HDLayout: Hierarchical and Directional Layout Planning for Arbitrary Shaped Visual Text Generation

Tonghui Feng[1], Chunsheng Yan[2], Qianru Wang[1], Jiangtao Cui[1], Xiaotian Qiao[1,2*]

[1] School of Computer Science and Technology, Xidian University, China
[2] Guangzhou Institute of Technology, Xidian University, China
thfeng@stu.xidian.edu.cn, qiaoxiaotian@xidian.edu.cn

Abstract

Visual text generation, which aims to generate photo-realistic images with coherent and well-formed scene text being rendered, has attracted widespread attention. Although recent works have achieved promising performance, the limited flexibility and controllability hinder their practical applications. We observe that different from natural objects, visual text in real scenes often has an arbitrarily shaped structure with different granularities (i.e., character, word, or line). In this paper, we consider the modality gap between image and text, and propose a new separation and composition pipeline for flexible and controllable visual text generation from only text prompts. At the core of our framework is a novel **H**ierarchical and **D**irectional **Layout** representation, i.e., *HDLayout*, which can model the sequential and multi-granularity nature of the visual text. Under this formulation, we are able to generate arbitrarily shaped visual text automatically. Extensive experiments demonstrate that our method outperforms several strong baselines in a variety of scenarios both qualitatively and quantitatively, yielding state-of-the-art performances on arbitrarily shaped visual text generation.

Introduction

Text descriptions, e.g., street and storefront signs, banners, and book covers, exist widely in real-world scenarios and act as an important information carrier in facilitating visual communication. With the rapid development of generative models, astounding advances have been achieved in generating photo-realistic and prompt-aligned scene images (AI 2024; OpenAI 2023; Zheng et al. 2023; Yang et al. 2022) in terms of fidelity and diversity. Despite showcasing impressive performance, existing generative models still face a persistent challenge in generating clear and contextually appropriate text on scene images, due to the huge modality gap between abstract text and natural objects. Thus, how to generate well-formed and readable visual text to improve the scene image quality is still an open and crucial problem.

Until recently, several models have emerged to address this problem by using the predefined text layout as an additional input or distilling the text prompt to predict bounding boxes automatically. GlyphDraw (Ma et al. 2023) enhances text generation capabilities through simple glyph guidance to generate a single row or column of text. GlyphControl (Yang et al. 2024) exhibits robust text generation capabilities via diverse user-defined layout shapes. TextDiffuser (Chen et al. 2024) leverages a transformer to achieve multi-line text rendering, by predicting various word-level layouts automatically.

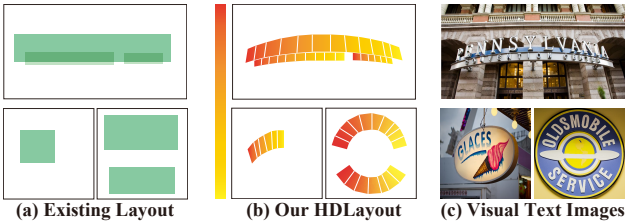

Figure 1: Differences between (a) standard textual layout representation and (b) the proposed *HDLayout* representation. Existing works use rectangular bounding boxes to represent either character or word level visual text, lacking shape details and different granularities. In contrast, we propose *HDLayout*, a novel hierarchical and directional layout representation that can model both geometric and sequential information simultaneously, enabling flexible and controllable arbitrarily shaped visual text generation (c).

Nevertheless, we have noticed several drawbacks that hinder these models from realizing their full potential. **(1) Additional input requirement.** Most existing works require users to provide manual layout guidance for visual text rendering. Such a requirement prevents them from converting user prompts to scene images automatically, seriously limiting the flexibility of text styles. **(2) Rectangular bounding box representation.** As shown in Fig. 1(a), existing works follow the scene layout modeling pipeline, and use rectangular bounding boxes to represent either character or word level text. However, such a representation lacks geometric details of the visual text, and is quite different from real scene text.

From our study, we have investigated the distinctive characteristics of visual text that set them apart from natural objects in a scene, and made the following observations. **(1) Arbitrarily shaped structure.** Visual text in real scenes often has various layout structures (e.g., horizontal, multi-oriented, curved, etc.), consisting of a set of arbitrarily

*Corresponding author.
Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

shaped text lines. Existing rectangular bounding box representation cannot fully capture such a characteristic. **(2) Multi-granularity.** Visual text often appears with the correlation of intrinsic structural information across different text granularities, e.g., character, word, or line in scene images. Manually defining the text positions and sizes at multiple levels would be labor-consuming. It would be important to learn to predict multi-granularity layout automatically for fine-grained and controllable visual text generation.

Inspired by the above observations, in this paper, we make the first attempt to address the visual text generation problem from a novel layout perspective. To this end, we represent the visual text structure as **H**ierarchical and **D**irectional Layout, i.e., *HDLayout*, a novel layout representation that contains fine-grained geometric details of the textual region. Based on the new layout representation, we introduce a flexible and controllable framework for arbitrarily shaped visual text generation. In particular, our framework contains three main modules. First, we separate the input prompt into two parts. One is the textual description of the image, and the other is the keywords labeled by the quotation mark. We adopt a pretrained Text-to-Image (T2I) module to generate the visual content. Second, we propose an HDLayout generation module to automatically predict the fine-grained visual text structures from both the keywords and the visual content obtained in the first module. Third, we composite the visual and textual information together to generate the visual text image via the Visual Text Render module.

To evaluate the effectiveness of our model, we conduct quantitative and qualitative experiments across various scenarios. To obtain the corresponding training dataset, we also construct a new HDLayout3k dataset with diverse and arbitrarily shaped text structures. Experimental results show that our *HDLayout* can capture the unique characteristics of textual content, significantly enhancing the fine-grained rendering quality and controllability of visual text generation.

In summary, the main contributions of this work are:

- To the best of our knowledge, we make the first effort to solve the arbitrarily shaped visual text generation problem from a novel layout perspective. A hierarchical and directional layout (i.e., *HDLayout*) representation is proposed to model the unique characteristics of visual text.
- We propose a new separation and composition framework to generate the textual and visual information separately, enabling flexible and controllable visual text generation from only text prompts.
- We construct a new HDLayout3k dataset with diverse and arbitrarily shaped text. The extensive experiments show that our method achieves state-of-the-art performances in arbitrarily shaped visual text generation.

Related Work

Image Generation. Early works mainly explore various generative adversarial networks or variational autoencoder designs. PLGAN (Wang et al. 2022a) is a GAN-based image synthesis method that distinguishes between instance and stuff. Recent works have made significant performance improvements, due to the power of diffusion models. DALL·E3 (OpenAI 2023) and FLUX.1 (Black-Forest-Labs 2024) have demonstrated improved high-resolution text-to-image generation quality, displaying great coherence and accuracy in generating scene text images. Imagen (Saharia et al. 2022) takes advantage of the capabilities of large transformer language models T5 (Raffel et al. 2020) to comprehend text while relying on the robustness of diffusion models for generating high-fidelity images. However, all these works focus on visual content generation. In contrast, our work focuses on textual content generation, i.e., visual text on scene images, which are very different from natural objects in visual, structural, and semantic characteristics.

Visual Text Generation. Early works (Yang, Huang, and Lin 2020) mainly focus on scene text editing, where models learn to render a reference text style on a target image. With the significant progress of diffusion models, several deep models have emerged to address the visual text generation problem by using the predefined text layout as an additional input or distilling the text prompt to predict bounding boxes automatically. GlyphControl (Yang et al. 2024) combines Stable Diffusion (Rombach et al. 2022) and ControlNet (Zhang et al. 2023), enabling the model to produce diverse glyph layouts under user-guided text layout inputs. GlyphDraw (Ma et al. 2023) employs specific diffusion temporal images to construct text layouts to guide the visual text generation. However, GlyphDraw exhibits low quality when dealing with multiple text characters when generating diverse layouts. TextDiffuser (Chen et al. 2024) leverages a transformer to predict bounding boxes for each character, followed by diffusion-based image synthesis with the text description and character layout. The constraints imposed by character layout may have a negative impact on the quality of generated image parts. All of the above methods use the predefined text layout or distilling the text prompt to predict rectangular bounding boxes for visual text generation. In contrast, we consider the modality gap between image and text, and propose a separation and composition pipeline with the novel *HDLayout* representation, enabling arbitrarily shaped visual text generation from only text prompts.

Layout Generation. Our work is in line with a series of works on layout generation. LayoutGAN (Li et al. 2020) synthesizes layouts by modeling the geometric relationships of 2D elements, using a differentiable wireframe rendering layer and a CNN-based discriminator to optimize layouts in image space. LayoutTransformer (Gupta et al. 2021) introduced a self-attention-based framework for generating and completing scene layouts across various domains by learning the contextual relationships between layout elements. Wang et al. (Wang et al. 2022b) propose a content-aware layout generation network that synthesizes aesthetic text logo layouts by evaluating character placement trajectories and rendered shapes using a dual-discriminator module. Concurrent with our work, SceneVTG (Zhu et al. 2024) also enables arbitrarily shaped text generation with a multimodal large language model. In contrast to existing works that use rectangular bounding boxes to represent elements in the layout, we propose *HDLayout* to model the sequential and multi-granularity nature of the visual text.

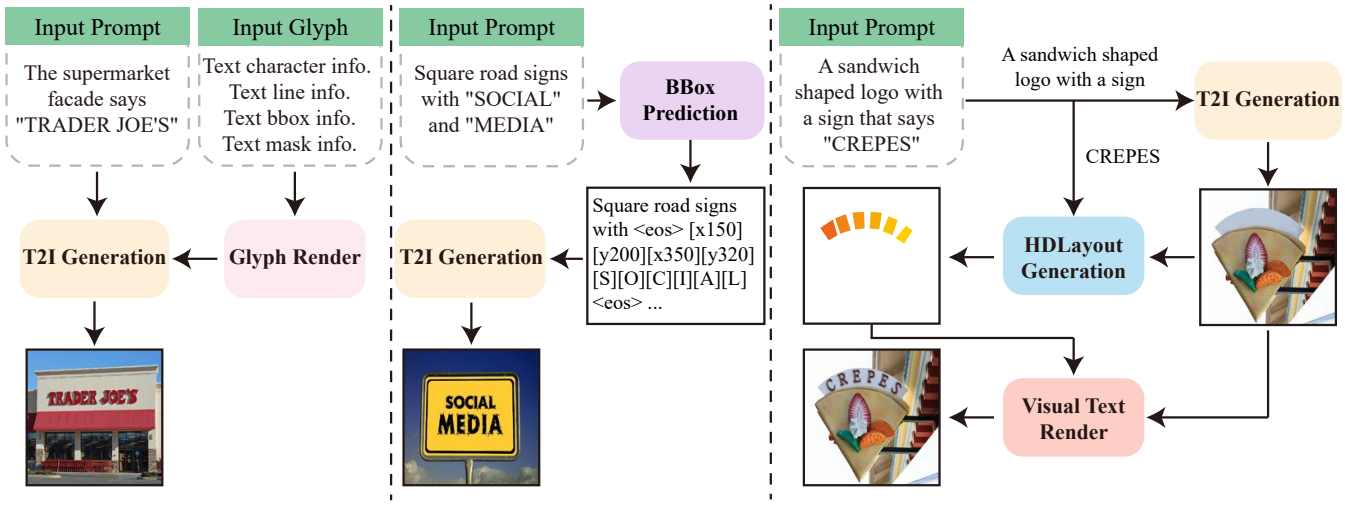

Figure 2: A brief comparison between existing pipelines (left & middle) and the proposed pipeline (right) for visual text generation. Existing works can be grouped by using the predefined Glyph Render (left) or distilling the text prompt to predict rectangular bounding boxes (middle), and then performing T2I generation. In contrast, our approach considers the modality gap between image and text, and proposes a separation and composition pipeline with the novel *HDLayout* representation, enabling arbitrarily shaped visual text generation from only text prompts.

Method

As depicted in Figure 2, the scene text rendering process initiates with the input prompt text being passed through the T2I Generation module, which meticulously generates a background image devoid of any text elements. Once the background image is created, it is then processed by the HDLayout generation module, which constructs a hierarchical and directional layout to ensure the accurate placement of characters. Following the integration of the rendered text with the background, the combined output is refined through the Visual Text Render module, which enhances the overall clarity and sharpness of the text.

T2I Generation Module

In this module, the prompt text undergoes processing to generate a background image free of any text, preparing the canvas for character-level layout generation. The input prompt, denoted as T, specifies the words intended for rendering in the image, formatted as "xxx". For instance, *A sandwich-shaped logo with a sign that says "CREPES"*. To ensure that the generated background image remains text-free, we adopt two approaches: removing text-related content directly from the prompt and applying PP-OCRv3 (Li et al. 2022) to detect and regenerate the image containing text. Subsequently, a pretrained Diffusion model is employed to generate a background image $I_{bg} \in \mathbb{R}^{3\times H\times W}$, where $H \times W$ denotes the image dimensions and 3 represents the color channels.

HDLayout Generation Module

The HDLayout generation module utilizes the content-aware properties of the background image to iteratively refine and generate character-level Bézier curve layouts. This process is illustrated in detail in Figure 3, which showcases the comprehensive HDLayout generation module pipeline.

In particular, we first generate coarse region-level bounding boxes, which are progressively refined into line-level bounding boxes. We then employ Bézier curves to adjust the text generation area, ultimately leading to the segmentation of the Bézier-controlled regions based on the number of characters. By employing a hierarchical optimization strategy, the model can ensure a contextually accurate and appropriate character-level layout within the image. Figure 4 presents a series of *HDLayout* results that contain the region-level, line-level, and character-level text structures.

Image Encoder. This module utilizes a conventional CNN encoder as its backbone to process the input background image I_{bg}, reducing its dimensions to $H_e \times W_e$. This results in an encoded feature map $f \in \mathbb{R}^{C\times H_e\times W_e}$, where $C = 2048$ and $H_e, W_e = 16, 16$.

Attention Mechanism. The decoders across the region, line, and character levels all share a unified architecture based on the standard Transformer decoder. Each input sequence undergoes a series of transformations through multiple layers of self-attention, cross-attention, and feed-forward networks (FFNs), ultimately producing the feature vector hs. A distinctive feature of our approach is the use of only $N = 2$ decoder layers at each level.

Region Decoder. This module initiates by generating an all-zero input sequence $t_r \in \mathbb{R}^{M_r\times E_r}$, where M_r denotes the anticipated number of region-level bounding boxes. Prior to feeding sequences into the region attention module, the feature map f is first embedded via the region embedding layer, resulting in z_r:

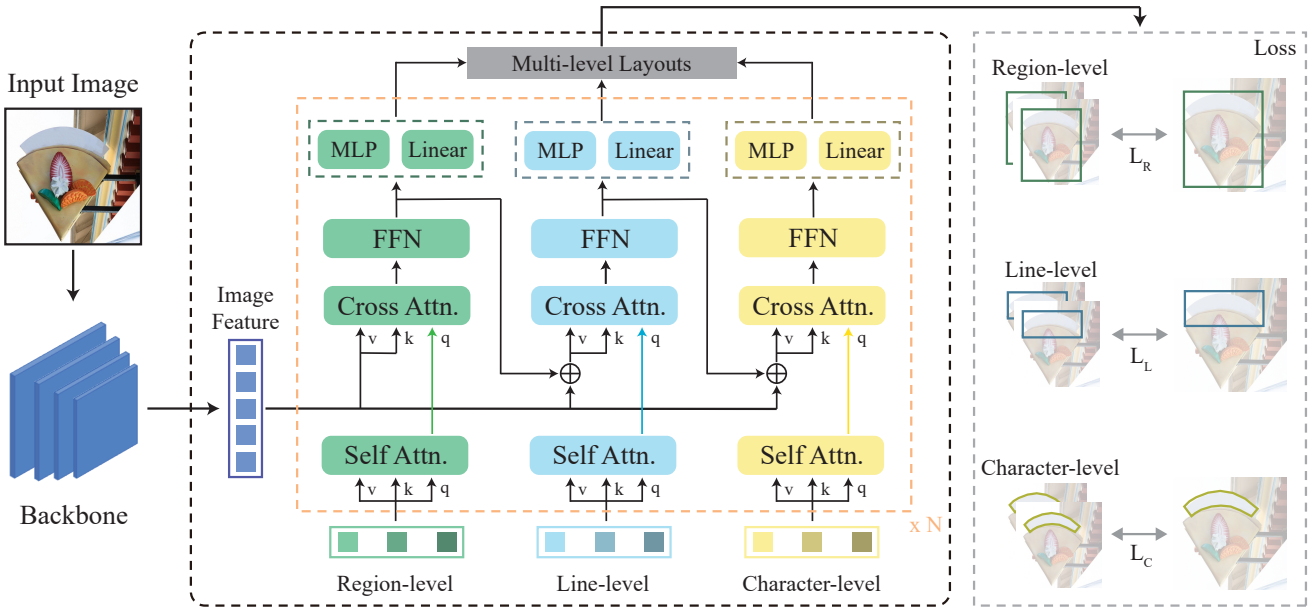

Figure 3: The overall pipeline of the HDLayout generation module. Given an input image and the keywords, we first extract image features from a backbone network. We then progressively refine the *HDLayout* in a coarse-to-fine manner, including region-level, line-level, and character-level. Note that we use rectangular bounding boxes to represent both the region-level and line-level text structures, and use Bézier curves to represent the character-level varieties.

Figure 4: Results of the generated *HDLayout* representation. Given the input image (top), our HDLayout generation module is able to generate hierarchical and directional layout structures (bottom) for the visual text. Bounding boxes with blue and green colors represent the region-level and line-level structure respectively, while irregular shapes with red color represent the character-level structure.

$$z_r = Region_{emb}(f), \tag{1}$$

where $z_r \in \mathbb{R}^{E_r \times (HW)}$, E_r is the region embedding dimension, and $HW = H_e \times W_e = 256$, which is consistent across all levels. After z_r embedded in multi-cross-attention, the input sequence vectors are processed to obtain the region feature vector hs_r:

$$hs_r = Region_{dec}(t_r, z_r, p_r), \tag{2}$$

where $hs_r \in \mathbb{R}^{N \times M_r \times E_r}$, $p_r \in \mathbb{R}^{M_r \times E_r}$ represents the learnable positional encoding. The feature vector hs_r is further decoded by an MLP, producing the region bounding boxes $Box_r \in \mathbb{R}^{N \times M_r \times 4}$, with the four bounding box values corresponding to (x_1, y_1, x_2, y_2). Additionally, a fully connected layer (FC) is applied to hs_b to generate the confidence scores $P_r \in \mathbb{R}^{N \times M_r \times 1}$ for each bounding box.

Line Decoder. This module adopts the same input structure as the region decoder, generating the input $t_l \in \mathbb{R}^{M_l \times E_l}$, where M_l represents the number of line bounding boxes. The line embedding layer is integrated with the region feature vector hs_r to acquire z_l:

$$z_l = \lambda_1 BL(hs_r) + Line_{emb}(f), \tag{3}$$

where $z_l \in \mathbb{R}^{E_l \times HW}$, E_l represents the embedding dimension at the line-level, $\lambda_1 = 1.5$ is a weighting coefficient, and BL is the region output feature encoding function. Similar to the region decoder, the line decoder processes the input sequence to generate the line-level feature vector hs_l,

$$hs_l = Line_{dec}(t_l, z_l, p_l), \tag{4}$$

where $hs_l \in \mathbb{R}^{N \times M_l \times E_l}$, $p_l \in \mathbb{R}^{M_l \times E_l}$ signifies the learnable positional encoding. After hs_l further passing through an MLP and an FC, the model outputs the line-level bounding boxes $Box_l \in \mathbb{R}^{N \times M_l \times 4}$ and their corresponding confidence scores $P_l \in \mathbb{R}^{N \times M_l \times 1}$.

Character Decoder. This module's input structure mirrors that of the region decoder, generating the input $t_c \in \mathbb{R}^{M_c \times E_c}$, where M_c represents the expected number of

Bézier curve. The character embedding layer is combined with the line-level feature vector hs_l to encode z_c,

$$z_c = \lambda_2 LB(hs_l) + Char_{emb}(f), \tag{5}$$

where $z_c \in \mathbb{R}^{E_c \times HW}$, E_c represents the embedding dimension, $\lambda_2 = 3$ is a coefficient, and LB is the line output feature encoding function. As in the line decoder, the character decoder processes the input sequence to obtain the character-level feature vector hs_c,

$$hs_c = Char_{dec}(t_c, z_c, p_c), \tag{6}$$

where $hs_c \in \mathbb{R}^{N \times M_c \times E_c}$ and $p_c \in \mathbb{R}^{M_c \times E_c}$ represents the positional encoding. An MLP is used to produce the Bézier curves $C \in \mathbb{R}^{N \times M_c \times 16}$, with the 16 Bézier curve values representing the 4 control points $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)_{upper}$ for the upper curve and the 4 control points $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)_{lower}$ for the lower curve. Additionally, hs_c is passed through an FC layer to obtain the confidence score $P_c \in \mathbb{R}^{N \times M_c \times 1}$.

Text Rendering Module

The primary task of this module is to leverage the generated layout to correctly render the text specified in the prompt at the designated positions on the background image. Initially, the Bézier curves, generated in earlier modules, are segmented at the character-level according to the number of characters specified in the prompt, ensuring that each character aligns with a distinct polygon layout. Subsequently, using existing font libraries, the individual characters are sequentially embedded into their respective positions on the background image I_{bg}, culminating in a pre-populated scene text image I_{wt}, where $I_{wt} \in \mathbb{R}^{3 \times H \times W}$. Finally, both I_{wt} and I_{bg}, along with the entire prompt text, are processed through the Diffusion model, producing the final scene text image with the text accurately rendered.

Loss Function

We utilize the layout of bounding boxes within the same hierarchical level to macroscopically constrain the layout of Bézier curves, allowing the Bézier curves to be adjusted and refined based on the bounding boxes. Additionally, inspired by the auxiliary decoding loss and matching loss in DETR (Carion et al. 2020), we compute the loss for each decoder layer's output and calculate the matching loss between outputs and ground truth, using the Hungarian algorithm for matching to obtain the final loss. Moreover, we utilize a cross-entropy loss to predict the confidence of each bounding box and Bézier curve.

Bounding Box Loss. To effectively perceive the image, we calculate the L1 loss for the generated bounding boxes:

$$L_{L1} = \frac{1}{N} \sum_{i=1}^{N} |\hat{B} - B|, \tag{7}$$

where N denotes the matched number of bounding boxes, $\hat{B}$ and B represent the coordinates of the i^{th} predicted and ground truth bounding box seperately.

The GIoU loss is particularly effective for improving the accuracy of bounding box localization, especially in cases where the boundaries are close but not fully overlapping. GIoU extends the standard IoU by incorporating the distance between the predicted and ground truth bounding boxes, leading to better convergence properties:

$$\text{GIoU} = \frac{|\hat{B}| \cap |B|}{|\hat{B}| \cup |B|} - \frac{|C| - (|\hat{B}| \cup |B|)}{|C|}, \tag{8}$$

$$L_{\text{GIoU}} = \frac{1}{N} \sum_{i=1}^{N} (1 - \text{GIoU}(\hat{B}, B)), \tag{9}$$

where $|C|$ denotes the area of the smallest enclosing box that contains both $\hat{B}$ and B.

To prevent the model from generating multiple bounding boxes in the same location, we calculate the overlap loss for all generated bounding boxes. By minimizing the overlap loss, the model generates bounding boxes with a well-distributed spatial arrangement, improving the clarity and readability of the text layout. The overlap loss ensures that the generated bounding boxes maintain a reasonable layout structure by avoiding mutual occlusion:

$$L_{\text{ol}} = \frac{2}{M(M-1)} \sum_{i=1}^{M} \sum_{j=i+1}^{M} \text{IoU}(B_i, B_j), \tag{10}$$

where M is the number of generated bounding boxes.

The total bounding box loss is calculated as follows:

$$L_{\text{bbox}} = cost_{\text{bbox}} = c_1 L_{L1} + c_2 L_{\text{GIoU}} + c_3 L_{\text{ol}}, \tag{11}$$

where c_1, c_2, c_3 are coefficients set to $c_1 = 5, c_2 = 2, c_3 = 1$, reflecting the relative importance of each loss component. $cost_{bbox}$ serves as the basis for the Hungarian algorithm in matching bounding box pairs.

Bézier Loss. Currently, there is no ideal loss function specifically designed for constraining Bézier curves, so we use the simple approach by employing the L1 loss.

$$L_{\text{Bézier}} = cost_{\text{Bézier}} = L_{L1}, \tag{12}$$

where $c_4 = 5$. The term $cost_{\text{Bézier}}$ represents the basis for the Hungarian algorithm in matching Bézier curve pairs.

Confidence Loss. The confidence score is primarily used to determine whether a bounding box or Bézier curve should be selected from the generated candidates. We compute the confidence loss using the standard cross-entropy loss:

$$L_{\text{conf}} = -p \cdot \log(q), \tag{13}$$

where q represents the confidence score predicted by the model and $p = 1$ is the ground truth label indicating the presence of the object.

The combined loss function is given by:

$$L = \lambda_1 L_{\text{bbox}} + \lambda_2 L_{\text{Bézier}} + \lambda_3 L_{\text{conf}}, \tag{14}$$

where $\lambda_1 = 1$, $\lambda_2 = 5$, and $\lambda_3 = 1$.

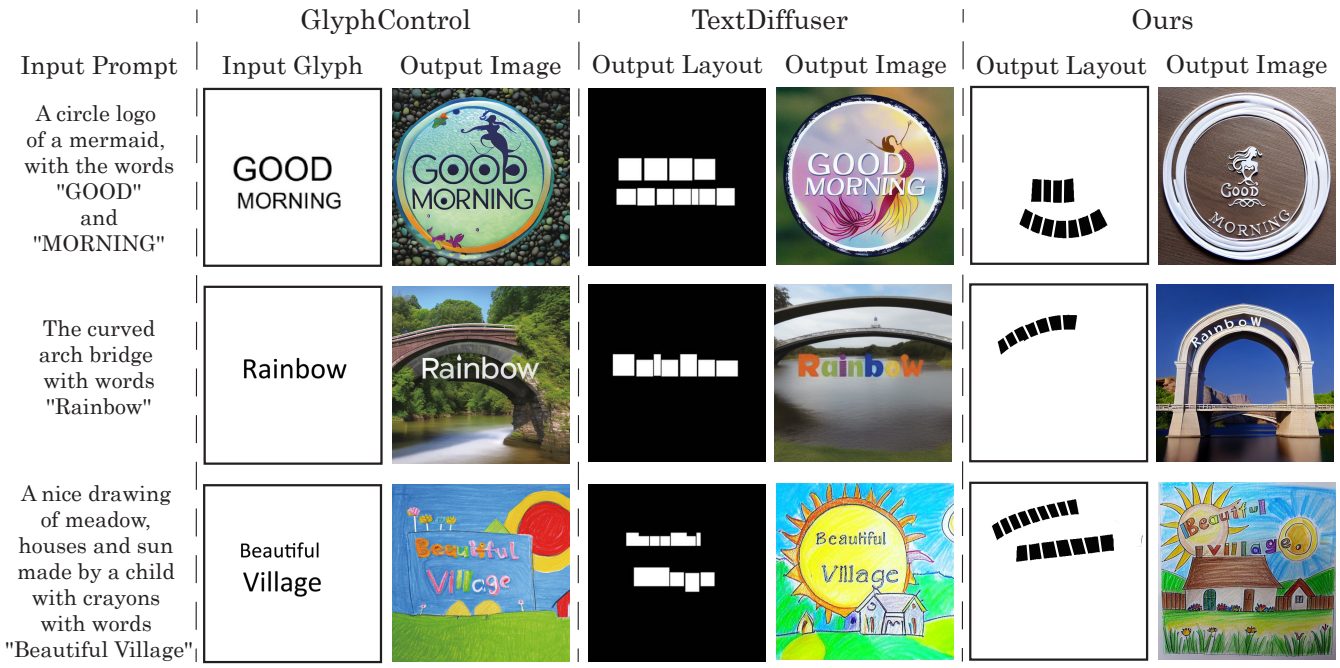

Figure 5: Qualitative comparison between the baselines and our model. Given the input prompt (1st column), we show the generated visual text results from baselines (3rd and 5th columns) and our model (7th column). Note that GlyphControl (Yang et al. 2024) needs an additional glyph image (2nd column) as input. We further show the layouts predicted by TextDiffuser (Chen et al. 2024) (4th column) and our model (6th column).

Experiment

Dataset. We construct the HDLayout3k dataset for this problem. It is derived from SCUT-EnxText (Liu et al. 2020) by reorganizing all word-level polygon data. At the region-level, bounding boxes are defined using the top, bottom, left, and right edges of all polygon points. At the line-level, we compute the minimum enclosing box for each polygon as the bounding box. At the character-level, Bézier curves are fitted to the segments of the polygon's top (or left) and bottom (or right) boundaries. The two sets of Bézier control points, which respectively represent the two curve boundaries, allowing for a smooth representation of the character-level structure. The HDLayout3k dataset comprises 2,749 training data and 813 test data. To make fair comparisons, we further utilize the Anytext-benchmark (Tuo et al. 2023) as our image generation test dataset. This dataset contains 1000 images with scene text, along with corresponding prompt descriptions, the text to be rendered, and their positions.

Implementation Details. During the training phase, we conduct layout generation training for 600 epochs using the HDLayout3k dataset on one RTX 4090. We use AdamW optimizer with a learning rate of 1e-4, a weight decay of 1e-4, and a batch size of 2. The CNN encoder in our architecture is based on a trainable ResNet50 with a learning rate of 1e-5 and the input image size is 512×512.

Evaluation Metrics. We utilize FID (Heusel et al. 2017) to calculate the similarity between generated and real images. FID extracts feature vectors from images and computes the Fréchet distance between these feature distributions, with a lower FID indicating a higher resemblance between generated and real images. Furthermore, to demonstrate the significant advantage of our model in layout generation, we adopt the Layout-FID metric, similar to the one used in LayoutDETR (Yu et al. 2022), to evaluate the quality of generated layouts. Like FID, Layout-FID aims to capture the discrepancy between generated and real layouts to assess the performance of the layout generation model. In this evaluation, we compute Layout-FID using the generated and the ground truth layout mask images, with FIDNetV3 (Inoue et al. 2023) serving as the feature extractor.

Compared Methods. Our method begins by using an adjusted prompt to generate a background image without text through a diffusion model. Subsequently, the HDLayout generates a plausible text layout based on this background image. Finally, the complete prompt, background image, and text layout are processed together into the diffusion model to produce the final scene text content. GlyphControl (Yang et al. 2024) initially renders glyph images according to the provided glyph instruction and combines them with prompt features to control the diffusion model for generating scene text images. TextDiffuser (Chen et al. 2024) employs a transformer to predict text layout from the prompt and utilizes the Pillow to obtain character layout bounding boxes, which are then input into the diffusion model to generate scene text images. While GlyphControl (Yang et al. 2024) requires predesigned layouts, TextDiffuser (Chen et al. 2024) and our method rely solely on the prompt.

Methods	FID (↓)	
	Image FID	Layout FID
SD-2.1 (Rombach et al. 2022)	92.462	-
SD-XL (Podell et al. 2023)	102.128	-
ControlNet (Zhang et al. 2023)	89.802	-
GlyphControl (Yang et al. 2024)	83.402	-
TextDiffuser (Chen et al. 2024)	82.068	27.135
Ours	**78.027**	**12.343**

Table 1: Quantitative comparison of the proposed method with the baselines. We evaluate their performances using Image FID and Layout FID scores. The best results are highlighted in bold.

HDLayout	FID (↓)	
	Image FID	Layout FID
w/o Region & Line	115.84	16.474
w/o Region	102.67	16.094
w/o Line	113.13	15.697
Ours (Layout) + ControlNet	88.920	-
Ours (Layout) + GlyphControl	86.842	-
Ours (Full)	**78.027**	**12.343**

Table 2: Results of the ablation study. The best results are highlighted in bold.

Qualitative Evaluation

We perform an in-depth comparative analysis against leading state-of-the-art (SOTA) models in the text-to-image task. As depicted in Figure 5, GlyphControl (Yang et al. 2024) requires additional glyph inputs, which underscores the challenges faced by models that rely on rigid input structures. Moreover, TextDiffuser (Chen et al. 2024), while more flexible in generating layouts based on prompts, is still restricted to arranging characters in a linear fashion, thus failing to support more complex text layouts such as curves or inclined text. In contrast, our model stands out by leveraging only standard prompts to autonomously generate a wide range of text layouts, including curved and tilted configurations. This flexibility allows our model to adapt the text placement in harmony with the background image's content, ensuring a seamless integration of text within the visual context. The resulting images not only adhere closely to the user's prompt but also exhibit a natural and aesthetically pleasing layout that enhances the overall image composition, demonstrating a clear advantage over existing SOTA approaches.

Quantitative Evaluation

We evaluate all methods using metrics on the Anytext-benchmark (Tuo et al. 2023) dataset. To ensure fairness, all compared scene rendering models are based on the SD-1.5 diffusion model, except for SD-2.1 (Rombach et al. 2022) and SD-XL (Podell et al. 2023). All methods use official weight files, with a fixed random seed of 100, a batch size of 4, 20 sampling steps, and identical prompts. The quantitative results are presented in Table 1.

The results demonstrate that our model achieves SOTA performance in image generation, surpassing SD-2.1 (Rombach et al. 2022), SD-XL (Podell et al. 2023), ControlNet (Zhang et al. 2023), GlyphControl (Yang et al. 2024), and TextDiffuser (Chen et al. 2024). This advancement is largely attributable to HDLayout's enhanced capability to generate more realistic layouts, including curves and slants. Furthermore, HDLayout exhibits superior layout generation capabilities compared to TextDiffuser (Chen et al. 2024). This advantage arises because these methods rely solely on bounding box layouts, resulting in more uniform and coarse layouts. In contrast, HDLayout generates Bézier layouts, which accurately reflect the true layout of the text.

Ablation Study

To validate the effectiveness of hierarchical generation over direct Bézier curve generation, we conducted an ablation study focusing on the region decoder and line decoder components, as presented in Table 2. We can see that the absence of both region-level and line-level results in the poorest performance in terms of both layout and image quality. It indicates that direct Bézier curve generation faces significant challenges in achieving optimal results without hierarchical guidance. When either the region-level or line-level is omitted, the model's performance improves, suggesting that both levels play a crucial role in Bézier curve generation. Our model is also used as plug-and-play guidance to replace the glyph control in existing works (Ours+ControlNet (Zhang et al. 2023), Ours+GlyphControl (Yang et al. 2024)). The results show the overall advantage of our model in visual text generation. Finally, when the full hierarchical generation approach is applied, both Image FID and Layout FID metrics reach their best values, further confirming the effectiveness of using a hierarchical method to indirectly control Bézier curve generation.

Conclusion

In this paper, we take a step towards arbitrarily shaped visual text generation. To this end, we propose a new separation and composition framework that captures the hierarchical and directional textual layout (i.e., *HDLayout*) representation, enabling flexible and controllable visual text generation from text prompts. Extensive qualitative and quantitative results show that our model outperforms several baselines on numerous scenarios with arbitrarily shaped visual text.

As the first attempt towards creating a novel layout representation for the arbitrarily shaped visual text generation, our method still has certain limitations. For small areas designated for text rendering, our method may not work well due to the difficulties in determining the accurate character positions. We believe that better modeling of the textual element relationships could be an important future direction.

Acknowledgments

This work was partially supported by the Guangdong Basic and Applied Basic Research Foundation (No. 2022A1515110740), and the National Natural Science Foundation of China (No. 62302356, No. 62372352, No. 62302367).

References

AI, S. 2024. Stable Diffusion 3: Advanced Text-to-Image Generation. https://stability.ai/news/stable-diffusion-3.

Black-Forest-Labs. 2024. FLUX.1: A Text-to-Image AI Model. https://flux-1.ai/.

Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; and Zagoruyko, S. 2020. End-to-end object detection with transformers. In *ECCV*.

Chen, J.; Huang, Y.; Lv, T.; Cui, L.; Chen, Q.; and Wei, F. 2024. Textdiffuser: Diffusion models as text painters. *NeurIPS*.

Gupta, K.; Lazarow, J.; Achille, A.; Davis, L. S.; Mahadevan, V.; and Shrivastava, A. 2021. Layouttransformer: Layout generation and completion with self-attention. In *ICCV*.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NeurIPS*.

Inoue, N.; Kikuchi, K.; Simo-Serra, E.; Otani, M.; and Yamaguchi, K. 2023. LayoutDM: Discrete Diffusion Model for Controllable Layout Generation. In *CVPR*.

Li, C.; Liu, W.; Guo, R.; Yin, X.; Jiang, K.; Du, Y.; Du, Y.; Zhu, L.; Lai, B.; Hu, X.; et al. 2022. PP-OCRv3: More attempts for the improvement of ultra lightweight OCR system. *arXiv preprint arXiv:2206.03001*.

Li, J.; Yang, J.; Hertzmann, A.; Zhang, J.; and Xu, T. 2020. Layoutgan: Synthesizing graphic layouts with vector-wireframe adversarial networks. *PAMI*.

Liu, C.; Liu, Y.; Jin, l.; Zhang, S.; Luo, C.; and Wang, Y. 2020. EraseNet: End-to-End Text Removal in the Wild. *TIP*.

Ma, J.; Zhao, M.; Chen, C.; Wang, R.; Niu, D.; Lu, H.; and Lin, X. 2023. GlyphDraw: Seamlessly Rendering Text with Intricate Spatial Structures in Text-to-Image Generation. *arXiv preprint arXiv:2303.17870*.

OpenAI. 2023. DALL·E 3. https://openai.com/dall-e-3.

Podell, D.; English, Z.; Lacey, K.; Blattmann, A.; Dockhorn, T.; Müller, J.; Penna, J.; and Rombach, R. 2023. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *CVPR*.

Saharia, C.; Chan, W.; Saxena, S.; Li, L.; Whang, J.; Denton, E. L.; Ghasemipour, K.; Gontijo Lopes, R.; Karagol Ayan, B.; Salimans, T.; et al. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *NeurIPS*.

Tuo, Y.; Xiang, W.; He, J.-Y.; Geng, Y.; and Xie, X. 2023. Anytext: Multilingual visual text generation and editing. *arXiv preprint arXiv:2311.03054*.

Wang, B.; Wu, T.; Zhu, M.; and Du, P. 2022a. Interactive image synthesis with panoptic layout generation. In *CVPR*.

Wang, Y.; Pu, G.; Luo, W.; Wang, Y.; Xiong, P.; Kang, H.; and Lian, Z. 2022b. Aesthetic text logo synthesis via content-aware layout inferring. In *CVPR*.

Yang, Q.; Huang, J.; and Lin, W. 2020. Swaptext: Image based texts transfer in scenes. In *CVPR*.

Yang, Y.; Gui, D.; Yuan, Y.; Liang, W.; Ding, H.; Hu, H.; and Chen, K. 2024. GlyphControl: Glyph Conditional Control for Visual Text Generation. *NeurIPS*, 36.

Yang, Z.; Liu, D.; Wang, C.; Yang, J.; and Tao, D. 2022. Modeling image composition for complex scene generation. In *CVPR*.

Yu, N.; Chen, C.-C.; Chen, Z.; Meng, R.; Wu, G.; Josel, P.; Niebles, J. C.; Xiong, C.; and Xu, R. 2022. LayoutDETR: detection transformer is a good multimodal layout designer. *arXiv preprint arXiv:2212.09877*.

Zhang, L.; Rao, A.; Agrawala, M.; et al. 2023. Adding conditional control to text-to-image diffusion models. In *ICCV*.

Zheng, G.; Zhou, X.; Li, X.; Qi, Z.; Shan, Y.; and Li, X. 2023. Layoutdiffusion: Controllable diffusion model for layout-to-image generation. In *CVPR*.

Zhu, Y.; Liu, J.; Gao, F.; Liu, W.; Wang, X.; Wang, P.; Huang, F.; Yao, C.; and Yang, Z. 2024. Visual text generation in the wild. *ECCV*.