arden Ma, Christopher R'e, and Kayvon Fatahalian. Collage diffusion. *ArXiv*, abs/2303.00262, 2023. 2
- [53] Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis. *arXiv preprint arXiv:2301.09515*, 2023. 2
- [54] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 2
- [55] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021. 2
- [56] Zijun Sun, Xiaoya Li, Xiaofei Sun, Yuxian Meng, Xiang Ao, Qing He, Fei Wu, and Jiwei Li. Chinesebert: Chinese pretraining enhanced by glyph and pinyin information. *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021. 2
- [57] Yu-Wing Tai, Jiaya Jia, and Chi-Keung Tang. Local color transfer via probabilistic segmentation by expectation-maximization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 747–754 vol. 1, 2005. 3
- [58] Raphael Tang, Akshat Pandey, Zhiying Jiang, Gefei Yang, K. V. S. Manoj Kumar, Jimmy Lin, and Ferhan Ture. What the daam: Interpreting stable diffusion using cross attention. *ArXiv*, abs/2210.04885, 2022. 4
- [59] Hung-Yu Tseng, Qinbo Li, Changil Kim, Suhib Alsian, Jia-Bin Huang, and Johannes Kopf. Consistent view synthesis with pose-guided diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2
- [60] Colorado State University. tutorial: Rich text format (rtf) from microsoft word - the access project - colorado state university, 2012-07-08. 1
- [61] Daniel Watson, William Chan, Ricardo Martin-Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models, 2022. 2
- [62] Ian H Witten, David Bainbridge, and David M Nichols. *How to build a digital library*. Morgan Kaufmann, 2009. 1
- [63] Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Weixian Lei, Yuchao Gu, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. *arXiv preprint arXiv:2212.11565*, 2022. 2
- [64] Li Xu, Qiong Yan, and Jiaya Jia. A sparse control model for image and video editing. *ACM Transactions on Graphics (TOG)*, 32:1 – 10, 2013. 3
- [65] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1192–1200, 2020. 2
- [66] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research*, 2022. 2
- [67] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023. 1, 2
- [68] Qinsheng Zhang, Jiaming Song, Xun Huang, Yongxin Chen, and Ming-Yu Liu. Diffcollage: Parallel generation of large content with diffusion models. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2
- [69] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conference on Computer Vision (ECCV)*, 2016. 3
- [70] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics (TOG)*, 9(4), 2017. 3
- [71] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 3
- [72] Xiaojin Zhu, Andrew B Goldberg, Mohamed Eldawy, Charles R Dyer, and Bradley Strock. A text-to-picture synthesis system for augmenting communication. In *AAAI Conference on Artificial Intelligence*, 2007. 2

In this appendix, we provide additional experimental results and details. In section A, we show the images generated by our model, Attend-and-Excite [10], Prompt-to-Prompt [16], and InstructPix2Pix [7] with various RGB colors, local styles, and detailed descriptions via footnotes. In section B, we provide additional details on the implementation and evaluation.

A. Additional Results

In this section, we first show additional results of rich-text-to-image generation on complex scene synthesis (Figures 15, 16, and 17), precise color rendering (Figures 18, 19, and 20), local style control (Figures 21 and 22), and explicit token re-weighting (Figure 23, 24, and 25). We also show an ablation study of the averaging and maximizing operations across tokens to obtain token maps in Figure 26. We present additional results compared with a composition-based baseline in Figure 27. Last, we show an ablation of the hyperparameters of our baseline method InstructPix2Pix [7] on the local style generation application in Figure 28.

A car¹ driving on the road. A bicycle² nearby a tree³. A cityscape⁴ in the background.

¹A sleek sports car gleams on the road in the sunlight, with its aerodynamic curves and polished finish catching the light. ²A bicycle with rusted frame and worn tires.

³A dead tree with a few red apples on it. ⁴A bustling Hongkong cityscape with towering skyscrapers.



Figure 15. **Additional results of the footnote.** We show the generation from a complex description of a garden. Note that all the methods except for ours fail to generate accurate details of the mansion and fountain as described.

A lush garden¹ with a fountain². A grand mansion³ in the background.

¹A garden is full of vibrant colors with a variety of flowers.

²A fountain made of white marble with multiple tiers. The tiers are intricately carved with various designs.

³An impressive two-story mansion with a royal exterior, white columns, and tile-made roof. The mansion has numerous windows, each adorned with white curtains.



Figure 16. **Additional results of the footnote.** We show the generation from a complex description of a garden. Note that all the methods except for ours fail to generate accurate details of the mansion and fountain as described.

A small chair¹ sits in front of a table² on the wooden floor. There is a bookshelf³ nearby the window⁴.

¹A black leather office chair with a high backrest and adjustable arms.

²A large wooden desk with a stack of books on top of it.

³A bookshelf filled with colorful books and binders.

⁴A window overlooks a stunning natural landscape of snow mountains.



Stable Diffusion (Plain-Text)



Stable Diffusion (Full-Text)



Ours



Attend-and-Excite



Prompt-to-Prompt



InstructPix2Pix

Figure 17. **Additional results of the footnote.** We show the generation from a complex description of an office. Note that all the methods except ours fail to generate accurate window overviews and colorful binders as described.



Figure 18. **Additional results of the font color.** We show the generation of different objects with colors from the *Common* category. Prompt-to-Prompt has a large failure rate of respecting the given color name, while InstructPix2Pix tends to color the background and irrelevant objects.

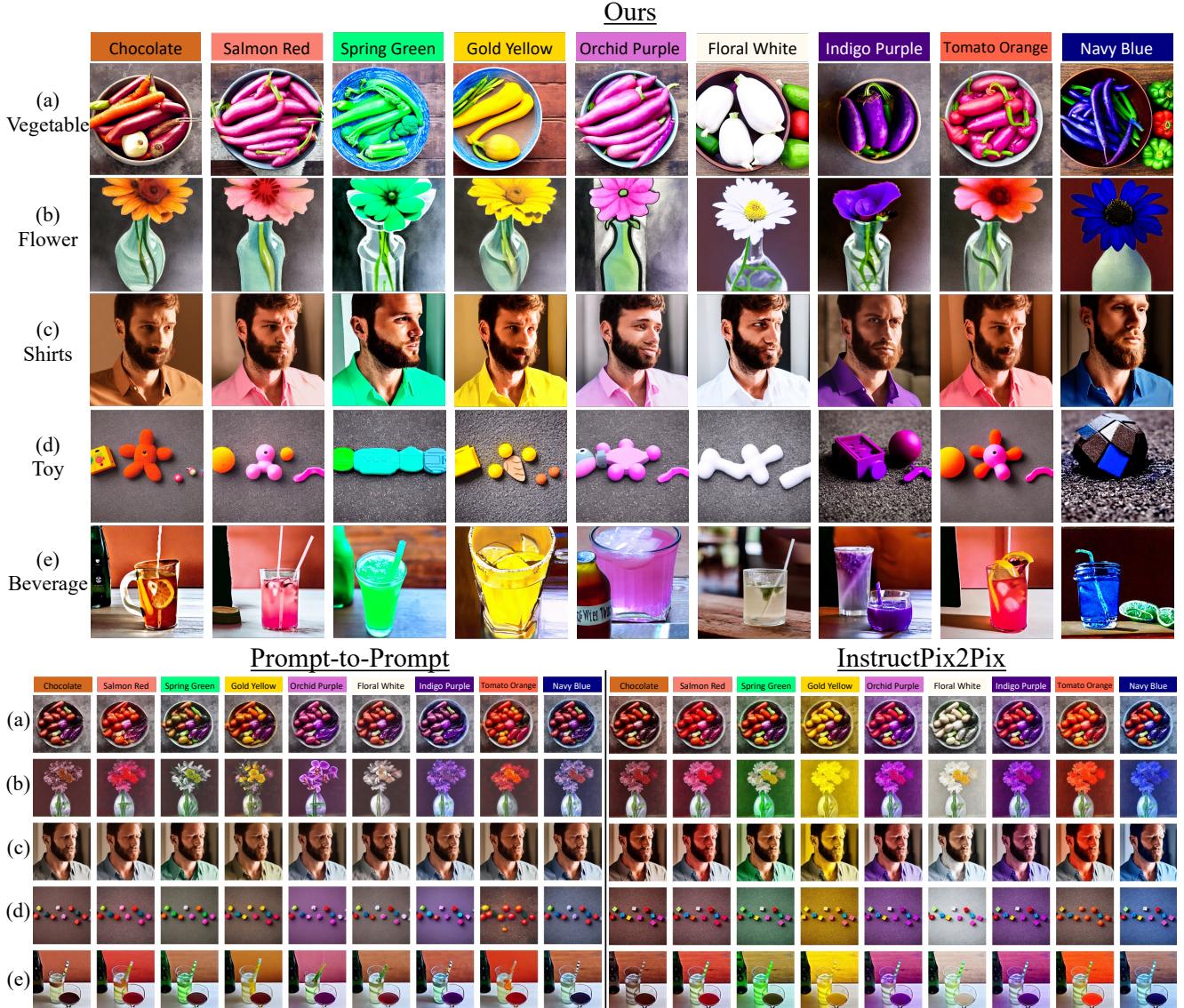


Figure 19. **Additional results of the font color.** We show the generation of different objects with colors from the *HTML category*. Both methods fail to generate the precise color, and InstructPix2Pix tends to color the background and irrelevant objects.

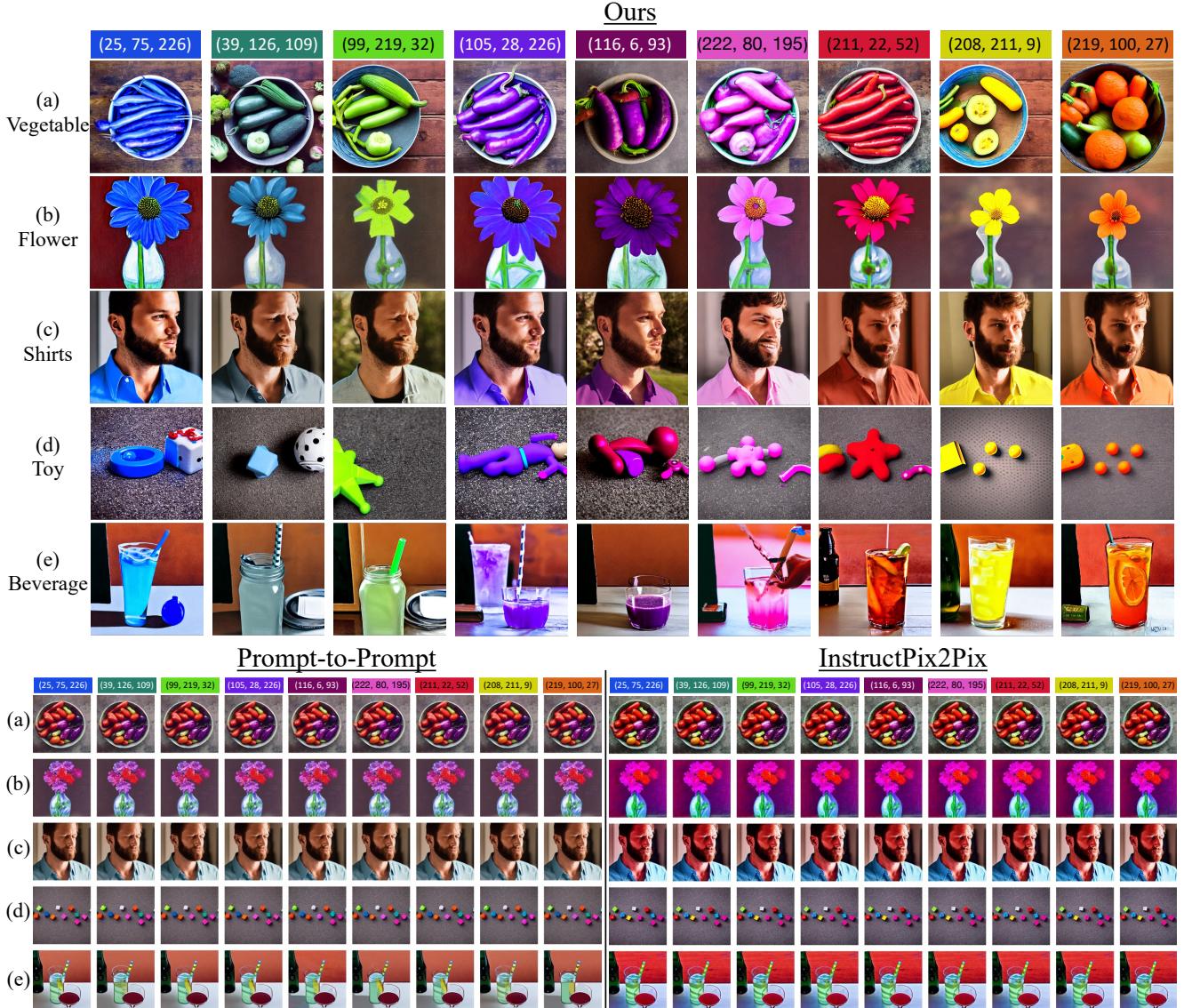


Figure 20. **Additional results of the font color.** We show the generation of different objects with colors from the *RGB category*. Both baseline methods cannot interpret the RGB values correctly.

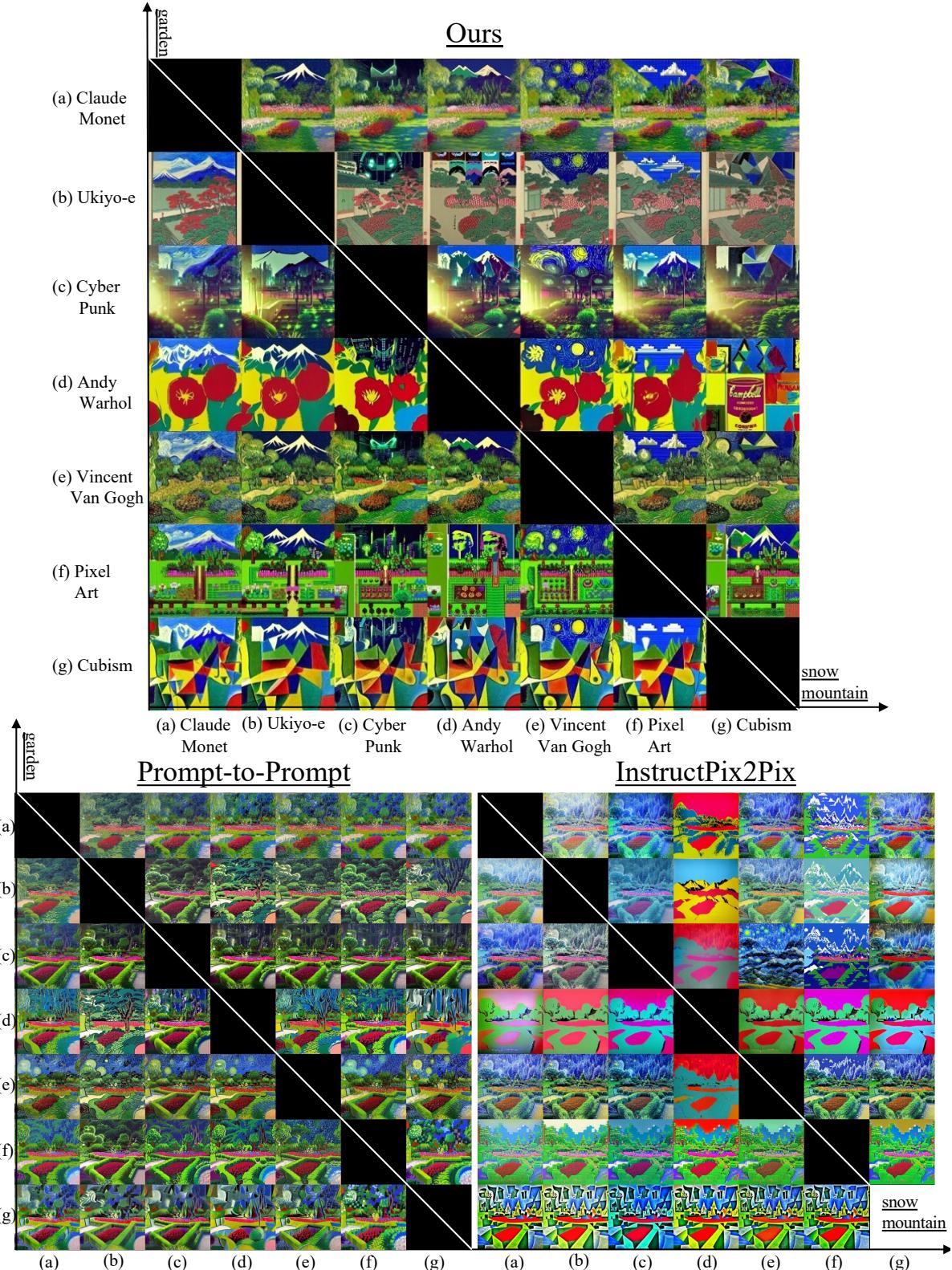


Figure 21. Additional results of the font style. We show images generated with different style combinations and prompt “a beautiful garden in front of a snow mountain”. Each row contains “snow mountain” in 7 styles, and each column contains “garden” in 7 styles. Only our method can generate distinct styles for both objects.

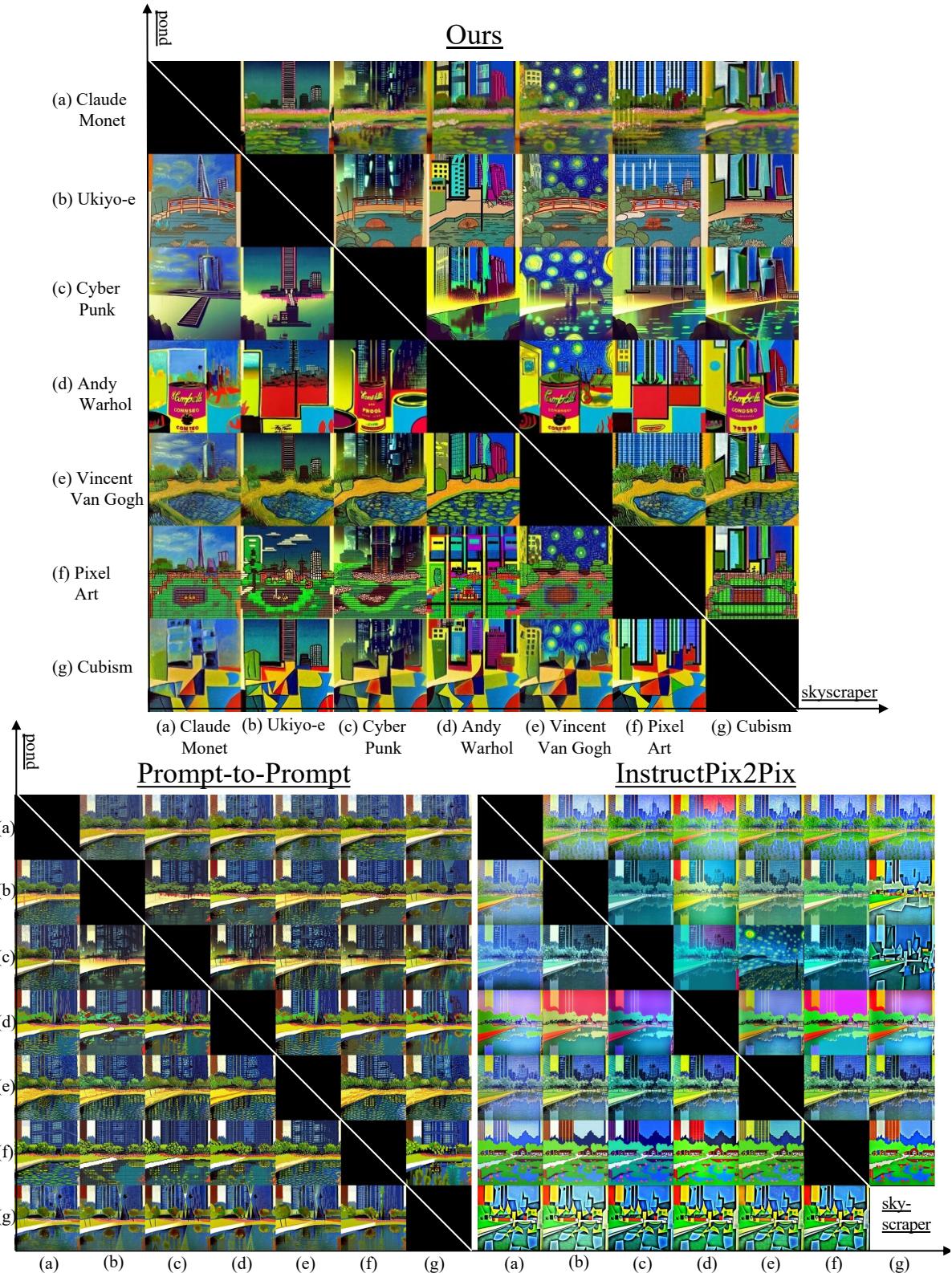


Figure 22. **Additional results of the font style.** We show images generated with different style combinations and prompt “a small pond surrounded by skyscraper”. Each row contains “skyscraper” in 7 styles, and each column contains “pond” in 7 styles. Only our method can generate distinct styles for both objects.



Ours: A pizza with pineapples, pepperonis, and mushrooms.



Prompt-to-Prompt: A pizza with pineapples, pepperonis, and mushrooms.



Parenthesis: A pizza with pineapples, pepperonis, and ((mushrooms)).



Repeating: A pizza with pineapples, pepperonis, and mushrooms, mushrooms, mushrooms.

Figure 23. **Additional results of font sizes.** We use a token weight evenly sampled from 1 to 20 for the word ‘mushrooms’ with our method and Prompt-to-Prompt. For parenthesis and repeating, we show results by repeating the word ‘mushrooms’ and adding parentheses to the word ‘mushrooms’ for 1 to 10 times. Prompt-to-Prompt suffers from generating artifacts. Heuristic methods are not effective.



Ours: A pizza with pineapples, pepperonis, and mushrooms.



Prompt-to-Prompt: A pizza with pineapples, pepperonis, and mushrooms.



Parenthesis: A pizza with ((pineapples)), pepperonis, and mushrooms.



Repeating: A pizza with pineapples, pineapples, pineapples, pepperonis, and mushrooms.

Figure 24. **Additional results of font sizes.** We use a token weight evenly sampled from 1 to 20 for the word ‘pineapples’ with our method and Prompt-to-Prompt. For parenthesis and repeating, we show results by repeating the word ‘pineapples’ and adding parentheses to the word ‘pineapples’ for 1 to 10 times. Prompt-to-Prompt suffers from generating artifacts. Heuristic methods are not effective.

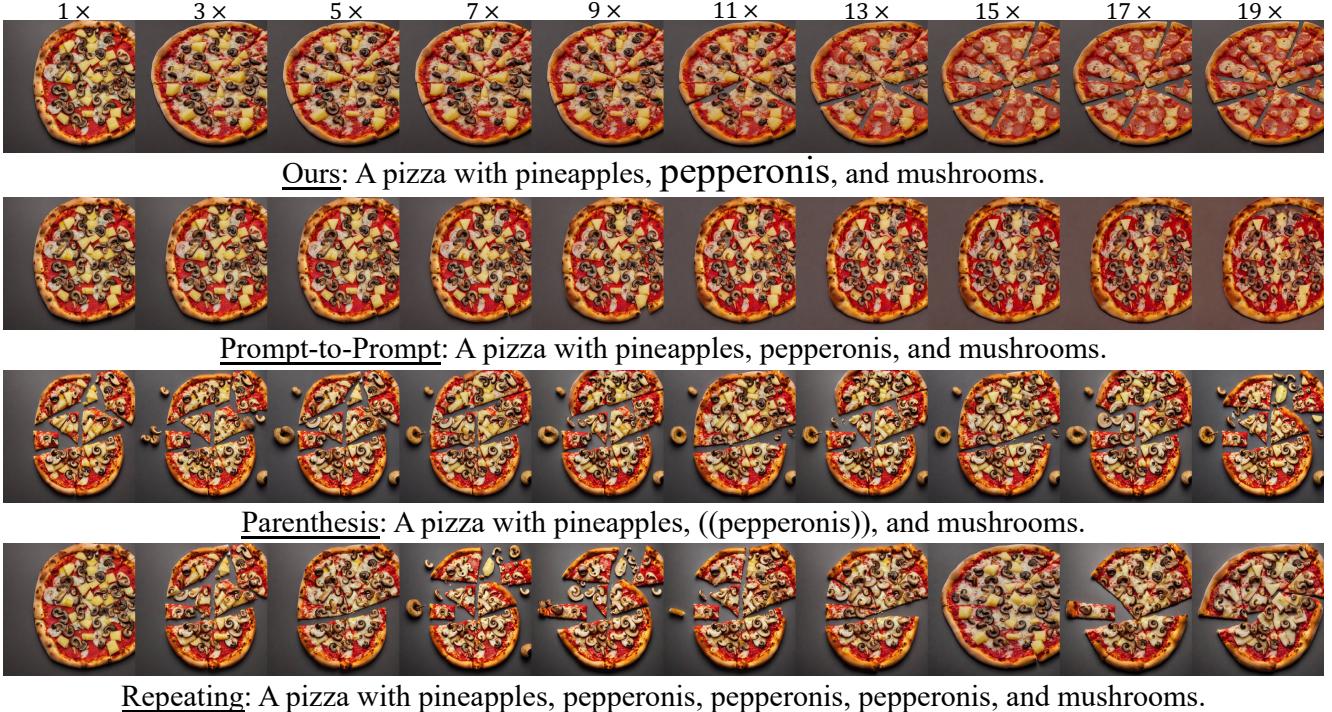


Figure 25. **Additional results of font sizes.** We use a token weight evenly sampled from 1 to 20 for the word ‘pepperoni’ with our method and Prompt-to-Prompt. For parenthesis and repeating, we show results by repeating the word ‘pepperoni’ and adding parentheses to the word ‘pepperoni’ for 1 to 10 times. Prompt-to-Prompt suffers from generating artifacts. Heuristic methods are not effective.

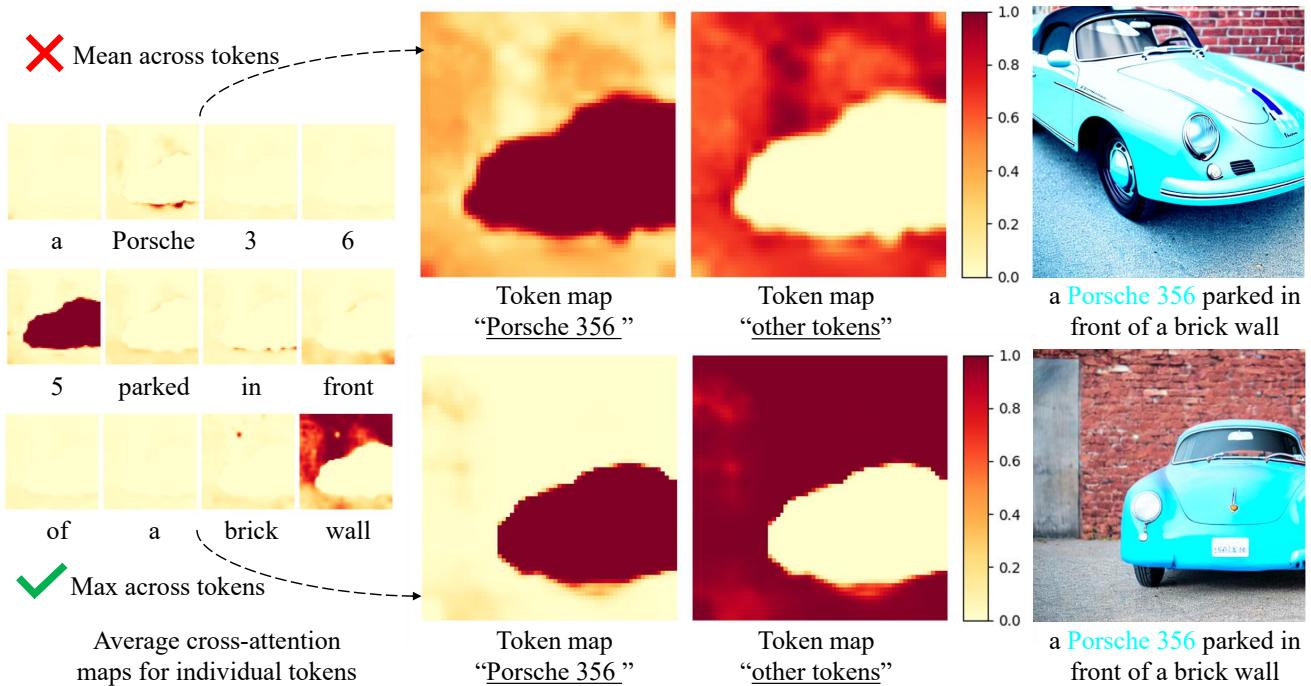


Figure 26. **Mean vs. max attention values for token maps.** To create the *token map*, we take the pixel-wise maximal value across all the tokens inside the span. This generally works better than taking the mean value, the span, especially for the remaining tokens without attributes, contains words like “a” and “of” whose attention maps have small values. Averaging over these tokens could dilute the *token map* of the entire span. As a result, the *token map* may not fully cover the desired region. The inaccurate region leads to changing the color of the floor into cyan.



a cat (Pixel Art) sitting on a meadow (Van Gogh).

Figure 27. **Comparison with a simple composed-based method using different random seeds.** Since the methods like Prompt-to-Prompt [16] cannot generate multiple styles on a single image, one simple idea to fix this is to apply the methods on two regions separately and compose them using the token maps. However, we show that this leads to sharp changes and artifacts at the boundary areas.

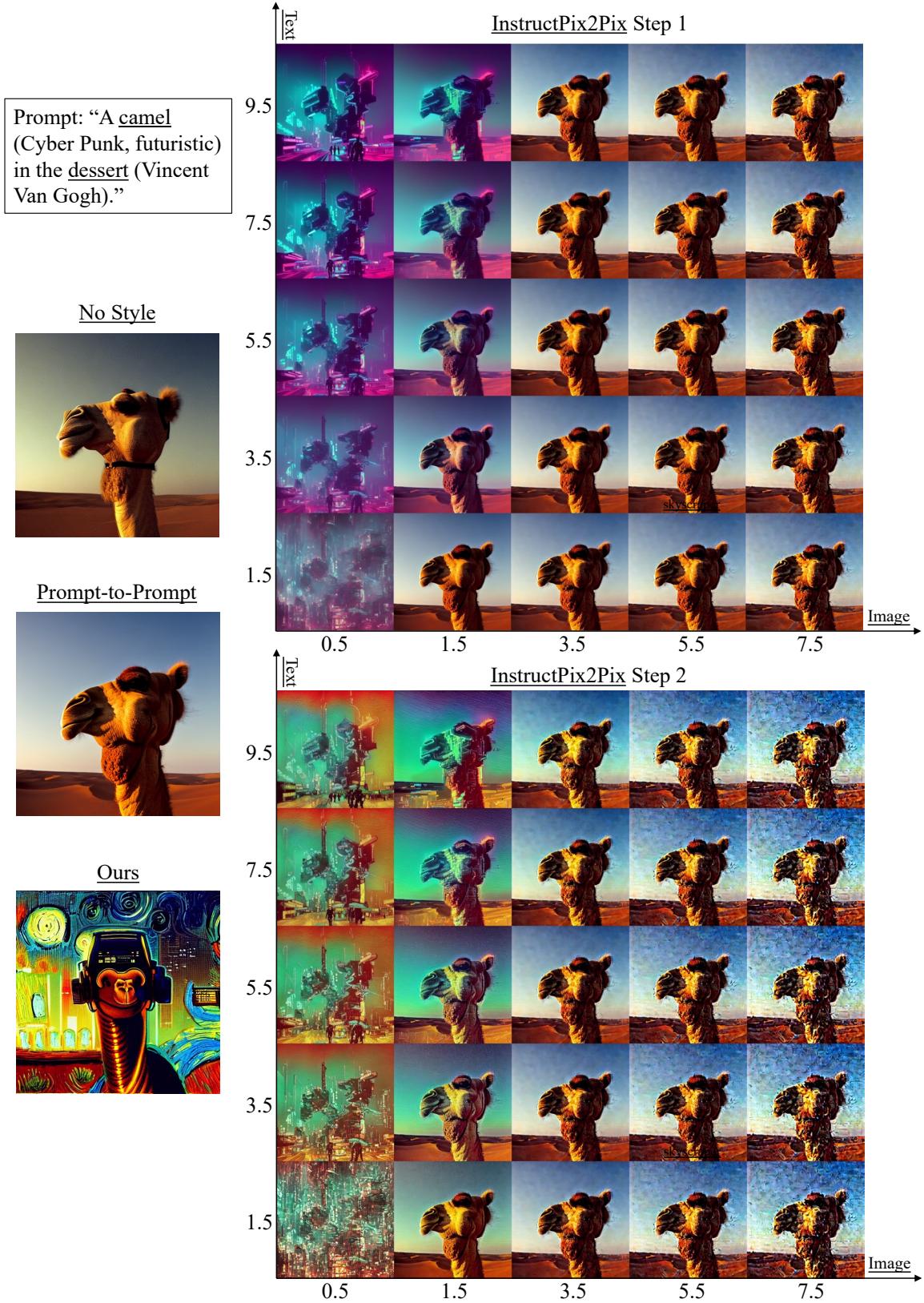


Figure 28. Ablation of the classifier free guidance of InstructPix2Pix. We show that InstructPix2Pix fails to generate both styles with different image and text classifier-free guidance (cfg) weights. When image-cfg is low, the desert is lost after the first editing. We use image-cfg= 1.5 and text-cfg= 7.5 in our experiment.

B. Additional Details

This section details our quantitative evaluation of the font style and font color experiments.

Font style evaluation. To compute the local CLIP scores at each local region to evaluate the stylization quality, we need to create test prompts with multiple objects and styles. We use seven popular styles that people use to describe the artistic styles of the generation, as listed below. Note that for each style, to achieve the best quality, we also include complementary information like the name of famous artists in addition to the style.

```
styles = [
    'Claud Monet, impressionism, oil on canvas',
    'Ukiyoe',
    'Cyber Punk, futuristic, blade runner, william gibson, trending on artstation hq',
    'Pop Art, masterpiece, Andy Warhol',
    'Vincent Van Gogh',
    'Pixel Art, 8 bits, 16 bits',
    'Abstract Cubism, Pablo Picasso'
]
```

We also manually create a set of prompts, where each contains a combination of two objects, for stylization, resulting in 420 prompts in total. We generally confirm that Stable Diffusion [48] can generate the correct combination, as our goal is not to evaluate the compositionality of the generation as in DrawBench [50]. The prompts and the object tokens used for our method are listed below.

```
candidate_prompts = [
    'a beautiful garden in front of a snow mountain':
        ['beautiful garden', 'snow mountain'],
    'a small pond surrounded by skyscraper': ['small pond', 'skyscraper'],
    'a busy city with a huge wave in the background': ['busy city', 'huge wave'],
    'a fountain in front of an elegant castle': ['fountain', 'elegant castle'],
    'a cat sitting on a meadow': ['cat', 'meadow'],
    'a dog walking on a boat': ['dog', 'boat'],
    'a portrait of a man, background is a wheat field':
        ['portrait of a man', 'wheat field'],
    'a lighthouse among the turbulent waves': ['lighthouse', 'turbulent waves'],
    'a stream train on the mountain side': ['stream train', 'mountain side'],
]
```

Font color evaluation. To evaluate precise color generation capacity, we create a set of prompts with colored objects. We divide the potential colors into three levels according to the difficulty of text-to-image generation models to depend on. The *easy-level* color set contains 17 basic color names that these models generally understand. The complete set is as below.

```
COLORS_easy = {
    'brown': [165, 42, 42],
    'red': [255, 0, 0],
    'pink': [253, 108, 158],
    'orange': [255, 165, 0],
    'yellow': [255, 255, 0],
    'purple': [128, 0, 128],
    'green': [0, 128, 0],
    'blue': [0, 0, 255],
    'white': [255, 255, 255],
    'gray': [128, 128, 128],
    'black': [0, 0, 0],
    'crimson': [220, 20, 60],
    'maroon': [128, 0, 0],
```

```

'cyan': [0, 255, 255],
'azure': [240, 255, 255],
'turquoise': [64, 224, 208],
'magenta': [255, 0, 255],
}

```

The *medium-level* set contain color names that are selected from the HTML color names ². These colors are also standard to use for website design. However, their names are less often occurring in the image captions, making interpretation by a text-to-image model challenging. To address this issue, we also append the coarse color category when possible, e.g., “Chocolate” to “Chocolate brown”. The complete list is below.

```

COLORS_medium = {
'Fire Brick red': [178, 34, 34],
'Salmon red': [250, 128, 114],
'Coral orange': [255, 127, 80],
'Tomato orange': [255, 99, 71],
'Peach Puff orange': [255, 218, 185],
'Moccasin orange': [255, 228, 181],
'Goldenrod yellow': [218, 165, 32],
'Olive yellow': [128, 128, 0],
'Gold yellow': [255, 215, 0],
'Lavender purple': [230, 230, 250],
'Indigo purple': [75, 0, 130],
'Thistle purple': [216, 191, 216],
'Plum purple': [221, 160, 221],
'Violet purple': [238, 130, 238],
'Orchid purple': [218, 112, 214],
'Chartreuse green': [127, 255, 0],
'Lawn green': [124, 252, 0],
'Lime green': [50, 205, 50],
'Forest green': [34, 139, 34],
'Spring green': [0, 255, 127],
'Sea green': [46, 139, 87],
'Sky blue': [135, 206, 235],
'Dodger blue': [30, 144, 255],
'Steel blue': [70, 130, 180],
'Navy blue': [0, 0, 128],
'Slate blue': [106, 90, 205],
'Wheat brown': [245, 222, 179],
'Tan brown': [210, 180, 140],
'Peru brown': [205, 133, 63],
'Chocolate brown': [210, 105, 30],
'Sienna brown': [160, 82, 4],
'Floral White': [255, 250, 240],
'Honeydew White': [240, 255, 240],
}

```

The *hard-level* set contains 50 randomly sampled RGB triplets as we aim to generate objects with arbitrary colors indicated in rich texts. For example, the color can be selected by an RGB slider.

```

COLORS_hard = {
'color of RGB values [68, 17, 237]': [68, 17, 237],
'color of RGB values [173, 99, 227]': [173, 99, 227],

```

²https://simple.wikipedia.org/wiki/Web_color

```

'color of RGB values [48, 131, 172]': [48, 131, 172],
'color of RGB values [198, 234, 45]': [198, 234, 45],
'color of RGB values [182, 53, 74]': [182, 53, 74],
'color of RGB values [29, 139, 118]': [29, 139, 118],
'color of RGB values [105, 96, 172]': [105, 96, 172],
'color of RGB values [216, 118, 105]': [216, 118, 105],
'color of RGB values [88, 119, 37]': [88, 119, 37],
'color of RGB values [189, 132, 98]': [189, 132, 98],
'color of RGB values [78, 174, 11]': [78, 174, 11],
'color of RGB values [39, 126, 109]': [39, 126, 109],
'color of RGB values [236, 81, 34]': [236, 81, 34],
'color of RGB values [157, 69, 64]': [157, 69, 64],
'color of RGB values [67, 192, 60]': [67, 192, 60],
'color of RGB values [181, 57, 181]': [181, 57, 181],
'color of RGB values [71, 240, 139]': [71, 240, 139],
'color of RGB values [34, 153, 226]': [34, 153, 226],
'color of RGB values [47, 221, 120]': [47, 221, 120],
'color of RGB values [219, 100, 27]': [219, 100, 27],
'color of RGB values [228, 168, 120]': [228, 168, 120],
'color of RGB values [195, 31, 8]': [195, 31, 8],
'color of RGB values [84, 142, 64]': [84, 142, 64],
'color of RGB values [104, 120, 31]': [104, 120, 31],
'color of RGB values [240, 209, 78]': [240, 209, 78],
'color of RGB values [38, 175, 96]': [38, 175, 96],
'color of RGB values [116, 233, 180]': [116, 233, 180],
'color of RGB values [205, 196, 126]': [205, 196, 126],
'color of RGB values [56, 107, 26]': [56, 107, 26],
'color of RGB values [200, 55, 100]': [200, 55, 100],
'color of RGB values [35, 21, 185]': [35, 21, 185],
'color of RGB values [77, 26, 73]': [77, 26, 73],
'color of RGB values [216, 185, 14]': [216, 185, 14],
'color of RGB values [53, 21, 50]': [53, 21, 50],
'color of RGB values [222, 80, 195]': [222, 80, 195],
'color of RGB values [103, 168, 84]': [103, 168, 84],
'color of RGB values [57, 51, 218]': [57, 51, 218],
'color of RGB values [143, 77, 162]': [143, 77, 162],
'color of RGB values [25, 75, 226]': [25, 75, 226],
'color of RGB values [99, 219, 32]': [99, 219, 32],
'color of RGB values [211, 22, 52]': [211, 22, 52],
'color of RGB values [162, 239, 198]': [162, 239, 198],
'color of RGB values [40, 226, 144]': [40, 226, 144],
'color of RGB values [208, 211, 9]': [208, 211, 9],
'color of RGB values [231, 121, 82]': [231, 121, 82],
'color of RGB values [108, 105, 52]': [108, 105, 52],
'color of RGB values [105, 28, 226]': [105, 28, 226],
'color of RGB values [31, 94, 190]': [31, 94, 190],
'color of RGB values [116, 6, 93]': [116, 6, 93],
'color of RGB values [61, 82, 239]': [61, 82, 239],
}

```

To write a complete prompt, we create a list of 12 objects and simple prompts containing them as below. The objects would naturally exhibit different colors in practice, such as “flower”, “gem”, and “house”.

```

candidate_prompts = [
    'a man wearing a shirt': 'shirt'

```

```

'a woman wearing pants': 'pants'
'a vehicle in the street': 'vehicle'
'a basket of fruit': 'fruit'
'a bowl of vegetable': 'vegetable'
'a flower in a vase': 'flower'
'a beverage on the table': 'beverage'
'a plant in the garden': 'plant'
'a candy on the table': 'candy'
'a toy on the floor': 'toy'
'a gem on the ground': 'gem'
'a church with beautiful landscape in the background': 'church'
]

```

Baseline. We compare our method quantitatively with two strong baselines, Prompt-to-Prompt [16] and InstructPix2Pix [7]. The prompt refinement application of Prompt-to-Prompt allows adding new tokens to the prompt. We use plain text as the base prompt and add color or style to create the modified prompt. InstructPix2Pix [7] allows using instructions to edit the image. We use the image generated by the plain text as the input image and create the instructions using templates “turn the *[object]* into the style of *[style]*,” or “make the color of *[object]* to be *[color]*”. For the stylization experiment, we apply two instructions in both parallel (InstructPix2Pix-para) and sequence (InstructPix2Pix-seq). We tune both methods on a separate set of manually created prompts to find the best hyperparameters. In contrast, it is worth noting that our method *does not* require hyperparameter tuning.

Running time. The inference time of our models depends on the number of attributes added to the rich text since we implement each attribute with an independent diffusion process. In practice, we always use a batch size of 1 to make the code compatible with low-resource devices. In our experiments on an NVIDIA RTX A6000 GPU, each sampling based on the plain text takes around 5.06 seconds, while sampling an image with two styles takes around 8.07 seconds, and sampling an image with our color optimization takes around 13.14 seconds.