2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

TIFVec: an Image Vectorization Approach Based on Texture Intensity Field

Yongjian Liu[1,2[0009-0005-8388-6736]], Jiaqi Liu[1,2[0009-0005-0566-1857]], Jiachen Li[1,2[0000-0002-0602-9360]], Yanchun Ma[3[0009-0001-4600-749X](✉)], Qing Xie[1,2[0000-0003-4530-588X]] and Anshu Hu[1,2[0009-0005-4568-579X]]

[1] Wuhan University of Technology, Wuhan, Hubei, China
[2] Engineering Research Center of Intelligent Service Technology for Digital Publishing, Ministry of Education
{liuyj,ljq_7,lijiachen,felixxq,anshuhu}@whut.edu.cn
[3] Wuhan Vocational College of Software and Engineering, Wuhan, Hubei, China
mayanchun@whvcse.edu.cn

Abstract. Image vectorization works to convert raster images into vector graphics, which is widely used in various fields. The current state-of-the-art approaches are learning-based models, which aims to establish the correlation between the raster image and a specific number of randomly distributed primitives through deep learning, such as Bézier curves. However, these methods have not payed attention to the influence of some important factors on the performance of vectorization, such as the number of primitives and the initial primitive positions. Therefore, the converted vector outputs usually suffer from various shortcomings such as excessively high number of primitives, unclear rendering of details, color mean errors, and prolonged primitives optimization time.

To address the aforementioned issues, we propose an image vectorization framework termed TIFVec, which takes the Bézier curves as primitives and discovers the interdependent mechanism among different factors. In the framework, we introduce the texture intensity field (TIF), which is able to guide the optimization of those factors above, in terms of the primitive initialization strategy and TIF-based objective function. Based on TIF, the connections among different factors can be constructed, and the performance of vectorization can be effectively improved.

The experimental results demonstrate that our method significantly outperforms the current state-of-the-art models across multiple datasets, in terms of the visual results, evaluation indicators, and primitives optimization time. Experiments demonstrate that our method achieves state-of-the-art among self-supervised models and unsupervised methods.

Keywords: Image vectorization, Texture Intensity Field, Vector Graphics.

1 Introduction

Image vectorization is the process of converting a raster image into a compact, easy to edit, distortion-free vector graphic, which has been studied extensively in vision, graphics, and other areas. Representative research on image vectorization can be categorized into heuristic approaches and learning-based approaches.

The heuristic approaches, such as mesh-based [1] and curve-based [2] methods, analyze the pixel points and fit the paths and shapes using vector primitives in a heuristic way. However, these methods may suffer from accuracy loss when dealing with certain image details or shapes. For instance, the primitive curves may fail to capture sharp edges or subtle variations in the image, while the grids may struggle to accurately represent complex shapes within the image.

The learning-based approaches propose to train a model that transforms the raster images into vector parameters, e.g, rendering images using a fixed number of randomly distributed Bézier curves [3], and hierarchical layering of Bézier curves for image rendering [4]. The core intuition of current approaches is to utilize a differentiable rasterizer as the bridge between the raster and vector domains, so the model can learn to optimize or modify the vector parameters by fitting the contents of raster images during the training process. Learning-based models have greatly improved the quality of vectorized images, but they still need to address the following challenges:

1. The number of the vector primitives should be minimized to accelerate the image rendering. Fewer primitives also benefits storage and transmission.
2. The distribution of the vector primitives should be optimized to describe the image details so as to minimize the image distortion.

However, the current approaches fail to discover the influence of those important factors on the performance of vectorization, such as the number of primitives, the initial primitive positions. They arbitrarily use a fixed number of primitives and randomly distribute them as initialization without considering the complexity of image textures, and both the areas of complex textures and the smooth areas are equally treated during the rendering process. These lead to an excessively high number of primitives, unclear rendering of fine texture details in complex areas, error in color mean, and prolonged primitives optimization time.

To address the aforementioned issues, we propose an image vectorization framework that relies on the texture intensity field(TIF), referred as TIFVec (Texture Intensity Field based Vectorization), which also takes the Bézier curves as primitives. The basic intuition is that the distribution of the primitives can be optimized according to the image textures. For example, the regions of complex textures should consume more primitives for the rendering of details, while smooth regions should consume fewer primitives. Based on this simple intuition, we introduce TIF to discover the variation of the image textures, which can further guide the number of employed primitives and their initial distribution. By constructing the connections among various factors that affect the performance of vectorization, the training objectives can be optimized for rendering different types of regions, so as to improve the overall vectorization quality. In details, our contributions can be summarized as follows:

1. By introducing the concept of texture intensity field, the texture complexity of an image can be estimated to guide the optimization of primitive distribution, based on which an effective and efficient model for image vectorization is proposed. We provide several solutions to construct the TIF, including finite-difference threshold filter, gray-level co-occurrence matrix filter, and segmentation of connected regions.
2. Based on the constructed TIF, 2 strategies are designed to initialize the number and position distribution of primitives, based on finite-difference threshold and region segmentation respectively. Both strategies can distribute a larger number of primitives in regions with complex textures and fewer primitives in regions with simpler textures.
3. We propose a novel TIF-based objective function to guide the model adjusting the rendering parameters according to different regions, based on which we can avoid the influence of the error in mean color and distortion, and achieve a significant increment in rendering efficiency.
4. We evaluate our TIFVec on 2 public datasets including the Fonts [5] and Emoji[1], and also 3 more sophisticated and challenging self-collected datasets including the animated images, artistic images, and realistic photos (will be released along with this work). Experimental results demonstrate that our TIFVec significantly outperform the state-of-art vectorization models in terms of the effectiveness and efficiency. With certain SSIM criteria, the appropriate number of primitives can be effectively controlled to guarantee the editability and compactness of the vectorized images.

2 Related works

2.1 Heuristic Methods

In the past decade, significant progress has been made using heuristic algorithms. Mesh-based methods divide the image into non-overlapping 2D patches and interpolate colors between them, with blocks in triangular [1,6,7], rectangular [8,9], or irregular [10,11] shapes. Curve-based methods typically use Bézier curves as geometric primitives with colors defined on both sides of the curves [12]. Xie et al. [2] proposed tracing curves in the Laplacian domain to create a hierarchical representation that accurately reconstructs vector art and natural images.

2.2 Learning-based Methods

In recent years, with the rise of deep learning, researchers have employed neural networks to tackle the vectorization problem. [13] proposed a transformer-based architecture that utilizes CNN to convert technical line drawings into vector parameters. Sketchformer [14] was proposed to recover raster images from sketches by a transformer-based network. [5] trained an image Variational Auto Encoder (VAE) and used a decoder to predict the vector parameters by the learned latent variables obtained from

[1] https://github.com/googlefonts/noto-emoji

the image. [15] introduced a hierarchical vectorization model to perform the interpolation and generation tasks.[3] proposed a differentiable rasterizer that allows for the manipulation and generation of vector parameters using raster-based objective functions and machine learning techniques. [4] proposed Layer-wise Image Vectorization, namely LIVE and used the differentiable rasterizer [3] to fit each vector curve. [16] proposed O&R, which is a fast Top-Down approach for raster to vector image conversion, optimizing and reducing vector parameters to aim at a low budget of shapes for vectorization.

3 Method

As mentioned before, the existing approaches initialize the primitives in an arbitrary manner, which results in many problems. We thus propose our TIFVec to provide a better solution. As shown in Fig. 1, we formulate TIFVec as 2 stages.

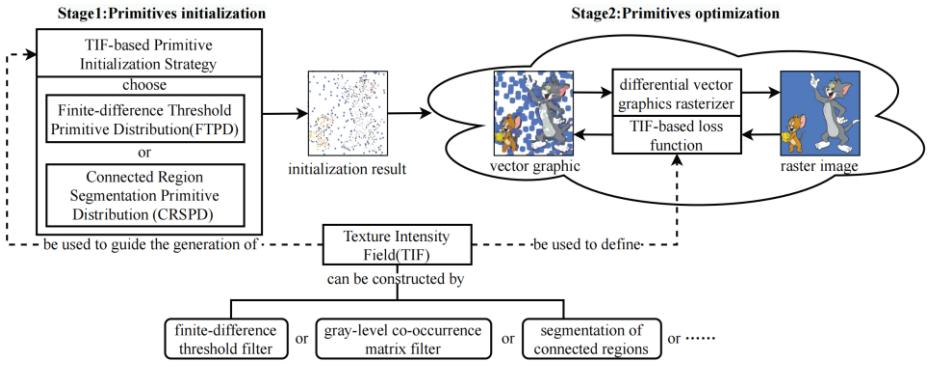

Fig. 1. Framework of the TIFVec.

Stage 1): Primitives initialization that distributing the primitives across a rectangle canvas of the same size as the raster image. We offer 2 primitive initialization strategies: one is referred to as FTPD, and the other is CRSPD, both of their implementations are guided by the TIF. Each one of the strategies can be chosen as needed to produce the initialization results.

At this stage, we actually distribute shapes that consist of 4 Bézier curve primitives connected end to end to form approximately small circles, with a radius set to 3 pixels. Previous work [4] has demonstrated that these small primitive-circles can effectively reduce the artifacts generated during the vectorization process. For these primitive-circles, the color is sampled from the corresponding positions in the raster image to fill the circles, so as to reduce the rendering errors caused by color randomness.

In the following discussion on primitives initialization, we use the term "primitives" to refer to "primitive-circles" for simplicity and clarity.

Stage 2): Primitives optimization that driven by a differential vector graphics rasterizer [3], during which the primitives are continuously adjusted by diffusion, deformation, movement, color transformation etc., to make the vectorized canvas approach

the raster image. At this stage, we introduce TIF to the loss function to enhance the model's focus on image details.

We provide 3 methods for constructing the TIF, but there are many more methods available for building TIF beyond these 3.

3.1 Texture Intensity Field

The textures can reflect the content complexity of a region in an image, which is crucial in guiding the primitives distribution, including their initial number, placement, and contribution weight on different region types. The principle is that regions of complex textures should consume more primitives for detailed rendering, while smooth regions should consume fewer primitives. We thus define the Texture Intensity Field (TIF) for the first time: TIF is a two-dimensional matrix of the same size as the raster image that each magnitude in the TIF should be able to reflect the complexity level of the textures of its corresponding neighborhood in the raster image. According to this definition, we provide 3 methods for constructing the TIF, as follows:

By finite-difference threshold filter: a) For each pixel $p_i, i \in \{1, \dots, w \times h\}$, where $w \times h$ represents the image size, we select an $n \times n$ neighborhood of p_i, where n=9 by default. b) Compute the finite-difference magnitude in eight directions for each pixel within the neighborhood and set a finite-difference threshold c_g. Empirically, we set c_g=50. c) Finally, calculate the count of pixels in the neighborhood whose finite-difference are greater than c_g in all eight directions, and the count is normalized as the texture intensity v_i.

By gray-level co-occurrence matrix filter: a) For each pixel $p_i, i \in \{1, \dots, w \times h\}$, select a neighborhood of size $n \times n$ to represent the surrounding region, where n=9 by default. b) Calculate the gray-level co-occurrence matrix (GLCM) of the neighborhood and use the entropy value as the texture intensity of p_i, denoted as v_i:

$$v_i = \sum_{j=1}^{n \times n} P_j^i log P_j^i \tag{1}$$

where P^i is the GLCM of the neighborhood surrounding p_i and j is the index of the matrix. c) Finally, v_i is normalized.

The GLCM [17] provides comprehensive information about an image's gray levels, including direction, adjacent spacing, and variation amplitude. High entropy values in GLCM indicate complex textures, vice versa.

By segmentation of connected regions: a) Utilize the Canny algorithm to detect the edges of the image, and then employ the breakpoints linking algorithm [18] to connect the non-closed edge regions to obtain all connected regions. b) Use a centroid to represent each of the connected regions. For the irregular regions whose centroid falls outside or the regions with large areas that a single point is insufficient to represent the entire region, we continuously perform segmentation operations on these regions. Specifically, taking the centroid point of each region as the origin, we conduct horizontal

and vertical cuts respectively, thereby dividing the region into four sub-connected regions. After each segmentation, we update the set of connected regions. The above segmentation and update operations are repeated until all regions meet the predefined conditions. c) Use a filter with size of 9×9 to statistically count the number of centroid points within each region.

Each intensity value in TIF should be normalized to [0, 1). Fig.2 exemplifies the heat-maps of the TIF constructed by 3 proposed methods. The red portion of the heat-map clearly indicates the more intricate texture present in the original image, particularly in areas such as the faces and hands of the cat and mouse. In contrast, the background heat-map is entirely blue, indicating a smooth area. It can be observed that there is hardly any significant difference between the heat-maps constructed by different methods. In fact, there are many other approaches to construct TIF, as long as it aligns with our definition of TIF. Each can be chosen to guide the generation of primitives initialization strategies and the construction of loss function. In the subsequent experiments described, we adopted TIF constructed by gray-level co-occurrence matrix filter.

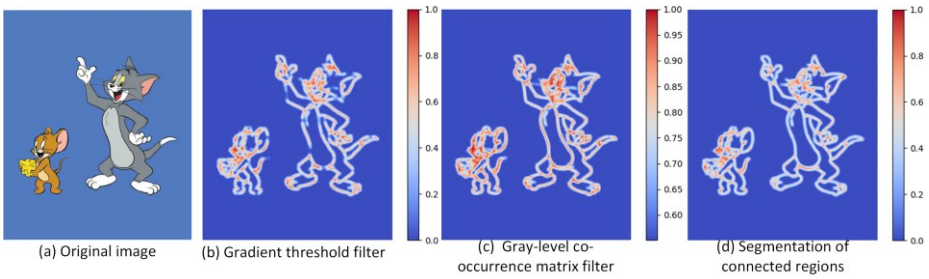

Fig. 2. Heat-maps of the TIF constructed by 3 methods. Regions that appear in red indicate higher magnitude of the texture complexity.

3.2 TIF-based Primitive Initialization Strategy

During the primitive initialization, TIF is used to guide the number and placement of the primitives. The principle is that the areas with high texture intensity are distributed densely packed primitives, and vice versa. Based on this, we propose 2 strategies to determine the distribution of the primitives:

Finite-difference Threshold Primitive Distribution (FTPD) strategy: a) Given a fixed number N of primitives and an arbitrary-sized raster image Im, and assume the textures in the Im is smooth. b) Segment the Im into K regions based on the TIF, and pixels with the same intensity magnitude are assigned to the same region. The probability distribution of primitives in region \$t\$ is defined as P_t:

$$P_t = \frac{v_t \, log \, n_t}{\sum_{i=1}^{K} vi \, log \, n_i} \tag{2}$$

where $t \in \{1, \ldots, K\}$, v_t is the texture intensity value in TIF, n_t is the number of pixels in region t that have the same value of v_t. The number of primitives distributed in region t is denoted as N_t, and it is equal to NP_t. Through this, we distribute more primitives in regions with higher texture intensity and larger area.

FTPD strategy targets the case of a fixed number of primitives. Since majority of current vectorization methods adopt a fixed number of primitives, this strategy enables us to show the performance advantages of our method compared with other methods under the condition of using the same number of primitives.

Connected Region Segmentation Primitive Distribution (CRSPD) strategy: This strategy is aimed at scenarios where a fixed number of primitives is not required. Intuitively, the more primitives there are, the better the rendering quality of the image. A fixed number of primitives may lead to insufficient image rendering and loss of texture details.

The CRSPD strategy can automatically determine the number and positions of the primitives. The idea of this strategy is consistent with the construction of TIF through segmentation of connected regions. It consists of 2 simple steps: 1)firstly segment the connected regions in the image using a traditional connected components analysis method [20]; 2) secondly select the centroid of each connected region as the initialization position for the primitives. The connected regions with higher texture intensity tend to be smaller and denser, resulting in a larger number of initialization positions for the primitives. Conversely, the connected regions with lower texture intensity tend to be larger and sparser, leading to a decrease in the number of initialization positions for the primitives. The number of

primitives in this strategy depends entirely on the number of connected regions in the image.

3.3 TIF-based Loss Function

In previous works [3,19], mean square error (MSE) loss $(Im - \overline{Im})^2$ are utilized to adjust the distinction between raster image $Im \in R^{w \times h \times 3}$, and rendered output $\overline{Im} \in R^{w \times h \times 3}$, where RGB channels are considered and $w \times h$ represents the image size.

MSE loss is simple and effective in optimization stage, however, it is calculated based on all available pixels, which leads the model paying excessive attention on the optimization of smooth regions that occupy the most area, while ignoring the details of the image. As shown in Fig.3(b), based on MSE loss, the color of the visual rendering results tends to be bias to the mean color, while the TIF-based loss function preserves the color and shape of the target.

To resolve this problem, we propose a novel TIF-based loss function, formulated as follows:

$$\ell_{TIF} = \frac{1}{3} \sum_{i=1}^{w \times h} v_i \sum_{c=1}^{3} (Im_{i,c} - \overline{Im_{i,c}})^2 \tag{3}$$

where i is the index of the pixel in the image Im, v_i is intensity magnitude in TIF, c is the index of the RGB channel. The TIF-based loss function emphasizes the differences in region with higher texture intensity while suppressing the differences in region with lower texture intensity. By this design, we can avoid the impact of the error in color mean during the primitives optimization and maintain the accuracy of texture reconstruction, as shown in Fig.3(c). Additionally, the TIF-based loss function can significantly reduce the primitives optimization time as well.

Fig. 3. Differences between MSE and TIF-based Loss.

4 Experiments

4.1 Datasets

Since the TIF defined reflects image texture complexity, we can define the texture complexity of each dataset as:

$$Complexity = \frac{1}{n}\sum_{t=1}^{n} L1_{norm}\,(TIF) \tag{4}$$

Where n is the number of images in dataset. We evaluate the effectiveness and superiority of our proposed TIFVec on 5 datasets from various aspects, which are detailed in Table.1 ordered by texture complexity.

Table 1. Description of the 5 datasets, Fonts is from [5] and Emoji is widely used in existing vectorization methods [4,16,19] separately.

Dataset	Quantity	Contents	public	Complexity
Fonts	134	characters	yes	0.179
Emoji	203	emojis	yes	0.257
Animations	93	animations	no	0.330
Art	52	arts	no	0.532
Realistic	35	photos	no	0.733

4.2 Implementation

Our TIFVec is implemented in PyTorch and all comparisons are performed on hardware with an NVIDIA Geforce RTX 3090 GPU and 12 Intel(R) Silver 4214R CPU. In the primitive initial phase, we set the circle radius to 3 pixels and distribute 12 control points evenly along the circumference. Additionally, we use 1000 primitives in FTPD strategy for all experiments, and set the number of iterations as 1000 in the primitive optimization stage.

4.3 Evaluation metric of vectorization.

We employ PSNR (Peak Signal to Noise Ratio) and SSIM (Structural Similarity) to quantify the similarity between the input raster images and output vector graphics. During the evaluation process, all vector outputs are rasterized into images with their original size by CairoSVG[2] for comparison.

4.4 Comparison Experiments

Fig. 4. Comparison with learning-based methods on Fonts dataset (low complexity dataset).

Visual Comparison: We compare our method with some representative learning-based vectorization methods, including SVG-VAE [5], DeepSVG [15], Im2Vec [19], LIVE [4], and O&R [16]. The previous work [21] has already demonstrated that SVG-VAE, DeepSVG, Im2Vec are unable to handle objects with complex content. For a fair comparison, we perform experiments exclusively on Fonts and Emoji datasets. We use the FTPD stratery and set the number of primitives as 16 that is same to LIVE and O&R. Fig.4 presents part of the visual results. It is worth noting that SVG-VAE and DeepSVG require vector supervision during the rendering process, while the Emoji dataset does not have corresponding vector ground truth. Therefore, SVG-VAE and DeepSVG cannot be tested on this dataset. It can be observed that LIVE, O&R and our method excellently accomplish the task of vectorizing simple images.

To demonstrate the performance of our TIFVec on complex datasets, we compared it with representative heuristic methods and learning-based vectorization methods on the Animations, Art and Realistic photos datasets. The methods in comparison include

[2] https://github.com/kozea/cairosvg

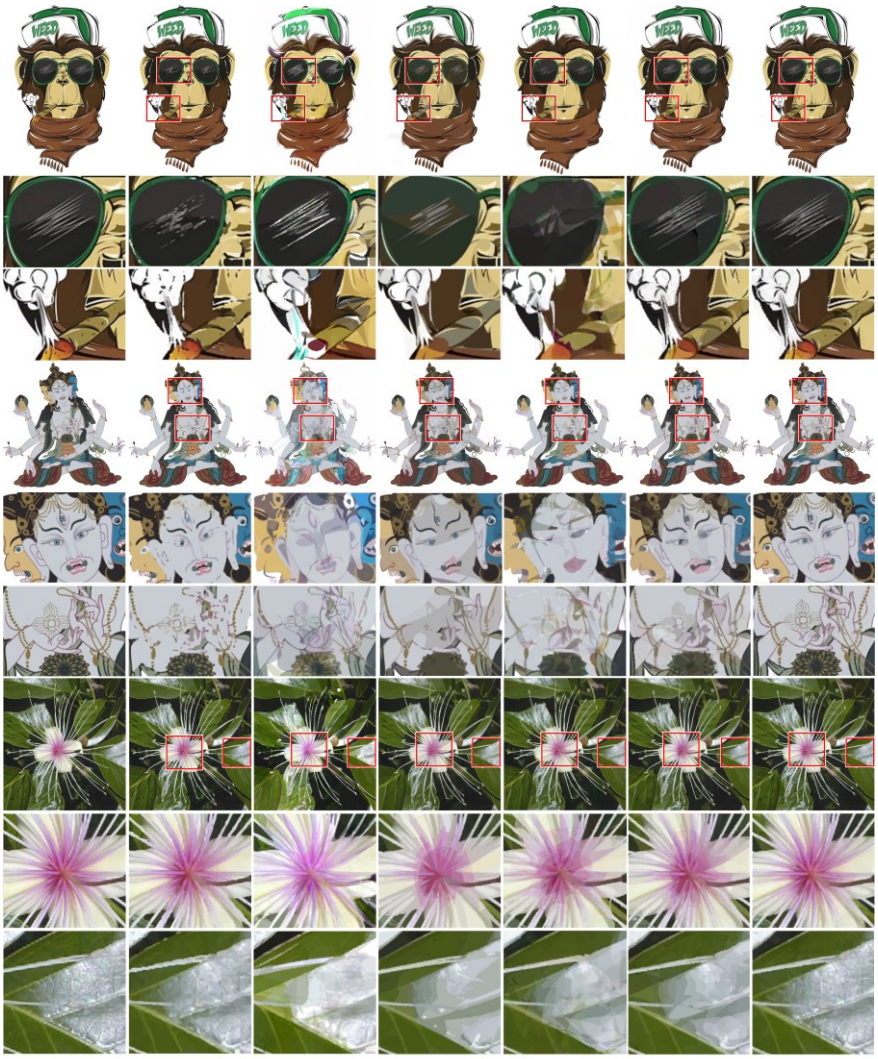

Fig. 5. Comparison with representative heuristic methods and learning-based methods on 3 datasets of high complexity. The first 3 rows consist of images from the Animations dataset, along with 2 enlarged image details. The middle 3 rows display images from the Art dataset, and the final 3 rows showcase images from the Realistic photos dataset.

Adobe (a commercial software), DiffVG [3] (primitive number=1000), LIVE [4] (primitive number=1000), for O&R [16], we ensure that the primitive number after optimization and reduction is 1000. For each image, we select two detail areas and magnify them. All results are illustrated in Fig.5.

We magnify 2 regions with complex textures from each image to showcase the details. Adobe experiences content loss, such as in light reflection on the green leaf and the eyes of the Buddhist statue. DiffVG encounters color bias issues in processing all images, O&R did not handle the gaps between primitives well, leading to a fragmented feeling in the image. while LIVE suffers from detail loss with complex images. By contrast, TIFVec with FTPD strategy performs better than LIVE overall, but still has some detail loss when processing Buddhist statue image due to primitive number limitations. However, using the CRSPD strategy in TIFVec preserves details well in each image, making it the most effective method.

Table 2. Average SSIM and PSNR comparison among learning-based methods and our TIFVec(FTPD) on Fonts, Eomji datasets.

Methods	Fonts		Emoji	
	PSNR	SSIM	PSNR	SSIM
SVG-VAE	22.25	0.433	×	×
DeepSVG	24.71	0.524	×	×
Im2Vec	27.86	0.808	30.51	0.942
LIVE	29.40	0.885	32.94	**0.969**
O&R	28.15	0.796	31.77	0.955
FTPD	**32.29**	**0.985**	**35.87**	0.668
	+2.89	+0.100	+2.93	-0.001

Quantitative Comparison: We evaluate the vectorization accuracy on the Fonts dataset and Emoji dataset by comparing the PSNR and SSIM metrics. For fair comparison, we calculate metrics for each image in the dataset. Table.2 presents the average results of objective metrics. The LIVE model slightly surpassed our TIFVec(FTPD) by 0.001 only in the SSIM while consuming 16 times the primitive optimization time of our method. Except this, our method achieves top-notch performance on both datasets. Regarding primitives optimization time, we will provide an analysis in subsequent sections.

Table 3. Average SSIM and PSNR comparison among representative heuristic methods, learning-based methods and our TIFVec(FTPD and CRSPD) on Animations, Art, Realistic photos datasets. number of primitives is set to 1000 in DiffVG, LIVE, O&R.

Methods	Animations		Art		Realistic photos	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Adobe	36.38	0.978	**34.98**	0.972	31.06	0.964
DiffVG	33.47	0.913	30.07	0.852	29.18	0.860
LIVE	36.47	0.991	32.45	0.964	30.72	0.910
O&R	36.34	0.967	31.95	0.928	30.55	0.899
FTPD	**36.81**	**0.992**	33.14	0.976	31.28	0.976
CRSPD	36.30	0.991	33.34	**0.982**	**31.70**	**0.985**

Quantitative Comparison for Animations, Art and Realistic photos datasets are summarized in Table.3. We calculate the average PSNR and SSIM for each dataset. Our TIFVec achieves the highest SSIM value across all datasets, while PSNR also received the highest score on all datasets with the exception of only the Art dataset. The exception is due to our vectorization method, which emphasizes capturing textures in images, fine details, and minimizing texture loss. This may introduce some redundant noise, resulting in a slight decrease in signal-to-noise ratio value. In contrast, the PSNR indicator focuses on the average signal-to-noise ratio value contributed by all pixels.

Table 4. Average time to achieve specified SSIM threshold on 3 datasets compared learning-based methods.

Methods	**Animations**		**Art**		**Realistic photos**	
	Num	time	Num	time	Num	time
DiffVG	1000	20.3m	1000	30.5m	1000	35.6m
LIVE	1000	23.5h	1000	73.5h	1000	89.5h
O&R	1000	24.3m	1000	32.2m	1000	45.9m
FTPD	1000	**14.5m**	1000	**19.5m**	1000	**27.8m**
CRSPD	522	16.8m	2069	24.5m	3848	32.9m

Primitives Optimization Time Comparison: The primitives optimization time consumed for image vectorization is also important, we compare average value for the learning-based methods including DiffVG, LIVE, O&R and our TIFVec.

We test the primitives optimization time to achieve a specified SSIM threshold on the Animations, Art, Realistic photos datasets. Here, we set the SSIM threshold to 0.90. It is worth noting that we do not limit the number of iterations for the rendering stage until the SSIM threshold is reached. In addition, we also present the average number of primitives, since the number of primitives in CRSPD strategy is not fixed. As shown in Table.4, the time consumed by O&R and DiffVG is very close, with Live taking the longest time. Our method takes the shortest average time to reach the SSIM threshold in the 3 datasets. This is because our method focuses on areas with high texture intensity, which avoids the time loss caused by correcting details. In addition, LIVE is also an iteration-based method, but each time it optimize a primitive, it takes 500 iterations (1-5 seconds for each time). Therefore, optimizing 1,000 primitives takes 500,000 iterations, which results in the huge time cost. The previous work [21] also shows that the LIVE method takes an average of 25.88 hours in the process of vectorizing manga, which is intolerable.

4.5 Ablation Experiments

We implement the ablation experiments on the Emoji dataset and the Animations dataset. The configurations in Table.5 display the combination of each component of our method. In the case of using strategy FTPD on the Emoji dataset, we set the number of vector primitives to 16, while on the Animations dataset, we set it to 1000.

The experimental results demonstrate that our proposed FTPD and CRSPD strategies, along with the GLCM function, all contribute to improving the accuracy of vectorization. Among them, the GLCM function exhibits the greatest improvement in accuracy. This is because during the image optimizing process, we pay more attention to regions with higher texture intensity which refers to the detailed parts, thereby avoiding the loss of details. Additionally, our FTPD and CRSPD strategies ensure a higher distribution of primitives in regions with greater texture intensity, further enhancing the accuracy of vectorization.

Table 5. Ablation Experiments on 2 datasets. AD represents average distribution.

configuration		**Emoji**		**Animation**	
distribution	loss	PSNR	SSIM	PSNR	SSIM
AD	MSE	32.12	0.790	33.47	0.913
FTPD	MSE	32.28	0.795	33.81	0.936
CRSPD	MSE	32.83	0.939	33.60	0.934
AD	GLCM	34.45	0.960	36.11	0.989
FTPD	GLCM	35.87	0.968	36.81	0.992
CRSPD	GLCM	36.65	0.995	36.30	0.991

4.6 Analysis of the Number of Primitives

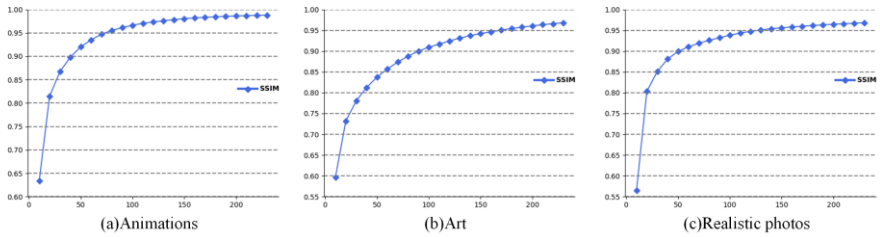

Fig. 6. Minimum number of primitives required to achieve a specific SSIM on each dataset, x-axis is primitive number.

To improve the model's practicality, we conducted experiments to determine the minimum number of primitives needed to achieve a specific SSIM, which affects the editability and compactness of vectorized images. We followed the FTPD strategy and initially set the number of primitives to 10. Once stability was reached, we added 10 more primitives to the current vector output and repeated this process, recording the number of primitives and corresponding SSIM values at each stability point. The results for each dataset are shown in Fig.6, with the vertical axis representing the SSIM threshold and the horizontal axis representing the minimum number of primitives required to reach that threshold. We observed that as the number of primitives increased, so did the stable SSIM value. However, this increase eventually slowed down and converged.

5 Conclusion

In this work, we propose a vectorization framework based on the texture intensity field. The framework includes 2 strategies for distributing primitives, and our experiments show that the initial primitive distribution, number of primitives, and image texture significantly impact the quality of image vectorization. Additionally, we introduce a TIF-based objective function to emphasize areas with higher texture intensity during primitive optimizing process. Extensive experiments demonstrate the effectiveness of our TIFVec framework, which achieves the state-of-the-art performance.

Acknowledgments. This research is partially supported by National Natural Science Foundation of China (Grant No. 62271360) and Natural Science Foundation of Hubei Province(Grant No.2025AFB950).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Hettinga, G., Echevarria, J., Kosinka, J.: Efficient image vectorisation using mesh colours. In: Italian Chapter Conference 2021: Smart Tools and Apps in Graphics. Eurographics Association (2021)
2. Xie, G., Sun, X., Tong, X., Nowrouzezahrai, D.: Hierarchical diffusion curves for accurate automatic image vectorization. ACM Transactions on Graphics (TOG) 33(6), 1–11 (2014)
3. Li, T.M., Lukáč, M., Gharbi, M., Ragan-Kelley, J.: Differentiable vector graphics rasterization for editing and learning. ACM Transactions on Graphics (TOG) 39(6), 1–15 (2020)
4. Ma, X., Zhou, Y., Xu, X., Sun, B., Filev, V., Orlov, N., Fu, Y., Shi, H.: Towards layer-wise image vectorization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16314–16323. IEEE (2022)
5. Lopes, R.G., Ha, D., Eck, D., Shlens, J.: A learned representation for scalable vector graphics. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7930–7939 (2019)
6. Zhou, H., Zheng, J., Wei, L.: Representing images using curvilinear feature driven subdivision surfaces. IEEE transactions on image processing 23(8), 3268–3280 (2014)
7. Liao, Z., Hoppe, H., Forsyth, D., Yu, Y.: A subdivision-based representation for vector image editing. IEEE transactions on visualization and computer graphics 18(11), 1858–1867 (2012)
8. Price, B., Barrett, W.: Object-based vectorization for interactive image editing. The Visual Computer 22, 661–670 (2006)
9. Lai, Y.K., Hu, S.M., Martin, R.R.: Automatic and topology-preserving gradient mesh generation for image vectorization. ACM Transactions on Graphics (TOG) 28(3), 1–8 (2009)
10. Sun, J., Liang, L., Wen, F., Shum, H.Y.: Image vectorization using optimized gradient meshes. ACM Transactions on Graphics (TOG) 26(3), 11–es (2007)

11. .Xia, T., Liao, B., Yu, Y.: Patch-based image vectorization with automatic curvilinear feature alignment. ACM Transactions on Graphics (TOG) 28(5), 1–10 (2009)
12. Orzan, A., Bousseau, A., Winnemöller, H., Barla, P., Thollot, J., Salesin, D.: Diffusion curves: a vector representation for smooth-shaded images. ACM Transactions on Graphics (TOG) 27(3), 1–8 (2008)
13. Egiazarian, V., Voynov, O., Artemov, A., Volkhonskiy, D., Safin, A., Taktasheva, M., Zorin, D., Burnaev, E.: Deep vectorization of technical drawings. In: European conference on computer vision. pp. 582–598. Springer (2020)
14. Ribeiro, L.S.F., Bui, T., Collomosse, J., Ponti, M.: Sketchformer: Transformer-based representation for sketched structure. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 14153–14162. Computer Vision Foundation / IEEE (2020)
15. Carlier, A., Danelljan, M., Alahi, A., Timofte, R.: Deepsvg: A hierarchical generative network for vector graphics animation. Advances in Neural Information Processing Systems 33, 16351–16361 (2020)
16. Hirschorn, O., Jevnisek, A., Avidan, S.: Optimize & reduce: A top-down approach for image vectorization. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 2148–2156 (2024)
17. Mohanaiah, P., Sathyanarayana, P., GuruKumar, L.: Image texture feature extraction using glcm approach. International journal of scientific and research publications 3(5), 1–5 (2013)
18. LI Rui-long, L.Y., Song-hai, Z.: Cartoon animations segmentation and vectorization based on canny optimization. Computer Science 44(8), 27 (2017). https://doi.org/10.11896/j.issn.1002-137X.2017.08.005
19. Reddy, P., Gharbi, M., Lukac, M., Mitra, N.J.: Im2vec: Synthesizing vector graphics without vector supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7342–7351. Computer Vision Foundation / IEEE (2021)
20. Bolelli, F., Allegretti, S., Baraldi, L., Grana, C.: Spaghetti labeling: Directed acyclic graphs for block-based connected components labeling. IEEE transactions on image processing 29(1), 1999–2012 (2019)
21. Su, H., Niu, J., Liu, X., Cui, J., Wan, J.: Vectorization of raster manga by deep reinforcement learning. CoRR abs/2110.04830 (2021), https://arxiv.org/abs/2110.04830