Vector Logo Image Synthesis Using Differentiable Renderer

Ryuta Yamakura[1] and Keiji Yanai[1]

The University of Electro-Communications, Tokyo, JAPAN
{yamakura-r, yanai}@mm.inf.uec.ac.jp

Abstract. Vector images are widely used in the design field, particularly for logos and icons, due to their scalable properties. Consequently, the flexible and high-quality creation of such images is expected to support creative activities. In this study, we leverage a recently proposed differentiable renderer and the strong raster image generation capabilities of Stable Diffusion to generate vector-format logo images. This is achieved through optimizing vector parameters based on losses calculated from text prompts and shape images. Additionally, we address the self-intersection issue, a common challenge in vector image generation through optimization methods, by introducing a new technique called Radiation Loss. This approach explicitly monitors control points to enhance the quality of the output. While this method successfully generates logo images that maintain the input text and shape, challenges remain, including the persistence of unnecessary paths and difficulty in controlling the output entirely by text prompts. The experimental results showed the effectiveness of the proposed methods.

Keywords: vector graphics · differentiable renderer · logo image

1 Introduction

Vector images, which represent visuals through parametric mathematical primitives, are widely used in design fields such as fonts and icons due to their inherent advantages. These advantages include the ability to scale without losing image quality and typically smaller data sizes compared to equivalent raster images. Vector graphics are often employed for various visual elements, with logo creation being especially challenging. This is because logos need to embody artistic expression while also aligning with a specific visual identity.

Moreover, creating vector images is a time-consuming and labor-intensive process that requires not only proficiency in vector graphics but also the use of specialized tools. The ability to intuitively and flexibly generate vector images using text and image inputs as guides could significantly support artistic activities, streamlining the design process and minimizing unnecessary iterations.

This paper focuses on the generation of vector-format logo images. It explores methods for producing a diverse range of vector images that incorporate both visual identity and artistic design by utilizing shape-indicating images and text prompts to define the content.

Currently, vector graphics generation using generative models is primarily achieved through two methods: language-based approaches, which rely on models trained with explicit supervision of vector graphics command sequences, and image-based approaches, which combine pre-trained Text-to-Image models with image vectorization. The former approach is often limited to specific domains, such as logos and fonts, due to the challenges of collecting large, high-quality vector graphics datasets and the fact that vector graphics do not always have a unique representation.

In contrast, the latter approach benefits from recent advances in image generation models, as it can draw upon the knowledge of models trained on large datasets of raster images. Moreover, with the advent of differentiable renderers [6], raster images can now be vectorized through a straightforward optimization process using loss functions, as illustrated in Figure 1, without the need for complex vector transformation steps.

This paper adopts the latter approach, leveraging the capabilities of Text-to-Image models to generate diverse and stable vector graphics. Our contributions in this paper are summarized as follows:

- We propose a method for generating vector-format logo images by combining a differentiable renderer with the raster image generation capabilities of Stable Diffusion.
- We introduce Radiation Loss, a novel loss function that addresses the self-intersection problem in vector image generation, improving the quality and structure of the output.
- We demonstrate the successful generation of logo images that preserve both input text and shapes through extensive experiments using text prompts and shape images.

2 Related Work

2.1 Differentiable Renderer

Traditionally, the rendering process for vector graphics has been unidirectional, and vectorization of raster images required special methods [3, 15] that involve tracing edges. However, vectorization by these methods is unrelated to the original vector metrics and the vector graphics generated have a dramatically different structure, making it impossible to apply raster-based algorithms to vector graphics. Li et al. [6] addressed the problems of such vectorization methods by using two methods for pixel prefiltering based on the differentiable 3D renderer [7]: analytical prefiltering and multisampling antialiasing. They proposed a renderer that can automatically compute gradients for vector parameters.

2.2 Image Generation with Differentiable Renderer

With the advent of differentiable renderers, several methods were proposed to optimize vector parameters by directly applying raster-based loss functions and

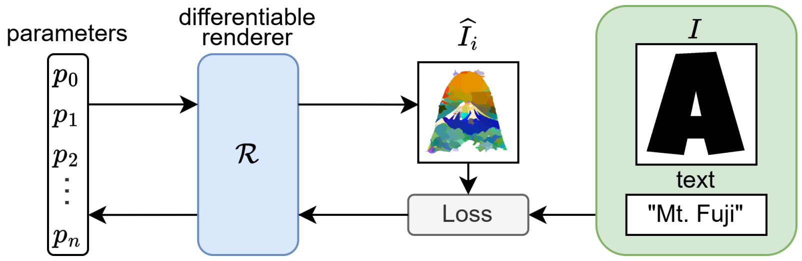

Fig. 1: Outline

machine learning methods to vector content, bypassing the limitations of the need to collect data for high-quality, large-scale vector content.

CLIPDraw [2] uses a large model of learned image-text relationships published by OpenAI, CLIP [10], to compute the similarity between an input text and an output image. The method optimizes the parameters of Bezier curves via a differentiable renderer, and was the basis for similar frameworks that followed.

CLIPDraw uses CLIP's image and text encoders to optimize the input text and output image similarity (CLIP Loss) for each iteration as a loss function to obtain a vector image output that matches the text. However, CLIPDraw uses Bezier curves as lines because the main purpose of CLIPDraw is to draw with vector paths, and the poor representation remains as an issue.

StyleCLIPDraw [14] used the VGG16 model to condition images with auxiliary style loss, and extended it to transfer styles to images generated by CLIPDraw.

In addition, VectorFusion [5] uses a trained text-to-image (T2I) model, Stable Diffusion [11]. Score Distillation Sampling (SDS) [9], a method that distills the model so that its output is transformed into an arbitrary parameter space, such as 3D or vector parameters, tailored for vector graphics. By using it as a loss function, we were able to generate more consistent vector graphics. The SDS Loss used in VectorFusion is also used in this study. Compared to CLIPDraw, VectorFusion has greatly improved drawing capability by using a robust output space of diffusion models. However, it does not have the ability to maintain the shape of the input image because it only receives text as input.

3 Method

3.1 Outline

A schematic diagram of the method is shown in Figure 1. In this method, a set of Bezier curve parameters (color, opacity, and control point coordinates), which is generated from the input image I, is rendered by using a differentiable renderer $\mathcal{R}$ with gradient generation enabled and $\hat{I}_i$ is obtained in each cycle, i. After enhancement using cropping and perspective transformation, the result is input into a stable diffusion model.

In this process, the Tone Loss, Radiation Loss, and SDS Loss are calculated and used to optimize the Bezier curve parameters through the gradient descent method.

3.2 Representation format and initialization

The primitives that make up the vector image are restricted to closed cubic Bezier curves, based on the concept of CLIPDraw [2]. Bezier curves are defined by a set of control points, and by connecting multiple Bezier curves, a wide variety of shapes can be approximated. This allows for a simple implementation and evaluation without losing expressive power.

In this method, a vector graphic $\hat{I} = \{P_0, ..., P_{N-1}\}$ is formed using N closed Bezier curve paths. Each path consists of M cubic Bezier curve segments, and the four control points p_m that make up each segment define $P_n = \{p_{0,0}, p_{0,1}, p_{0,2}, ..., p_{M-1,0}, p_{M-1,1}, p_{M-1,2}\}$. Note that a path always passes through the starting control point of a segment, and the ending control point of a segment is shared with the starting control point of the next segment to ensure connectivity. Furthermore, each P_n possesses a single color, and vector generation is performed by individually optimizing the control point coordinates and colors.

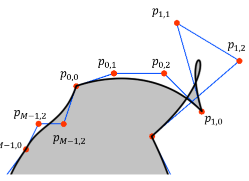

Fig. 2: Relationship of path and control points

The vector path is initialized by arranging all control points in a circular layout, following the idea of LIVE [8]. This method is expected to prevent path self-intersection issues in advance.

Additionally, to improve the convergence speed of the Tone Loss, the center coordinates of the path are pre-set within the range of the input image I. An example of this initialization is shown in Figure 3.

3.3 Self crossing problem

LIVE [8] points out that some vector paths self-intersect as a result of the path initialization and optimization process, leading to the generation of detrimental artifacts and improper topology. As shown in the upper left corner of Figure 4, paths with self-intersecting problems may generate artifacts when scaling beyond the default rendering size and additional paths may be generated to cover the artifacts. To address this problem, the study assumes that all vector primitives are cubic Bezier curves, and if the control points of a path are A, B, C, D in

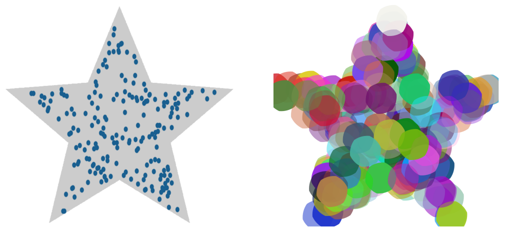

Fig. 3: Example of initialization. The center position of the Bezier curve is randomly obtained according to the input image (left figure) and initialized to a circular shape (right figure).

that order, the angle between $\overrightarrow{AB}$ and $\overrightarrow{CD}$ is The Xing Loss is expressed by the following equation:

$$D_1 = \mathbb{I}\left(\overrightarrow{AB} \times \overrightarrow{CD}\right), D_2 = \frac{\overrightarrow{AB} \cdot \overrightarrow{CD}}{\left|\left|\overrightarrow{AB}\right|\right|\left|\left|\overrightarrow{CD}\right|\right|} \tag{1}$$

$$\mathcal{L}_{xing} = D_1(\mathrm{ReLU}(-D_2)) + (1 - D_1)(\mathrm{ReLU}(D_2)), \tag{2}$$

where $\times$ is the outer product, $\cdot$ is the inner product, and $\mathbb{I}(\cdot)$ is the sign function.

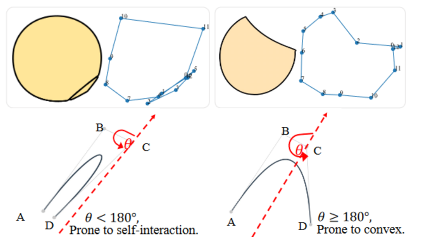

Fig. 4: Self crossing problem. (cited from [8])

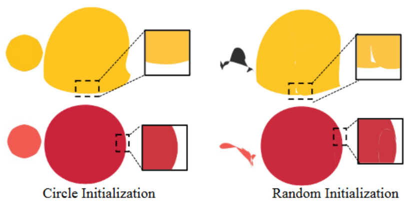

Fig. 5: Example where the path intersects between segments due to the initialization method. (cited from [8])

3.4 Loss Function

Figure 6 shows the derivation flow of the loss functions. Since logo images possess the appearance of a specific shape as a unique identity that preserves artistic graphics and identity, SDS Loss and Tone Preserving Loss are introduced to control the content by text and shape, respectively, and to restrict the shape

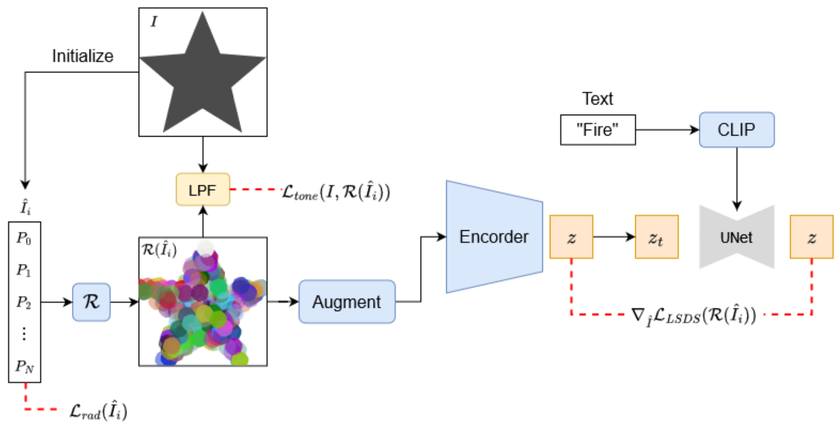
Initialize
I
Text
"Fire"
CLIP
LPF
$\mathcal{L}_{tone}(I, \mathcal{R}(\hat{I}_i))$
$\hat{I}_i$
P_0
P_1
P_2
P_N
$\mathcal{R}$
$\mathcal{R}(\hat{I}_i)$
Augment
Encorder
z
z_t
UNet
z
$\nabla_{\hat{I}} \mathcal{L}_{LSDS}(\mathcal{R}(\hat{I}_i))$
$\mathcal{L}_{rad}(\hat{I}_i)$

Fig. 6: Loss Function

to the region of the image where the input is located. In addition, Radiation Loss is introduced to solve the self-intersection problem, in which control points are optimized into uninterpretable shapes during the optimization process, a problem inherent in vector image generation using optimization methods.

Tone Preserving Loss Tone Preserving Loss is the loss proposed in Word-As-Image [4], which applies a low-pass filter to the rendered vector image and the image of the shape being referenced and takes the Euclidean distance between the images. This loss preserves the local tones of the image and limits the adjusted image from deviating too much from the input image.

$$\mathcal{L}_{tone} = ||LPF(\mathcal{R}(I)) - LPF(\mathcal{R}(\hat{I}_i))||_2^2 \tag{3}$$

where $\mathcal{R}(\cdot)$ is the renderer and P and $\hat{P}$ are the set of parameters that make up the vector image.

This method also introduces $\mathcal{L}_{tone}$ to restrict the shape to that of the input image. However, since Word-As-Image is a method intended to manipulate monotone font shapes, applying Equation (3) directly or applying a gray scale transformation to a rendered image will optimize the color value to be close to the mask value (usually 0), resulting in blackening the corresponding parts of the generated image. and the corresponding areas of the generated image will be blackened.

Therefore, Equation (5) applied to Tone Loss is used to create an image showing the region of the path from the difference with the background using an image rendered with a random noise background image, bg, according to the following equation:

$$\mathcal{R}(\hat{I}_i)_{binary} = \text{Sigmoid}\left(\alpha(\mathcal{R}(\hat{I}_i) - bg)^2)\right), \tag{4}$$

where α is the scaling factor and $\alpha = 10^6$.

$$\mathcal{L}_{tone} = ||LPF(\mathcal{R}(I)) - LPF(\mathcal{R}(\hat{I}_i)_{binary})||_2^2 \tag{5}$$

Score Distillation Sampling Loss Score Distillation Sampling (SDS) Loss is the loss function proposed in DreamFusion [9] using the sampling results when the Jacobian term in the diffusion model is omitted. More intuitively, it can be said to be a loss that optimizes the parameter θ to fit the conditional text prompt by minimizing.

At each iteration $t \in 1, 2, ..., T$, a randomly rendered image x is generated and then noise is added to this image, accompanied by the terms $\epsilon \sim \mathcal{N}(0, I)$, α_t, σ_t that control the noise schedule, $x_t = \ alpha_t x + \sigma_t \epsilon$ is formed. The noisy image is then passed to the Imagen [13] pre-trained UNet [12] model, which outputs a prediction of the noise ϵ. The SDS Loss is defined by the following equation:

$$\nabla_\theta \mathcal{L}_{SDS} = \mathbb{E}_{t,\epsilon}\left[w(t)\big(\hat{\epsilon}_\phi(x_t, t, y) - \epsilon\big)\frac{\partial x}{\partial \theta}\right], \tag{6}$$

where $\hat{\epsilon}_\phi$ is the UNet-based denoising network, y is the conditional text prompt, θ is the NeRF parameter, and $w(t)$ is a constant multiplier that depends on α_t.

While SDS Loss was utilized for the 3D object generation task in the proposed DreamFusion, VectorFusion [5] utilized SDS Loss for the vector graphics generation task. VectorFusion is defined similarly to DreamFusion with the raster images generated by Stable Diffusion as vectorized vector images or randomly initialized vector images as initial values, defined by the following formula:

$$\nabla_{\hat{I}} \mathcal{L}_{LSDS} = \mathbb{E}_{t,\epsilon}\left[w(t)\big(\hat{\epsilon}_\phi(\alpha_t z_t + \sigma_t \epsilon, y) - \epsilon\big)\frac{\partial z}{\partial z_{aug}}\frac{\partial x_{aug}}{\partial \theta}\right], \tag{7}$$

where x_{aug} is the enhancement by perspective transformation and cropping as shown in CLIPDraw [2], z is the encoding applying the stable diffusion pre-trained encoder $\mathcal{E}$, $z = \mathcal{E}(x_{aug})$. Similarly in this paper, SDS Loss is used to effectively utilize the T2I feature of Stable Diffusion as well as VectorFusion for the vector graphics generation task.

Radiation Loss Although the Xing Loss (Equation (2)), which solves the self-intersection problem, fully prevents self-intersections for single cubic Bezier curves, it is difficult to prevent segment intersections for actual Bezier curves with connected segments, and there is room for improvement in this respect. In this regard, LIVE prevents segment intersections by initializing the paths circularly. However, this method, which does not allow area-based initialization based on the target image and has a large variation of control points during the optimization process, is likely to cause segment intersections even if the same initialization is used. Figure 5 shows an example of segment intersections due to the path initialization method.

Therefore, we propose Radiation Loss, which is an extension of Xing Loss, taking into account the positional relationship with the segments before and after.

The simplest condition necessary for the entire connected path to not intersect is that all control points are in order toward one of the rotation directions. That is, with c as the median of the starting control point for each segment, for all control points comprising the path P_n, $\angle p_n c p_{n+1} < \angle p_n c p_{n+2}$ must be fulfilled. Figure 7 shows an overview of Radiation Loss. Radiation Loss is therefore defined as in the following equation:

$$\mathcal{L}_{rad} = \sum_n \mathrm{ReLU}(\angle p_n c p_{n+1} - \angle p_n c p_{n+2}) \tag{8}$$

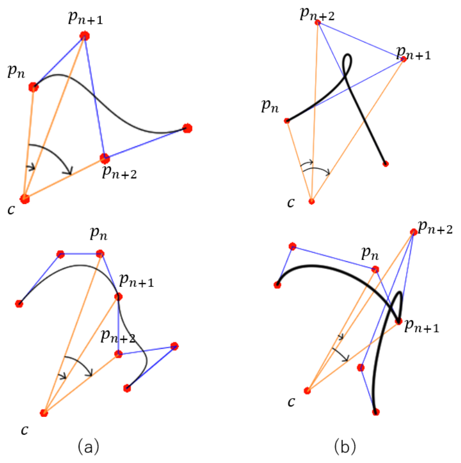

Fig. 7: Overview of Radiation Loss. (a) Example of sequential placement with respect to the direction of rotation, (b) Example of crossed paths as a result of placement against the direction of rotation.

Total Loss The above three losses are weighted and added together to define Total Loss.

$$\mathcal{L}_{total} = \lambda_{tone}\mathcal{L}_{tone} + \lambda_{rad}\mathcal{L}_{rad} + \lambda_{LSDS}\mathcal{L}_{LSDS} \tag{9}$$

Note that λ_{tone}, λ_{rad}, and λ_{LDSD} are weights to adjust each loss.

3.5 Path Elimination

Since SDS Loss is calculated using rasterized images in this method, path reduction and transparency occur so that Loss becomes smaller. These path ambiguities are not a problem for the raster image after rendering, but for the vector image, they have a negative impact on the data size and rendering process. Therefore, a more concise vector image is generated by removing the paths considering their impact on the rendered image during the optimization process. With a threshold value of τ, P_n is removed when the conditions represented in Equation (11) are satisfied.

$$\hat{I}_{i,n} = \mathcal{R}(\{P_n, ..., p_N\}) - \mathcal{R}(\{P_{n+1}, ..., p_N\}) \tag{10}$$

$$\frac{\sum_{x,y} \text{alpha}(\hat{I}_{i,n})}{w \times h} < \tau, \tag{11}$$

where $\mathcal{R}$ is the renderer, alpha$(\hat{I}_{i,n})$ is the image opacity, and w and h are the output image sizes. The expression 10 represents how much P_n is visible in the raster image after rendering, and in the experiment $\tau = 5.0 \times 10^{-4}$.

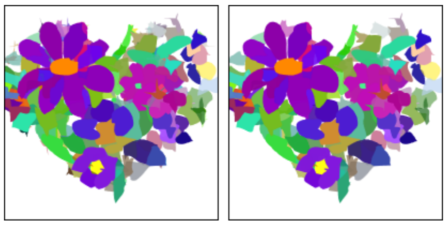

Fig. 8: Example before path deletion (left) and with path deletion applied (right). In this example, the number of Bezier curves has decreased from 200 to 134.

4 Experiments

4.1 Experimental Settings

The size of the input/output image is 600×600. The pre-diffusion model enhancement crops $\mathcal{R}(\hat{I}_i)$ to 512×512. By default, the number of Bezier curves is 200, the number of segments is 6, the number of parameter updates i is 1000, and the loss weights are $\lambda_{tone} = 200$, $\lambda_{rad} = 1$, $\lambda_{LSDS} = 1$, and for Tone loss the kernel size of the low-pass filter used was set to 101 and σ to 30. These values were set empirically. Also, the path was removed in case of $i = 800$. Prompt engineering was used for the input with reference to VectorFusion [5] and "a logo of {concept}. minimal flat 2d vector. lineal color. trending on artstation."

4.2 Results

The experimental results of the proposed method are shown in Figure 9. It can be seen that the structure shown in the text is generated in the shape of the input image. For example, in the case of the input "Mt. Fuji", a structure like

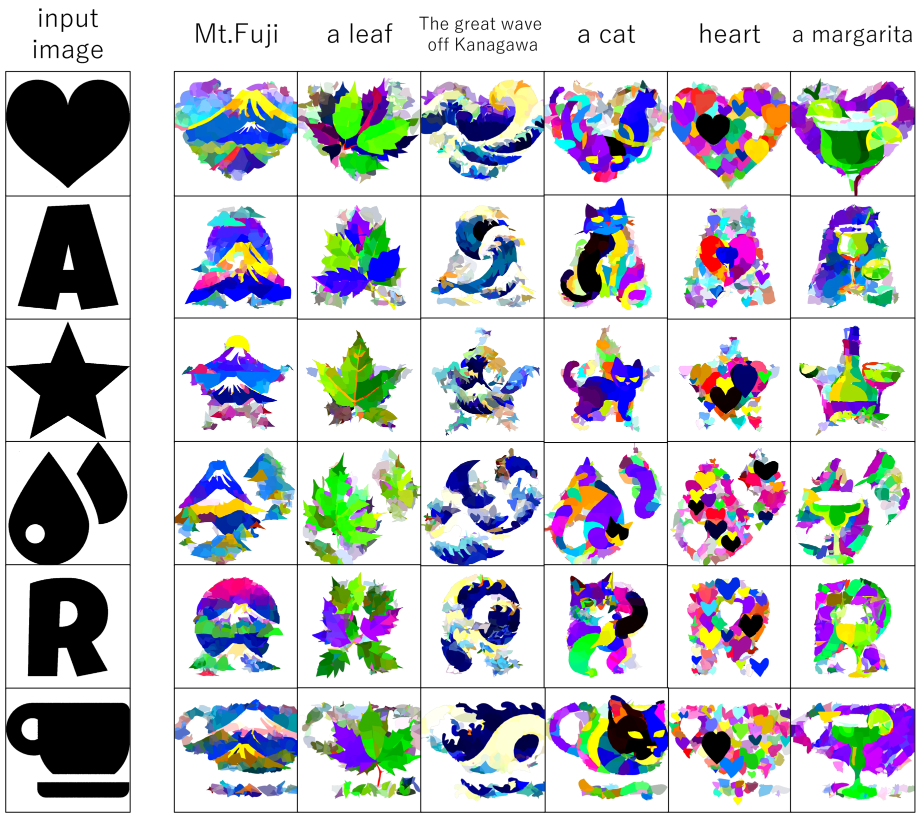

Fig. 9: Generated samples.

Mt. Fuji appears as the largest structure, while a path is generated that does not seem to make much sense and fills the input image area. In addition, as in the "rabbit" example, the color scheme is far from what is intuitively imagined from the input text.

4.3 Ablation Studies

Inference of Tone Loss The results of applying different weights to the Tone Loss are shown in Figure 10. It can be seen that the larger the weight, the closer the shape of the output image is to the input image. Therefore, it is considered possible to control the shape of the output image by adjusting this loss.

Inference of Radiation Loss The results of applying different weights to Radiation Loss are shown in Figure 11. The paths with Radiation Loss are completely convex, even when the weights are small. However, even when Radiation

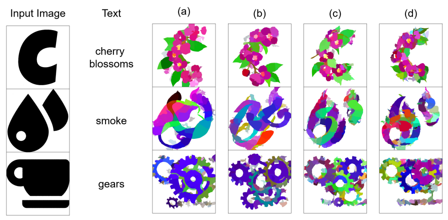

Fig. 10: An example of changing the Tone Loss weight. λ_{tone}. (a)$\lambda_{tone} = 1.0$, (b)$\lambda_{tone} = 1.0 \times 10$, (c)$\lambda_{tone} = 1.0 \times 10^2$, (d)$\lambda_{tone} = 3.0 \times 10^2$

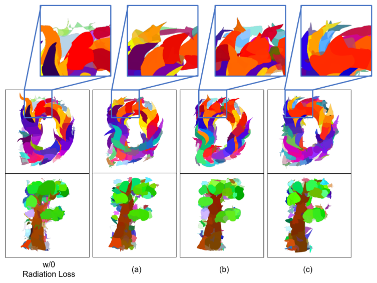

Fig. 11: Example of varying the Radiation Loss weight λ_{rad}. (a)$\lambda_{rad} = 1.0$, (b)$\lambda_{rad} = 1.0 \times 10$, (c)$\lambda_{rad} = 1.0 \times 10^2$

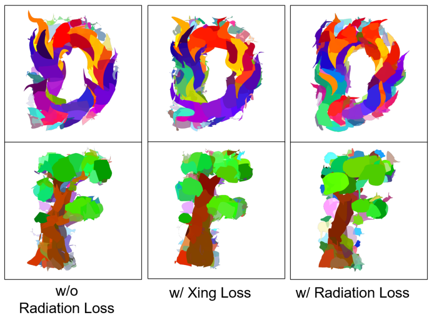

Fig. 12: Compare of Xing Loss and Radiation Loss

Loss is added, spine-like artifacts are still generated due to the increase in the size of the two points other than the starting control point.

A comparison of Xing Loss and Radiation Loss is also shown in Figure 12, showing that path self-intersections are reduced in Radiation Loss compared to the Xing Loss case, with each path forming a larger structure.

4.4 Other experiments

In order to examine the effect of manipulating text prompts on output results, experiments were conducted by adding colors to adjectives and by changing the prefix and suffix of the prompts. Figure 13 shows the results of generating text prompts as "{color} {concept}". It can be seen that the output examples with colored input reflect the color, while for achromatic colors such as "black" and "white", the color is either ignored or only partially applied.

Figure 14 shows an example of output when the number of parameter updates is changed. It can be seen that at $i = 1000$ most of the shape is formed, and at $i = 1500$ an almost completely stable shape is output. Therefore, it is advisable to choose an appropriate value between $1000 < i < 1500$.

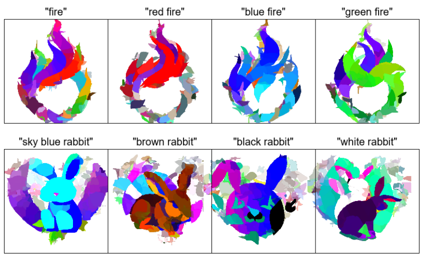

Fig. 13: Example output when color is added to adjectives

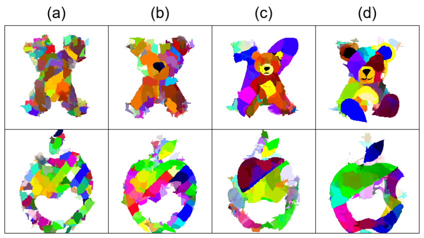

Fig. 14: Example output when i number of parameter updates is changed. (a)$i = 500$, (b)$i = 1000$, (c)$i = 1500$, (d)$i = 2000$,

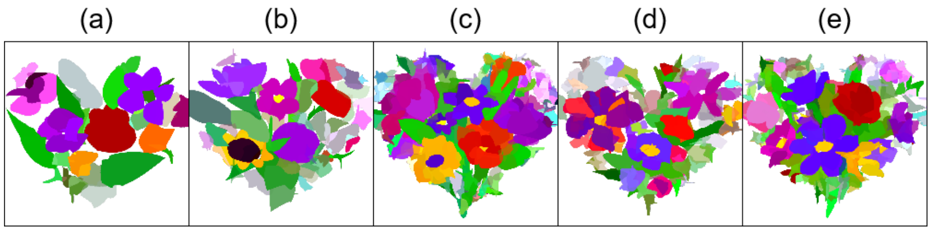

Fig. 15: Example output when the number of paths N is varied. The number of paths changes as follows by applying path deletion. (a) 32 to 32, (b) 64 to 52, (c) 128 to 116, (d) 256 to 141, and (e) 512 to 166.

An example of the generation when the number of paths N is changed is shown in Figure 15. It can be seen that by deleting paths, unnecessary paths are deleted when an excessive number of paths are specified.

5 Discussions

5.1 About the Output Results

Observing the sample of output results (Figure 9), we can see that in some parts, structures like those shown in the text appear, while in other parts, paths of little significance appear to satisfy the shape. This decrease is thought to be caused by the Tone Loss limiting the divergence between the sample results obtained with SDS Loss and the shape image. In addition, the color scheme of some of the samples deviates from the hue imagined from the text, and it is thought that the shape constraint imposed by Tone Loss distorts the shapes that could be generated. Since it is impossible to change the shape of the sampling result by SDS Loss with the shape constraint by Tone Loss, it is thought that these effects are similar to cropping to the shape of the input image. Therefore, it is necessary to extend the sampling results to maintain the shape of the input image to some extent by using the input image for conditioning the diffusion model.

5.2 Influence of Radiation Loss

As shown in Figure 11, Radiation Loss effectively suppressed the self-intersection problem and contributed to stabilizing the path shape. However, it remains inadequate for generating perfectly smooth paths, and artifacts often emerge where certain segments are elongated, resembling spines. This issue is likely caused by the optimization process attempting to minimize the effect of unnecessary segments on the drawing area within the constraints imposed by Radiation Loss.

Furthermore, because Radiation Loss incorporates information about the starting control point of each segment, it introduces unnecessary restrictions on its positioning, leading to impossible path shapes. Since this method employs SDS Loss to capture content based on the raster image post-rendering, the use of multiple Bézier curves to handle these impossible shapes results in suboptimal vector paths. To address this, an appropriate reference position corresponding to the center point c in Radiation Loss must be established.

6 Conclusions

In this paper, we propose a method for generating shape-preserving vector graphics by integrating the capabilities of a differentiable renderer [6] with SDS Loss [9] and Tone Loss [4]. Additionally, we introduce Radiation Loss, an extension of Xing Loss [8], to address the self-intersection problem in vector graphics generation. Our experimental results demonstrate that the proposed method can effectively incorporate textual content into vector graphics while preserving the shape of the input image. However, the output is influenced by the input conditions, indicating room for further improvement.

For future work, we consider employing image-to-image conditioning in Stable Diffusion, rather than directly controlling the shape, to produce outputs that more closely match the input image. Furthermore, we plan to refine Radiation Loss by adopting central-axis transformation [1] instead of the current central coordinate c, which will help to alleviate unnecessary restrictions on the starting control point.

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

References

1. Blum, H.: A Transformation for Extracting New Descriptors of Shape, pp. 362–380. MIT Press (1967)
2. Frans, K., Soros, L., Witkowski, O.: CLIPDraw: Exploring text-to-drawing synthesis through language-image encoders. In: Advances in Neural Information Processing Systems. vol. 35, pp. 5207–5218 (2022)
3. Hertzmann, A.: Painterly rendering with curved brush strokes of multiple sizes. Procs. of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98 pp. 453–460 (1998)
4. Iluz, S., Vinker, Y., Hertz, A., Berio, D., Cohen-Or, D., Shamir, A.: Word-As-Image for Semantic Typography. ACM Transactions on Graphics **42**(4), 1–11 (2023)
5. Jain, A., Xie, A., Abbeel, P.: VectorFusion: Text-to-SVG by Abstracting Pixel-Based Diffusion Models. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1911–1920 (2023)
6. Li, T., Lukáč, M., M, G., Ragan-Kelley, J.: Differentiable vector graphics rasterization for editing and learning. ACM Trans. Graph. (Proc. SIGGRAPH Asia) **39**(6), 193:1–193:15 (2020)
7. Li, T.M., Aittala, M., Durand, F., Lehtinen, J.: Differentiable monte carlo ray tracing through edge sampling. ACM Trans. Graph. (Proc. SIGGRAPH Asia) **37**(6), 222:1–222:11 (2018)
8. Ma, X., Zhou, Y., Xu, X., Sun, B., Filev, V., Orlov, N., Fu, Y., Shi, H.: Towards layer-wise image vectorization. In: Proc. of CVF/IEEE International Conference on Comput Vision and Pattern Recognition. pp. 16314–16323 (2022)
9. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: DreamFusion: Text-to-3D using 2D diffusion. In: The Eleventh International Conference on Learning Representations (2022)
10. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: Proc. of the International Conference on Machine Learning. vol. 139, pp. 8748–8763 (2021)
11. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-Resolution Image Synthesis with Latent Diffusion Models. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10674–10685 (2022)
12. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference. pp. 234–241 (2015)
13. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S.K.S., Gontijo-Lopes, R., Ayan, B.K., Salimans, T., et al.: Photorealistic text-to-image diffusion models with deep language understanding. In: Advances in Neural Information Processing Systems (2022)
14. Schaldenbrand, P., Liu, Z., Oh, J.: StyleCLIPDraw: Coupling Content and Style in Text-to-Drawing Translation. Procs. of the Thirty-First International Joint Conference on Artificial Intelligence pp. 4966–4972 (2022)
15. Selinger, P.: Potrace : a polygon-based tracing algorithm. `http://potrace.sourceforge.net/potrace.pdf` (2003)