IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

CLIPFontDraw: Stylizing Fonts with CLIP

KOTA IZUMI[1], AND KEIJI YANAI[2](Member, IEEE)
[1]The University of Electro-Communications, Tokyo (e-mail: izumi-k@mm.inf.uec.ac.jp)
[2]The University of Electro-Communications, Tokyo (e-mail: yanai@cs.uec.ac.jp)

Corresponding author: Keiji Yanai (e-mail: yanai@cs.uec.ac.jp).

ABSTRACT In this study, we propose a method called "CLIPFontDraw" which enables a user to generate font images stylized according to text input without reference style image. By combining a large-scale multimodal model(CLIP) and a differentiable renderer, our proposed method generates a style matching a short input text and renders font images of individual characters in that style. Furthermore, our approach can also render simple logo patterns with the generated styles. The results of an experimental evaluation of our proposed method show that its output was more readable and expressed the input text more clearly than existing baselines.

INDEX TERMS font style transfer, font synthesis, CLIP, differentiable renderer

I. INTRODUCTION

ALTHOUGH a broad range of artistic fonts is widely used in graphic media such as web pages and advertisements, their designs differ considerably and designing fonts for various character sets, such as those used in Chinese and Japanese, as well as common alphanumeric characters requires a great deal of effort. On the other hand, new methods for generating artistically designed characters have been proposed based on emerging image synthesis technologies using deep learning models. However, existing methods involve some significant limitations, such as depending on prepared reference images as sources for style transfer to a glyph and a corresponding background image. Recently, a method called CLIPstyler [1] was proposed to stylize general images using styles generated from short strings of text written in natural language. Although CLIPstyler can be used to style individual characters, it was not developed for that purpose, and its output may not be satisfactory in terms of graphic design. As shown in Figure 1, given individual characters as input images, CLIPstyler only changed the color of the text, primarily the background, and the outline and texture of the character itself were not significantly stylized according to the prompt. Additionally, because this method lacks a mechanism to distinguish between the foreground and background, when font images are used as input, the style may unintentionally be applied to backgrounds unrelated to the font.

In this study, we propose a method to stylize images of individual characters using only a short text string written in natural language as input. Our proposed approach optimizes the parameters of a set of Bezier curves using a differentiable renderer [2], transforming an input image with a style generated from a prompt written in natural language using the large-scale multimodal model CLIP [3]. Our method represents the font characters as a set of Bezier curves, and by updating only the parameters of the Bezier curves, the style is transferred without changing the background. Furthermore, it enables the generation of more artistic fonts than the result of style transfer with CLIPstyler.

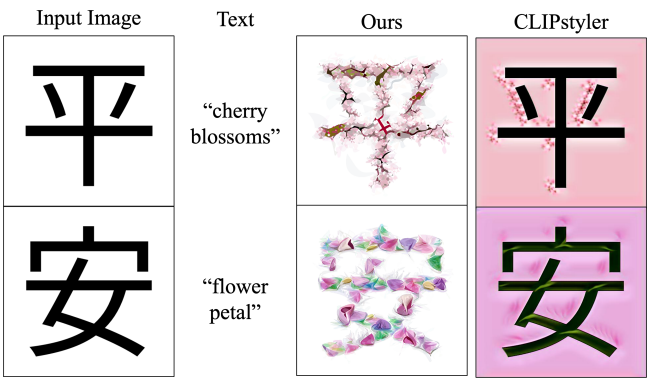

FIGURE 1. Comparison of the proposed method and CLIPstyler [1].

In this paper, we review the methodology of our previous

work [4] and remove loss functions that do not affect the generated results. In addition, we have added comprehensive experiments such as ablation study and user study. Code is available at https://github.com/Squ602/Font_Style_Transfer.

In summary, the contributions of this paper are as follows:

- We introduce CLIPFontDraw, a font style transfer method that uses text as input instead of a style image.
- By combining Bézier curve-based representation with CLIP, our method enables the application of styles specified by text exclusively to the character regions.
- Experiments demonstrate that the proposed method can be applied not only to characters but also to logos, and the degree of character decoration can be adjusted.

II. RELATED WORK

A. FONT STYLE TRANSFER

In contrast to ordinary style transformations, the shapes of characters must be preserved in any stylized font. Atarsaikhan *et al.* [5] introduced a called distance transform loss function for neural style transfer [6] to transfer styles to simple graphics or font elements while preserving the shape of the text characters or logo pattern.

Similarly, a model called Tet-GAN [7] was proposed as a similar approach designed to transfer decoration styles from one image of a character to another image using a deep neural network architecture comprising separate stylization and destylization models. By using these two networks, Tet-GAN can transfer decoration styles between arbitrary characters or delete a style from a given character. However, Tet-GAN is only designed to transfer styles to other characters in the same font. In addition, it can sometimes unintentionally transfer stylistic elements onto the background as well as to individual characters themselves. Therefore, FET-GAN [8] introduces an end-to-end framework to transfer decorations between characters in different fonts and to simple graphic logos.

In addition to font colors and features, graphics with text may include also decorations around the characters. To address this situation, Typography with Decor [9] was proposed as a new deep learning network designed to transfer decorations around characters that would be ignored by previous text stylization methods. The model network learns to separate, transfer, and resynthesize decorations, which enables it to transfer styles that include decorations around characters.

Conventional font style conversion methods do not allow adjustment of the degree of decoration of a given text. In contrast, the Shape-Matching GAN [10] framework can adjust the degree of decoration applied to individual characters. The model controllable in terms of scale and does not a training dataset require. That is, only an undecorated font image and a single style image for reference are needed to transfer the style to the text. In addition, Intelligent Typography [11] was proposed as a new framework for transferring more complex, scale-controllable style images to text.

However, all of these font style conversion methods require a reference image as the source of the style to be transferred, which may not always be available in practice, especially for styles that a designer might simply imagine. In contrast, our proposed method converts the style of characters or glyphs that could be used in a font or graphic without the need for reference images of the desired style.

B. TEXT-GUIDED SYNTHESIS

Recently, OpenAI provided a method for manipulating images with text using CLIP [3]. CLIP encodes images and text, learning to maximize the cosine similarity of each pair of embedded representations while minimizing that of incorrect or inaccurate pairings. The large number of images and texts used in training enables the model to compute similarity for a wide variety of images and texts.

Transforming images using CLIP has recently become a popular topic of active research. StyleCLIP [12] can edit images based on a text prompt by optimizing latent variables in a StyleGAN [13] model using CLIP. Furthermore, StyleGAN-NADA [14] introduced directional CLIP loss as a loss function using CLIP. This loss is used to modify the model based on text input to convert images to be converted to different subject or style domains without additional training data. Similarly, VQGAN-CLIP [15] can generate images according to a text prompt by optimizing the latent code of a VQGAN [16] model with CLIP. In this study, we also use CLIP for optimization. However, these existing methods rely heavily on pre-trained models, and the generated images are limited to the domain of the pre-trained data.

CLIPstyler [1] takes an image and a text string as input applies a style transfer to match the text without pre-training or obtaining styles from specific source images. The method optimizes the parameters of a lightweight convolutional network for each input. In addition to the directional CLIP loss proposed in StyleGAN-NADA, the authors introduced a new loss function for texture transfer called PatchCLIP loss, which computes the directional CLIP loss after a sufficient number of random slices of the image and applies perspective augmentation to each slice. These loss functions enable to the model to transfer styles that reflect the meaning of the input text down to the finest details of the image.

Therefore, we also adopt the loss function introduced in CLIPstyler to apply a texture matching a prompt to an input image of a character without pre-training.

CLIP has also been applied in vector graphics. In vector graphics formats the position of points, lines connecting points, colors, transparency, and so forth are recorded as numerical values. CLIPDraw [17] combines a differentiable renderer [2] with CLIP to generate a drawing from text simply by optimizing the parameters of a set of Bezier curves. In addition, StyleCLIPDraw [18] was proposed to transfer the style of a given reference image to drawing generated by CLIPDraw.

In the proposed approach, we combine CLIP with a vector representation. Specifically, a single character is represented

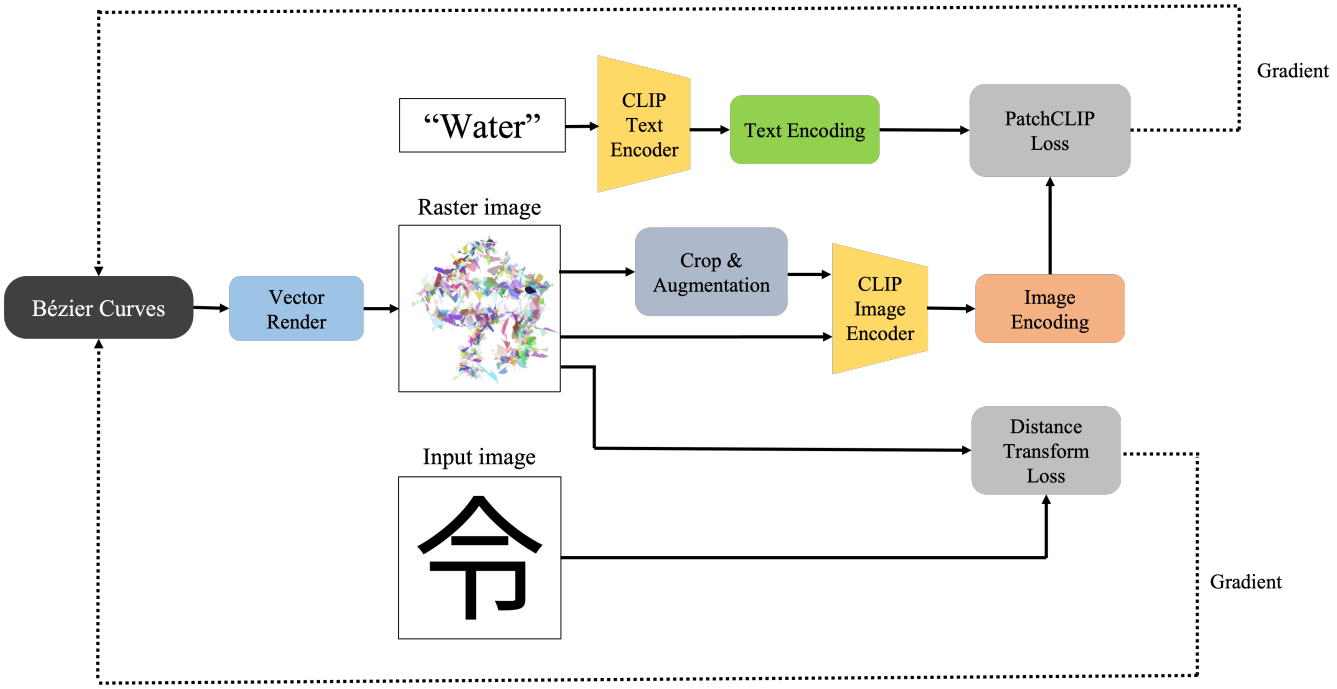

FIGURE 2. Overview of the proposed method. Overview of the proposed method: Bézier curves are rendered into raster images using a differentiable renderer. Two loss functions are then computed based on the rendered raster images: the Distance Transform Loss, which preserves the shape of the characters, and the PatchCLIPLoss, which measures the semantic similarity to the text. The parameters of the Bézier curves are updated based on these losses. By iteratively repeating this process, the characters are gradually transformed to match the style specified by the text prompt.

by a set of Bezier curves, the parameters of which are optimized using CLIP.

CLIPFont [19] has also been proposed as a font generation method using vector representation and CLIP. Although CLIPFont is similar to the proposed approach, it differs in that it takes a raster image as input and produces a large number of textures and patterns for a single character that match an input the prompt.

III. METHOD

Our proposed method is designed to apply a style generated from a text prompt written in natural language instead of using pre-selected reference images as a source for the style. Although CLIPstyler [1] uses a lightweight convolutional network to apply generated styles in raster format, this method does not generalize well to images of characters or glyphs that could be used in a font. Therefore, in this study, we adopt a vector format. In our proposed approach, characters are represented by a set of Bezier curves, and their parameters are gradually updated to match the prompts. The operation of the method is illustrated in Figure 2. The drawing is represented by a set of Bezier curves with control point positions, colors, and line thicknesses as parameters, rather than raster images represented by pixels.

First, the initial positions of the control points are determined based on the input image, and Bezier curves with random colors are placed on the canvas. Next, the vector drawing is converted to a raster image using a differentiable renderer. Random cropping and random perspective augmentation operations are then performed on the raster image, and the loss is calculated using the CLIP [3] image encoder and text encoder. The computed losses are propagated to minimize the total loss function, and the parameters of the Bezier curves are directly updated the through differential renderer. The PatchCLIP loss introduced in CLIPstyler [1] is applied and the distance transform loss proposed by Atarsaikhan *et al.* [5]. The role of each loss function is indicated below.

1) PatchCLIP Loss

The PatchCLIP loss function reflects the meaning of the text in the detailed parts of the font image.

2) Distance Transform Loss

The distance transform loss preserves the shape of the font.

A. DRAWING REPRESENTATION

The drawing is represented by a set of closed cubic Bezier curves as shown in Figure 3 based on the method provided by Li *et al.* [2]. Each curve is represented by the position of a control point, line thickness, color, and opacity. Closed Bézier curves are generated with three to five segments per shape, and their interiors are filled. The number of curves and control points are fixed during optimization and the position, color, and opacity of the control points are optimized via gradient descent. The initial position of the control points is centered on a point randomly sampled from the coordinates of the black area in the text image based on the number of closed Bezier curves to be drawn, as shown in Figure 4(b).

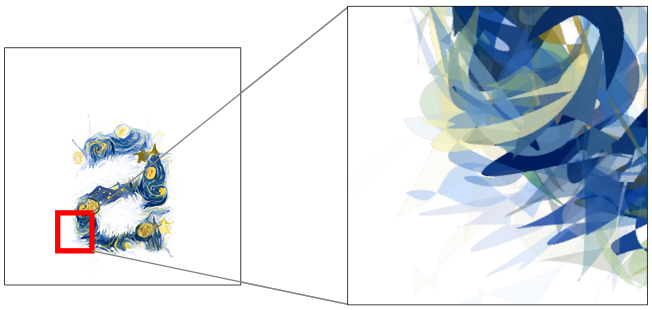

FIGURE 3. Drawing representation of our method.

(a) (b) (c)

FIGURE 4. (a) Input image (b) The initial positions of Bézier curves. Control points are distributed near these points. (c) The distance transform map. The further away from the text area, the larger the value.

This is done if a random position were used as the initial position, the Bezier curve would also be drawn on the background.

B. DISTANCE TRANSFORM LOSS

We considered that, the background of the rendered images should remain white as in, the input font image to maintain sufficient to read ability. Therefore, we apply the distance transform loss introduced by Atarsaikhan *et al.*. To calculate Distance Transform Loss, a distance transform is first performed on the input image to create a distance transform image. Figure 4(c) shows an example of a distance transform image in which the value of each pixel represents the squared distance to the nearest font pixel. Note that the value of the pixels over the input fonts is zero.

The distance transform images are then multiplied by the input and output images, and the mean squared error is calculated. The distance transform loss can be defined by:

$$L_{distance} = \frac{1}{2}\left(I_c \circ I_d - I_{draw} \circ I_d\right)^2, \tag{1}$$

where I_d, I_c, and I_{draw} denote the distance transformed image, the input image, and the raster image rendered using the differentiable renderer, respectively. The operator $\circ$ indicates taking the element-wise product of two matrices of the same size.

C. PATCHCLIP LOSS

CLIPstyler [1] introduced a new loss for texture transfer, PatchCLIP loss, inspired by Gatys *et al.* [6] and Frans *et al.* [17]. This loss function enables generated or transfered styles to be rendered down to the finest details of the image. Our method also adopts this loss function. Specifically, a sufficient number of raster images are cropped from a raster image that has also been transformed using a differentiable renderer, and random perspective augmentation is performed on the cropped patches to calculate the directional CLIP loss. Directional CLIP loss is a loss function proposed by StyleGAN-NADA [14] and is calculated as follows: (1) extract the feature embedding of the input and generated images, and compute the difference between them. (2) extract the feature embeddings of the input text and the predetermined text embedding, and computer the difference between them. (3) compute the cosine similarity of both the differences.

A threshold τ is set to reject gradient optimization for high-scoring patches. Thus, PatchCLIP loss can be defined as:

$$\begin{aligned}
\Delta T &= E_T(t_{sty}) - E_T(t_{src}),\\
\Delta I &= E_I(\mathrm{aug}(\hat{I}^i_{cs})) - E_I(I_c),\\
l^i_{patch} &= 1 - \frac{\Delta I \cdot \Delta T}{|\Delta I||\Delta T|}\\
L_{patch} &= \frac{1}{N}\sum_i^N R(l^i_{patch}, \tau),
\end{aligned} \tag{2}$$

$$\text{where } R(s,\tau) = \begin{cases} 0, & \text{if } s \le \tau \\ s, & \text{otherwise} \end{cases}$$

where E_T and E_I are the text and image encoders of CLIP, respectively. t_{sty} is the input text, t_{src} is a text determined in advance. $\hat{I}^i_{cs}$ is the i-th patch cut from the raster image rendered using the differentiable renderer and I_c is the input image. aug denotes a random perspective augmentation.

D. TOTAL LOSS

CLIPFontDraw uses two different losses for optimization. PatchCLIP loss transfers the style to the character according to the input text prompt, while distance transform loss preserves the shape of characters. Thus, the loss function during optimization can be defined as:

$$L_{total} = \lambda_p L_{patch} + \lambda_{distance} L_{distance} \tag{3}$$

IV. RESULTS

A. EXPERIMENTAL SETTINGS

In our experimental evaluation, the resolution of the input image was 512×512 pixels. Distance-transformed images were created using functions in the OpenCV library. The weightsλ_p, $\lambda_{distance}$ were set to 9×10^3, 1.5×10^3 based on empirical observations. The weights of PatchCLIP loss was set with reference to CLIPstyler [1]. The weights of the distance transform loss were set to values that reflected the style well and resulted in the best recognizabilty of the characters.

The effect of the loss weights is shown below in section IV-E. The Adam optimizer was used as the optimization

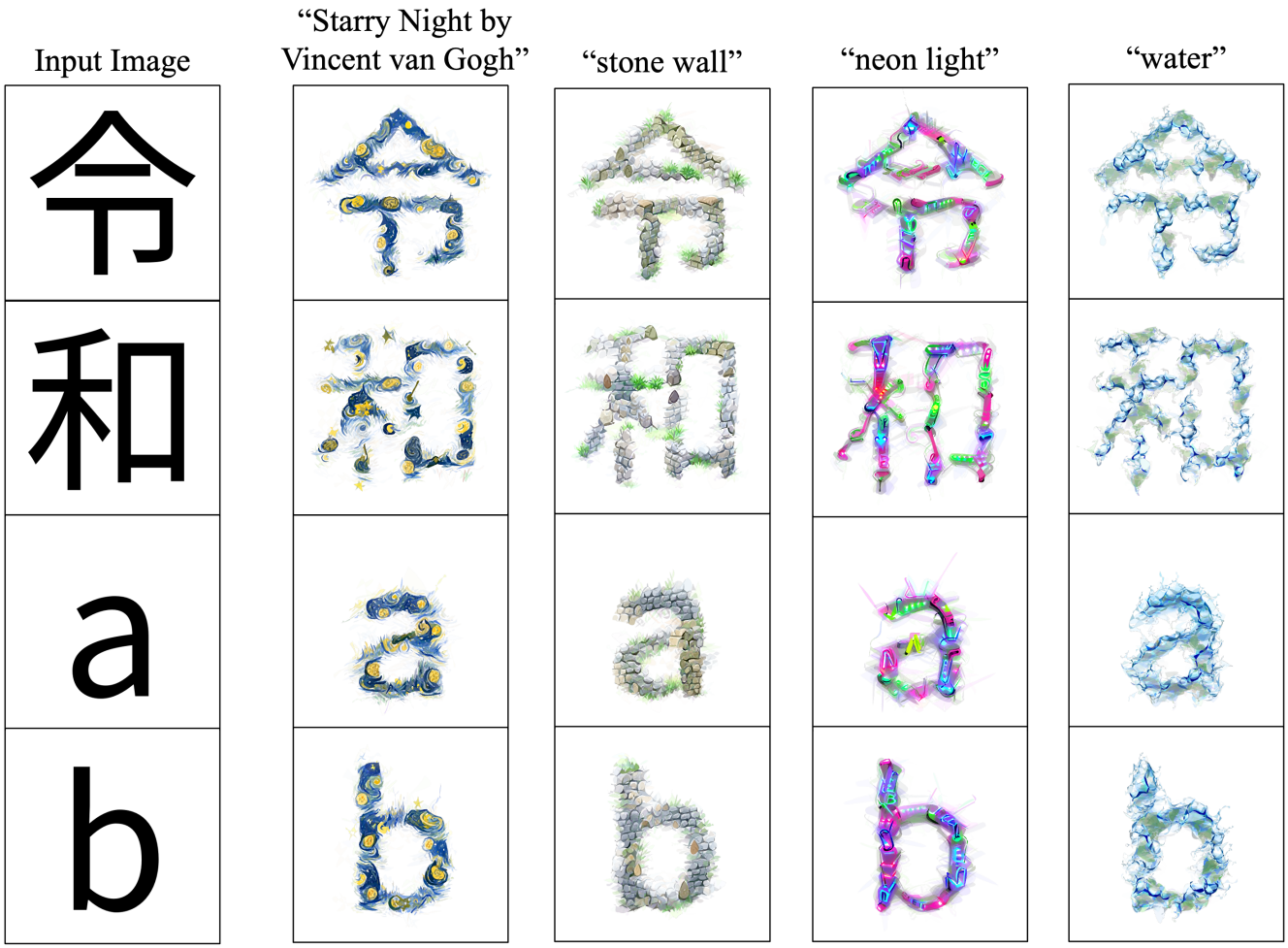

FIGURE 5. The result of style transfer of a font image using the proposed method.

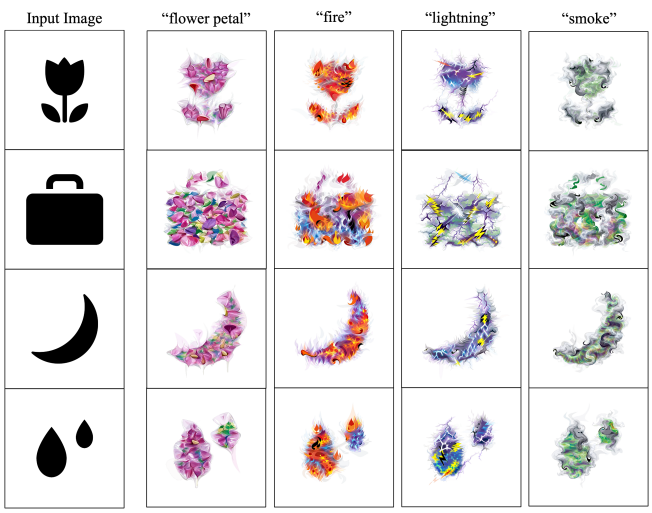

FIGURE 6. A simple logo used as an input image.

function, and the learning rate was set to different values of 1 and 1×10^{-2} for the position and color (including opacity) of the control points, respectively. The number of iterations was set to 200. The size of the cropped image was set as 160×160, which provided the best quality. The number of cropped images was set to 64. We used a function in the PyTorch framework to perform a random perspective augmentation, and the distortion scale was set to 0.5. The threshold τ was set to 0.7. We also used prompt engineering as proposed by Radford *et al.*. Prompt engineering methods were used to create a sentence from a word using a template such as “a photo of `{text}`” as input to CLIP. 512 closed Bézier curves were used to represent each character. The process of generating the stylized characters took about 60 to 80 seconds for each image using an NVIDIA GeForce RTX 2080 Ti.

B. QUALITATIVE EVALUATION

Some examples of the output of our proposed method are shown in Figure 5. It may be observed that input characters are rendered in a style that matches the input text prompt. For example, given the text prompt is “Starry Night by Vincent van Gogh”, the model stylized the input characters with artistic representations of the moon, sky, and stars that re-

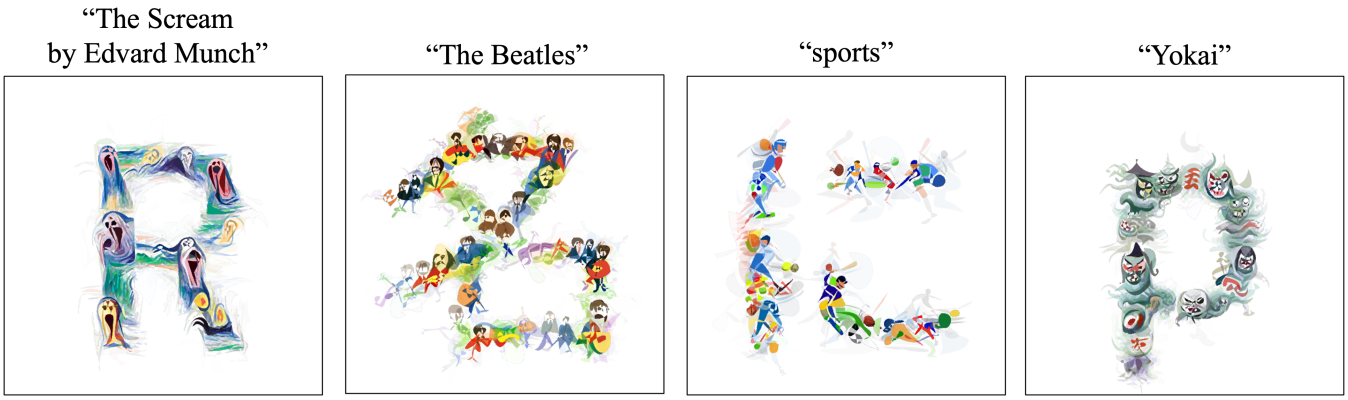

FIGURE 7. Further results.

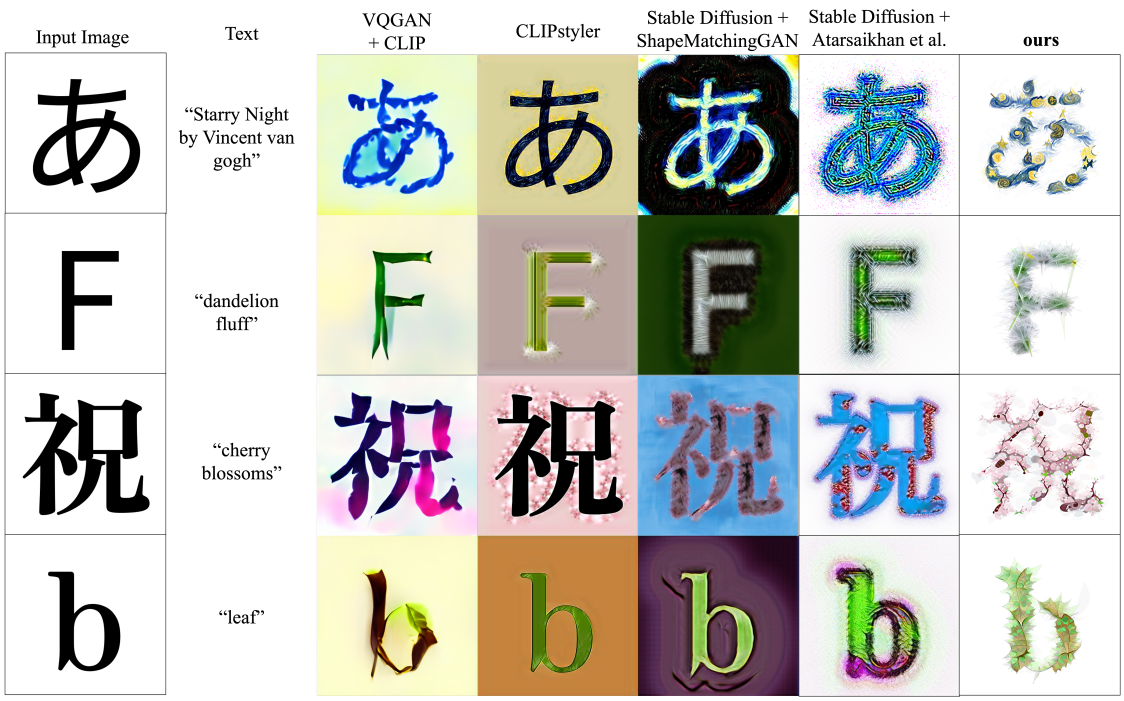

FIGURE 8. Comparison with other methods.

semble the original famous painting. These results also show that the characters were rendered without a background. Furthermore, for example, for the input prompt "stone wall", the stylized characters are rendered with a pattern resembling a stone wall with coloring that artistically expresses that concept. Furthermore, color and shape variations that look like plants growing between the stones also appeared to a greater or lesser degree in the output characters.

Figure 6 shows the results when a simple logo was used as the input image. It may be observed that the generated styles can be applied not only to the image of text characters but also to simple silhouettes. Figure 7 shows the results when "The Scream by Edvard Munch" and "The Beatles" were entered as input prompts to express a font style. It may be observed that faces like that appearing in the famous painting and faces that resemble members of the Beatles respectively appear in the stylized characters. Moreover, when the prompt string "sport" was used, the output included shapes resembling pictograms showing people exercising or in action poses expressing human movement appears. When "Yokai" (Japanese Ghost) was used, the characters was included stylized depictions of Japanese ghosts. From these results, it may be observed that the proposed method produces results that cannot be generated by conventional techniques that rely on separate input image as style references.

C. COMPARISON WITH EXISTING METHODS

We compared the performance of our method to that of other similar style transfer methods using natural language. For comparison, we used a VQGAN+CLIP[1] [15] model, which optimizes the latent code of pre-trained VQGAN [16] using CLIPstyler [1] and CLIP [3]. We also considered style transfer methods such as Shape-Matching GAN [10] and the method provided by Atarsaikhan *et al.* [5]. However, given that these methods require style images as a reference, they are not directly comparable to the proposed approach. Therefore, we generated style images using Stable Diffusion, which is currently attracting attention as a method for generating images from natural language, and compared the results obtained by using these generated images as inputs to the font style transfer methods. Stable Diffusion is a trained latent diffusion [20] trained on the large dataset LAION-5B [21]. Reference images for use as style sources were selected from among the generated images to include as clear background and foreground areas as possible to enable the style to be transferred to the font images more easily.

The results of the comparison with other methods are shown in Figure 8. The two columns at left show the input text prompt and the image of the character to be rendered. The five columns at right show the results generated by each method. VQGAN+CLIP produced text that is difficult to read because the shapes of the characters are distorted. CLIPstyler preserved the shapes of the font, but the meaning of the text was not expressed well in the generated font images. Stable Diffusion+Shape Matching GAN reflected the meaning of the text, but did not express the prompt graphically as a texture applied to the character. Stable Diffusion+Atarsaikhan *et al.* did express the meaning of the text in the generated images, but only to a very limited extent, and the quality of the images was poor.

In contrast, our method does not apply the style to the background at all, and the stylized characters expressed the text prompt more clearly artistically than the other methods.

Effect of adjectives

The results of adding different adjectives to the text for "stone" are shown in Figure 9. The result with "red stone" mainly changed the color of the result generated with the prompt "stone" , as one would expect from the meaning of the words in natural language. When the adjective "smooth" was added to the prompt, the stylized stone was rendered with a perceptibly smoother texture. Furthermore, the prompt "stone wall" yielded characters with subtle stylized depictions of plants growing on a stone wall, which artistically expresses the prompt in a way that should be intuitively clear to the viewer. Thus, these results show that adding adjectives to the text allowed us to specify more details of the style based on intuitive expectations from the meaning of the prompt in natural language.

[1]https://github.com/nerdyrodent/VQGAN-CLIP

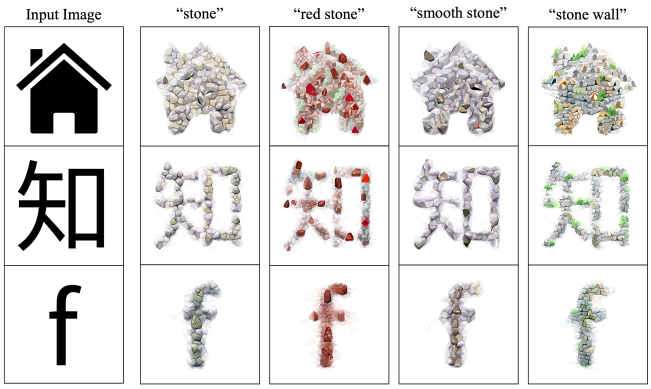

FIGURE 9. Result of using text prompts with different adjectives.

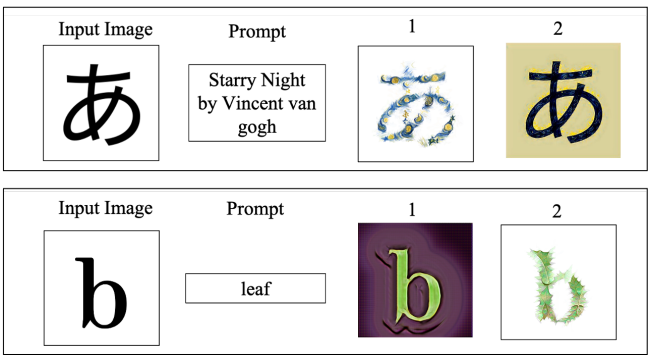

FIGURE 10. Image used for user evaluation.

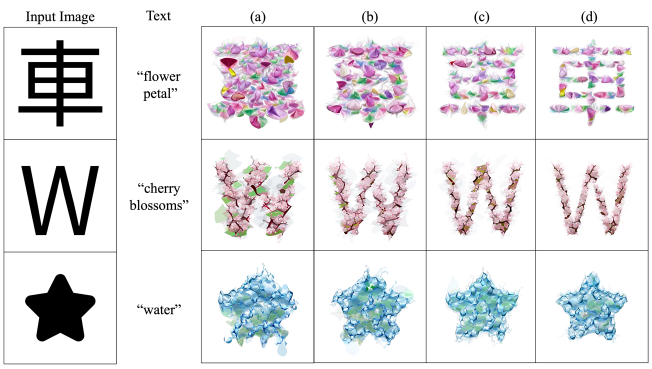

FIGURE 11. Result of changing the weight of $\lambda_{distance}$. (a) $\lambda_{distance} = 1.5 \times 10$. (b) $\lambda_{distance} = 1.5 \times 10^2$. (c) $\lambda_{distance} = 1.5 \times 10^3$. (d) $\lambda_{distance} = 1.5 \times 10^4$.

TABLE 1. The average preference ratio at which this method is chosen over existing methods.

	Ratio
VQGAN+CLIP vs ours.	0.913
CLIPstyler vs ours.	0.797
Stable Diffusion + ShapeMatchingGAN vs ours	0.710
Stable Diffusion + Atarsaikhan *et al.* vs ours	0.783

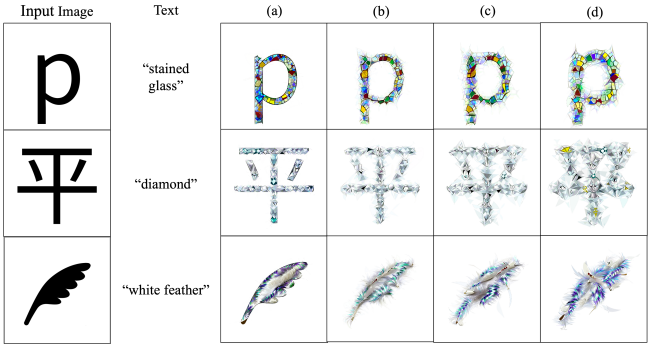

FIGURE 12. Result of changing the weight of λ_{patch}. (a) $\lambda_{patch} = 9.0$. (b) $\lambda_{patch} = 9.0 \times 10^2$. (c) $\lambda_{patch} = 9.0 \times 10^3$. (d) $\lambda_{patch} = 5.0 \times 10^4$.

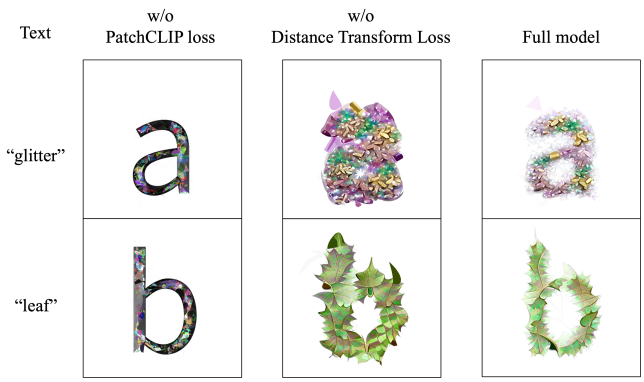

FIGURE 13. Results when the loss function was removed.

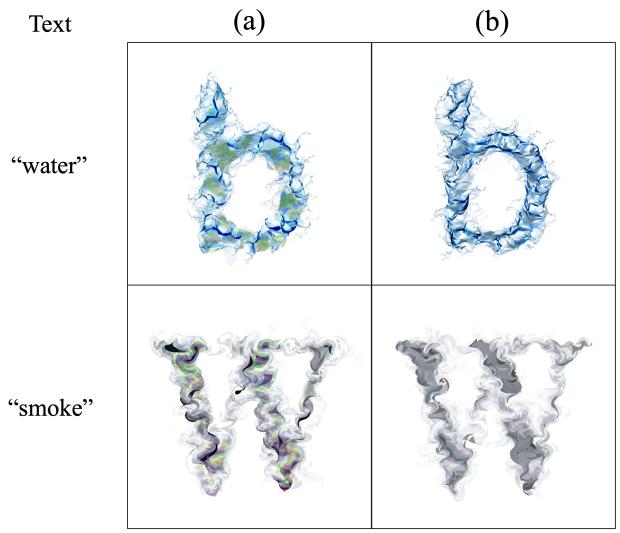

FIGURE 14. Results when the set of Bezier curves was initialized with a specific color. (a) initialized with a random color, (b) "water" initialized in blue, "smoke" in gray.

D. USER STUDY

We also performed a user evaluation using our proposed approach this method and the other methods. Six different pairs of character images and text prompts were input to each method and six output images were obtained. 30 images in total were used for evaluation to obtain results from the four methods and our method. The subjects were shown an image from one method and an image from another, the input image, and the text, as shown in Figure 10, and were asked which method followed the prompt better, especially considering how easily one could identify the input prompt text based on the rendered output. The questions were asked of 23 users. Note that for the user study, we used the results generated by our previous method [4]. Although the loss function of our previous method differs from that of this paper, there is little difference in the generated results.

The existing methods were compared with our approach, and the average rate of preference rate of our method is shown in Table 1. The results show that our method was preferred in over 70% of the responses provided, which quantitatively verifies the superior performance of the proposed method.

E. ABLATION STUDIES

Weight change of distance transform loss

The results of applying different weights to the distance transform loss are shown in Figure 11. The larger the weight, the closer the result matches the contour thickness of the input text image, while the smaller the weight, the more the Bezier curve deviates from the text' s contour. Therefore, it should be possible to adjust the degree of character decoration by adjusting the weights. Of note, small weights may lead to rendered characters that are difficult to recognize for complex shapes. For example, in the case of the letter "W" in Figure 11, when the weights were small, the rendered character with the generated style is more difficult to read. When characters with many strokes are used as input images, increasing the weights may help to preserve the outlines of the characters and generate more easily readable results.

Weight change of PatchCLIP Loss

The results of changing the weight of PatchCLIP loss are shown in Figure 12. Reducing the weights as in (a) would reduce the quality of the texture pattern and make the colors unnatural. On the other hand, when the weights are increased, the pattern appearing in the text becomes larger, resulting in thicker text. Thus, it can be seen that increasing the weight of PatchCLIP loss has a similar effect as when the weight of distance transform loss is reduced.

Effect of Loss function

Figure 13 compares the results of the proposed method with those of the cases without the PatchCLIP and distance transform loss functions, respectively. Without PatchCLIP loss, characters are not stylized well. Without the distance transform loss, the outline of the character became larger, making the character difficult to recognize.

FIGURE 15. Animation of the generation process. The leftmost output was the initial output, and the process of optimizing the Bezier curves for the styles generated from the prompts "Starry Night by Vincent van Gogh" and "Sunflowers by Vincent van Gogh" is shown. By saving all the intermediate outputs as images, the process can be animated as a continuous transformation.

FIGURE 16. Result of rendering the word "fire" with stylized fonts generated with different distance transform loss weights. The prompt was "red fire".(a) $\lambda_{distance} = 1.5 \times 10^5$. (b) $\lambda_{distance} = 1.5 \times 10^2$.

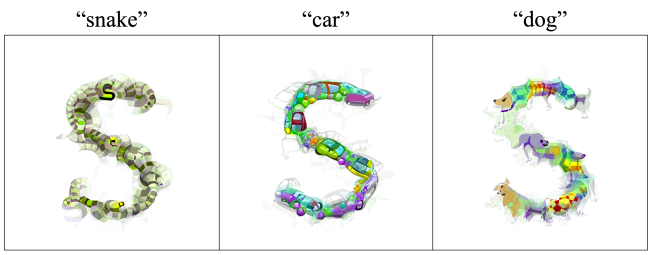

FIGURE 17. Failure cases when the style is not reflected well.

F. APPLICATIONS

Initial color of Bezier curves

In contrast to a simplistic interpretation of the text prompt, the generated style is rendered with green as well as for the prompt "water", as shown in Figure 5. This may be attributed to color of the Bezier curves being initialized randomly. Therefore, we found that by initializing the color of the Bezier curves with a specific color, it was possible to prevent undesired colors from appearing in the output. Figure 14 (a) shows the results when the color of the Bezier curves was initialized randomly, and Figure 14 (b) shows the results when the color was initialized according with a specific value. The results show that initializing the Bezier curve with a single color eliminate colors such as green that did not fit the interpretation of the prompt well.

Creating Animations

The output of the generation process can be animated by connecting all of the results. Furthermore, after optimizing a given character for a single prompt, we can create an animation in which the character that changes from one style to another successively by optimizing another prompt while preserving the Bezier curve information. Figure 15 shows the process of changing the style of a character from "Starry Night by Vincent van Gogh " to "Sunflowers by Vincent van Gogh". The Bezier curve starts with random colors and is gradually transformed into the "stone" style. Then, the shape and color of the character change again, as the style of "Sunflowers by Vincent van Gogh" is applied.

Generating fonts with different impressions

The weight of the distance transform loss can substantially change the appearance of the generated font. In Figure 16, we show the results of changing the magnitude of the weights. As shown in Figure 16 (a), a large weight only changed the color of the font, while a smaller weight as shown in Figure 16 (b) is rendered in a more flamboyant style that gives a strong impression of more active burning. These multiple characters were generated individually, cropped, and merged.

G. LIMITATIONS

This method is good for reflecting such as objects having no fixed shapes "water" and "fire" or painter's style. However, since the loss is calculated on the clipped image, some

prompts may not produce the expected results. For example, as shown in Figure17, for a prompt "snake," something like a pattern of snakes was generated, but a single snake did not appear as the letter S. Also, if an object or animal is specified as a prompt, it sometimes cannot be reflected as a style well. These issues could potentially be avoided by using models specialized for image generation, such as Diffusion Models, instead of CLIP.

V. CONCLUSIONS

In this study, we have proposed a method for stylizing characters or glyphs that can comprise graphic fonts based on input text strings written in natural language. The proposed method, unlike traditional font style transfer methods, allows for style transfer without the need for a style image by specifying the style through text. Our experimental results also confirmed that the method is effective for simple logos. Furthermore, we have presented the results of an ablation study showing that our approach can be used to express various textures depending on the input text prompt and that it can be used for complex characters by increasing the weight of the loss function. Reducing the time required for optimization remains as a topic for future research.

ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Numbers, 22H00540, 22H00548, and 22K19808.

.

APPENDIX A ADDITIONAL RESULTS

Examples of further results and adjustments to the results with adjectives are shown in Figure 18 and Figure 19.

REFERENCES

[1] Kwon Gihyun and Ye Jong Chul. Clipstyler: Image style transfer with a single text condition. Proc. of the CVF/IEEE Conference on Computer Vision and Pattern Recognition, pages 18062–18071, 2022.

[2] Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. ACM Transactions on Graphics (Proc. SIGGRAPH Asia), 39(6):193:1–193:15, 2020.

[3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Jack Clark Pamela Mishkin, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Proc.of the 38th International Conference on Machine Learning, volume 139, pages 8748–8763, 2021.

[4] Kota Izumi and Keiji Yanai. Zero-shot font style transfer with a differentiable renderer. In Proc. of the 4th ACM International Conference on Multimedia in Asia, number 32, pages 1–5, 2022.

[5] Gantugs Atarsaikhan, Brian Kenji Iwana, and Seiichi Uchida. Contained neural style transfer for decorated logo generation. In Proc. of 13th IAPR International Workshop on Document Analysis Systems, pages 317–322, 2018.

[6] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In Proc. of the CVF/IEEE Conference on Computer Vision and Pattern Recognition, pages 2414–2423, 2016.

[7] S.Yang, J.Liu, W.Wang, and Z.Guo. TET-GAN:text effects transfer via stylization and destylization. In Proc. of the AAAI Conference on Artificial Intelligence, volume 34, pages 1238–1245, 2019.

[8] Wei Li, Yongxing He, Yanwei Qi, Zejian Li, and Yongchuan Tang. FET-GAN: Font and effect transfer via k-shot adaptive instance normalization. In Proc.of the AAAI Conference on Artificial Intelligence, volume 34, pages 1717–1724, 2020.

[9] Wenjing Wang, Jiaying Liu, Shuai Yang, and Zongming Guo. Typography with decor: Intelligent text style transfer. In Proc. of the CVF/IEEE Conference on Computer Vision and Pattern Recognition, pages 5889–5897, 2019.

[10] Shuai Yang, Zhangyang Wang, Zhaowen Wang, Ning Xu, Jiaying Liu, and Zongming Guo. Controllable artistic text style transfer via shape-matching gan. In Proc. of the CVF/IEEE International Conference on Computer Vision, pages 4442–4451, 2019.

[11] Wendong Mao, Shuai Yang, Huihong Shi, Jiaying Liu, and Zhongfeng Wang. Intelligent typography: Artistic text style transfer for complex texture and structure. IEEE Transactions on Multimedia, pages 1–15, 2022.

[12] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. StyleCLIP: Text-driven manipulation of stylegan imagery. arXiv preprint arXiv:2103.17249, 2021.

[13] Tero Karras, Samuli Laine, , and Timo Aila. A style-based generator architecture for generative adversarial networks. In Proc. of the CVF/IEEE Conference on Computer Vision and Pattern Recognition, pages 4401–4410, 2019.

[14] Rinon Gal, Or Patashnik, Haggai Maron, Amit H. BERMANO, Gal Chechik, and Daniel Cohen-Or. Stylegan-NADA: Clip-guided domain adaptation of image generators. ACM Transactions on Graphics, 41(141):1–13, 2022.

[15] Katherine Crowson, Stella Biderman, Daniel Kornis, Dashiell Stander, Eric Hallahan, Louis Castricato, and Edward Raf. VQGAN-CLIP: Open domain image generation and editing with natural language guidance. In Proc. of European Conference on Computer Vision, 2022.

[16] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In Proc. of the CVF/IEEE Conference on Computer Vision and Pattern Recognition, pages 12873–12883, 2021.

[17] Kevin Frans, LB Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. arXiv preprint arXiv:2106.14843, 2021.

[18] Peter Schaldenbrand, Zhixuan Liu, and Jean Oh. StyleCLIPDraw: Coupling content and style in text-to-drawing translation. arXiv preprint arXiv:2202.12362, 2021.

[19] Yiren Song and Yuxuan Zhang. CLIPFont: Text guided vector wordart generation. In Proc. of British Machine Vision Conference, 2022.

[20] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In Proc. of the CVF/IEEE Conference on Computer Vision and Pattern Recognition, 2022.

[21] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Ross Wightman Cade Gordon, Theo Coombes Mehdi Cherti, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5B: An open large-scale dataset for training next generation image-text models. In Neural Information Processing Systems, 2022.

KOTA IZUMI received B.Eng. degrees from Cluster of Informatics and Computer Engineering, the University of Electro-Communications Tokyo, Japan, in 2023. He is now a master course student at the Department of Informatics, the University of Electro-Communications, Tokyo, Japan. He is working on font style transfer.

KEIJI YANAI is a professor at Department of Informatics, the University of Electro-Communications, Tokyo, Japan. He received B.Eng., M.Eng. and D.Eng. degrees from the University of Tokyo in 1995, 1997 and 2003, respectively. From 1997 to 2006 he was a research associate and until 2015 he was an associate professor in the Department of Computer Science, the University of Electro-Communications, Tokyo. From November, 2003 to September, 2004, he was a visiting scholar at the Department of Computer Science, University of Arizona, USA. His recent research interests include computer vision, deep learning and multimedia processing.

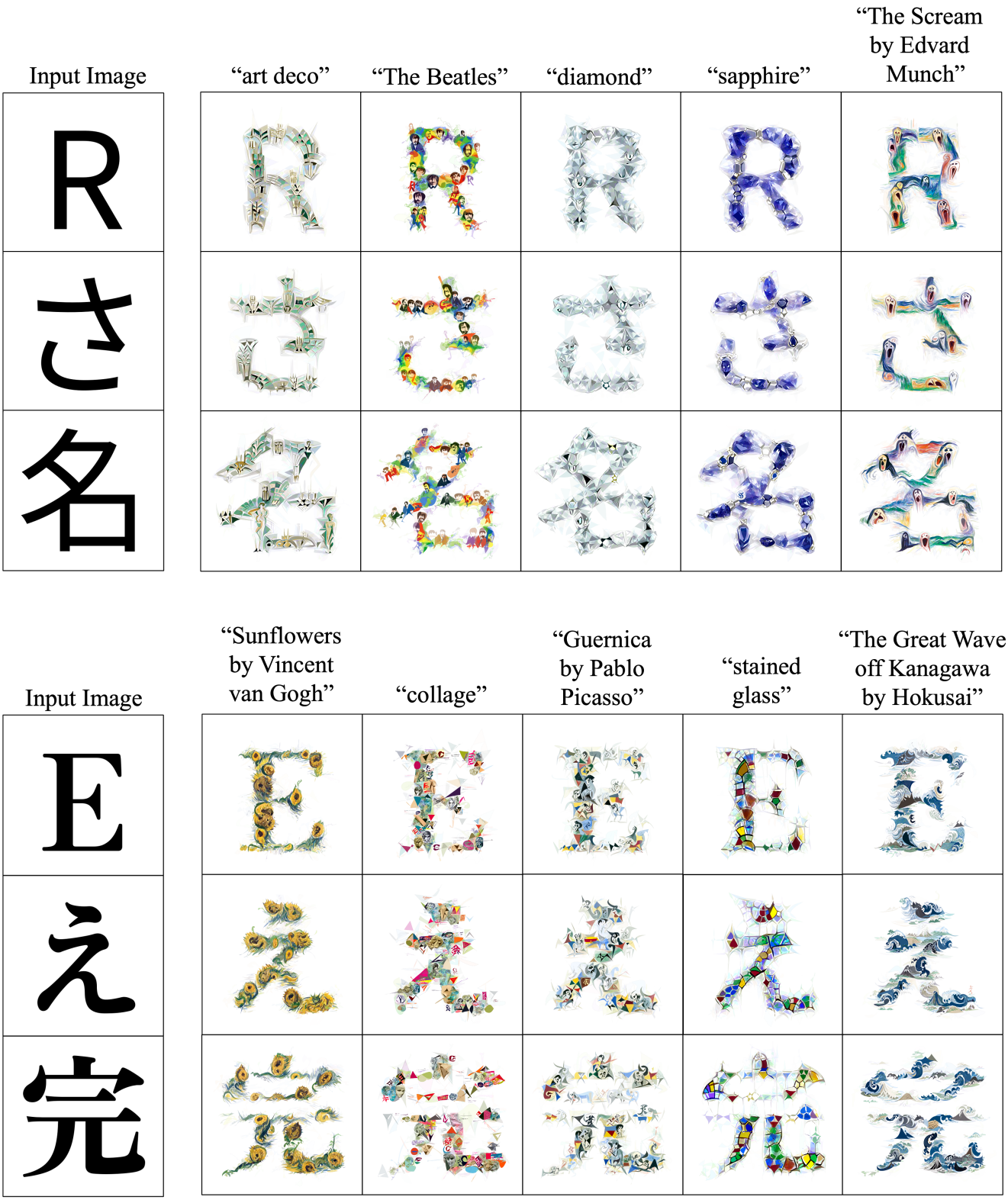

FIGURE 18. Additional results of our CLIPFontDraw.

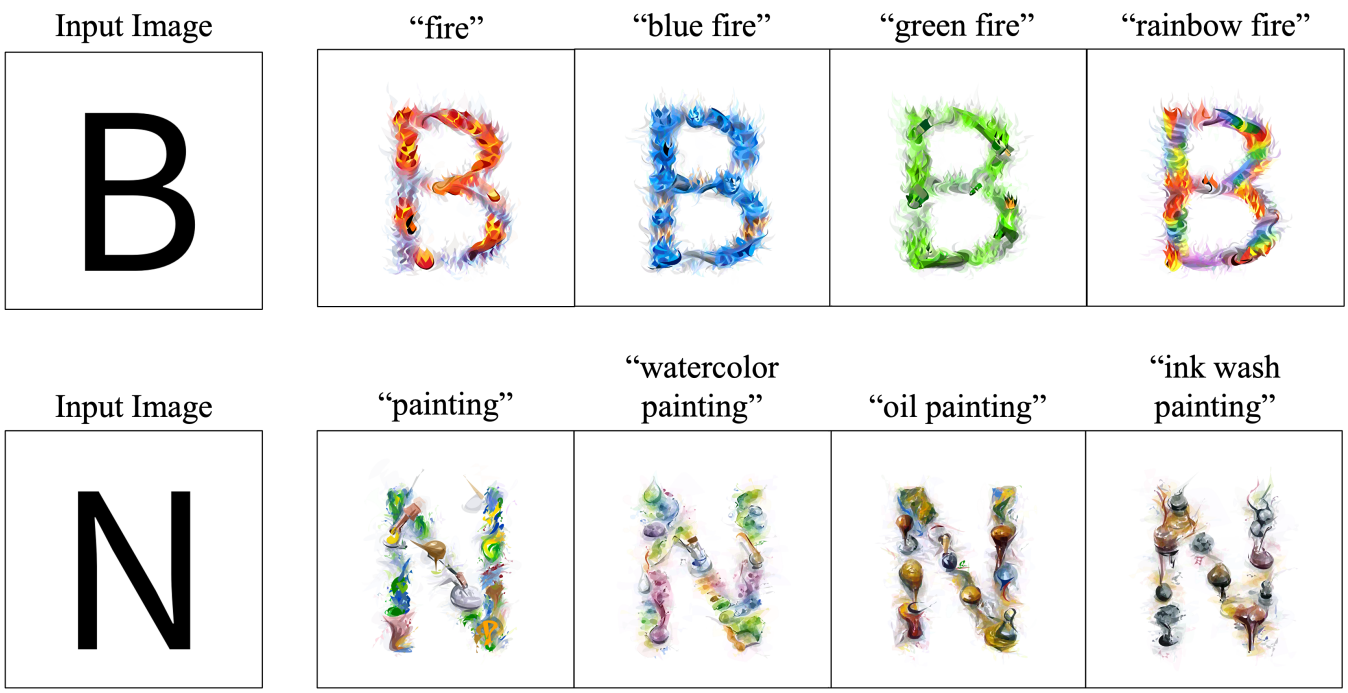

FIGURE 19. Additional results when changing adjectives.