LINR Bridge: Vector Graphic Animation via Neural Implicits and Video Diffusion Priors

Wenshuo Gao, Xicheng Lan, Luyao Zhang, Shuai Yang*
Wangxuan Institute of Computer Technology, Peking University, Beijing, China

***Abstract*—Vector graphics, known for their scalability and user-friendliness, provide a unique approach to visual content compared to traditional pixel-based images. Animation of these graphics, driven by the motion of their elements, offers enhanced comprehensibility and controllability but often requires substantial manual effort. To automate this process, we propose a novel method that integrates implicit neural representations with text-to-video diffusion models for vector graphic animation. Our approach employs layered implicit neural representations to reconstruct vector graphics, preserving their inherent properties such as infinite resolution and precise color and shape constraints, which effectively bridges the large domain gap between vector graphics and diffusion models. The neural representations are then optimized using video score distillation sampling, which leverages motion priors from pretrained text-to-video diffusion models. Finally, the vector graphics are warped to match the representations resulting in smooth animation. Experimental results validate the effectiveness of our method in generating vivid and natural vector graphic animations, demonstrating significant improvement over existing techniques that suffer from limitations in flexibility and animation quality.**

***Index Terms*—Vector Graphics, Implicit Neural Representation, Diffusion Model, Score Distillation Sampling**

I. Introduction

Vector graphics are a widely used form of flat visual content, distinct from traditional pixel-based raster images. Unlike raster images, vector graphics are typically composed of defined graphic elements, such as circles, squares, lines, and Bézier curves. They are characterized by simplicity, flatness, scalability without loss of resolution, and user-friendliness. As a result, vector graphics are extensively used in industrial and fine arts, such as icon design, animation, poster design and web design. Scalable Vector Graphics (SVG) [1] is a widely adopted XML-based representation for vector graphics.

Since vector graphics consist of graphic elements rather than pixels, their animation fundamentally differs from that of raster images. Vector graphic animations are created by transforming these graphic elements, providing greater comprehensibility and controllability. However, producing vector graphic animations that meet motion requirements while maintaining coordination and smoothness typically requires significant manual effort. Generally, creating a vector graphic animation involves first constructing a skeletal framework, followed by applying the corresponding transformations.

Therefore, a system capable of automating vector graphic animation is necessary. Recently, many text-to-video (T2V) models based on diffusion techniques like ModelScope [2], VideoCrafter2 [3], DynamiCrafter [4], and I2VGen-XL [5] have rapidly evolved and gained widespread applications. One approach is to simply render vector graphics into raster images and use pretrained T2V diffusion models to animate them using text prompts. However, since T2V models lack proper constraints on colors and shapes, and are typically trained on natural images rather than flat images, the results are often unstable and deviate from expectations [6], [7]. As a result, a specialized animation system for vector graphics is needed.

Although some methods for vector graphic animation have been proposed [6], [7], they have significant limitations. LiveSketch [6] animates vector graphics by directly optimizing SVG point parameters. However, it focuses on sketches and can only optimize global transformations and point-level local shifts, which lacks control over shapes, and leads to structural damage and unexpected motions. AniClipart [7] extracts the skeletal framework of SVG and applies As-Rigid-As-Possible (ARAP) deformation to animate cartoon art. However, it relies heavily on instance-dependent keypoint selection and skeletal movements, imposing overly rigid constraints on SVG deformation, which results in stiff motions. Moreover, both methods render parameterized SVG directly into raster images before feeding into diffusion models for optimization with video score distillation sampling (VSDS) [6]. This process introduces a significant domain gap, making optimization challenging and hindering the full utilization of the diffusion priors, ultimately preventing natural SVG animation.

This domain gap motivated our research. We observed that implicit neural representation (INR) [8]–[11], which is introduced and widely adopted in image and 3D modelling, is highly effective in reconstructing and animating vector graphics. INR models an image using a neural network, typically taking coordinate position information as input and producing corresponding pixel colors as output. INR has several advantageous characteristics. By using coordinates as input, INR constructs images with infinite resolution, aligning well with the nature of SVG. Moreover, INR offers shape manipulability, significantly bridging the domain gap between parameterized SVG and rasterized images. Our findings suggest that a color-wise layered implicit neural representation (LINR) can effectively reconstruct vector graphics, as they consist of various shapes, each assigned a specific color, enabling easier animation.

In this paper, we propose *LINR Bridge*, a novel coarse-to-

* Corresponding author

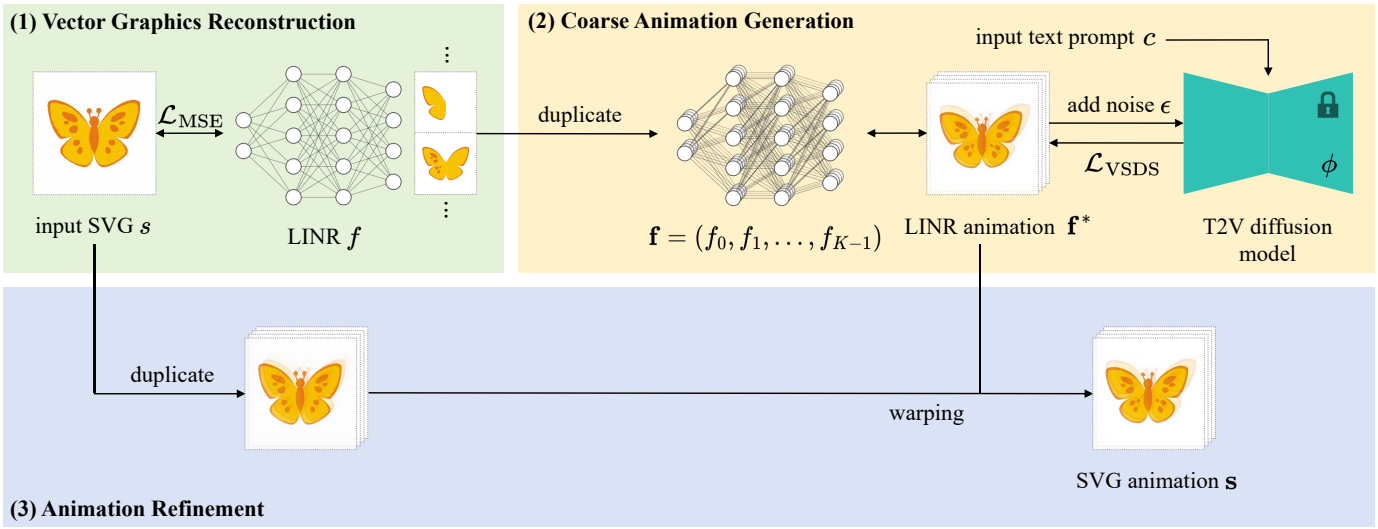

Fig. 1. The pipeline of *LINR Bridge* takes an SVG s and a text prompt c as inputs, and produces an SVG animation as output. The pipeline consists of three steps: (1) **Vector Graphics Reconstruction**: Optimize a LINR network f to reconstruct s. (2) **Coarse Animation Generation**: Replicate f K times to construct a K-frames initial video $\mathbf{f}$. Input the frames rendered from $\mathbf{f}$ along with the text prompt c into the T2V model and optimize using VSDS to obtain a coarse LINR animation. (3) **Animation Refinement**: Warp s based on optical flow to match the LINR animation, resulting in the final animated SVG $\mathbf{s}$.

fine framework for vector graphic animation that utilizes LINR to bridge the domain gap between vector graphic representations and existing T2V diffusion models. Specifically, we first reconstruct the input vector graphics using LINR. This LINR is then replicated across the video frame dimension and optimized using VSDS to generate a coarse LINR animation. The input vector graphics are warped based on optical flow to match the LINR animation, resulting in the final vector graphic animation. LINR effectively constrains the color and shape of the images, while the VSDS loss introduces motion priors from the pretrained T2V models. Experiments show that our method generates vivid and natural vector graphic animations. The rest of the paper is organized as follows. Sec. II introduces the proposed *LINR Bridge*. Sec. III presents experimental results, and Sec. IV provides concluding remarks.

II. Proposed Method

Our method automates the generation of K-frame animations for a given SVG using a text prompt. This approach consists of three steps: (1) **Vector Graphics Reconstruction**: Construct an LINR to reconstruct the input SVG; (2) **Coarse Animation Generation**: Replicate the LINR neural network K times and optimize it with a T2V model with the VSDS loss, to generate a coarse LINR animation; (3) **Animation Refinement**: Warp the original SVG to match the LINR animation, resulting in a smooth SVG animation. The pipeline is illustrated in Fig. 1.

A. Vector Graphics Reconstruction

Implicit neural representation (INR) [9] involves using a neural network to represent image information. Typically, this involves constructing a neural network to map coordinate inputs (X, Y) to color outputs (R, G, B). Since the input is continuous, this representation produces an image with infinite resolution, similar to SVG. We follow the common practice to use SIREN [12] as the neural network. SVG consists of various elements each defined by its own color. To better fit this layered and flat nature, we follow NIVeL [10] to replace the neural network outputs (R, G, B) with $(m_1, m_2, \ldots, m_L)$. Here, L is the number of layers, $m_i \in [0, 1]$ represents the coloring intensity for color c_i at the i-th layer. Such representation is called layered implicit neural representation (LINR), which enhances the model's ability to handle layered graphics efficiently.

Specifically, based on the input SVG s, we extract its color information based on layer stacking order, and merge adjacent regions with the same color. We denote the resulting color scheme as $(c_1, c_2, \ldots, c_L)$, with L representing the number of layers. We construct an LINR network f to map a coordinate input $\mathbf{p} = (X, Y)$ to an intensity vector $\mathbf{m} = (m_1, m_2, \ldots, m_L)$, with $m_i \in [0, 1]$. The process can be expressed as $\mathbf{m} = f(\mathbf{p})$.

We define c_0 as the background color and c_i as the color for the i-th layer of the SVG. The RGB image $C(f)$ represented by a LINR f is constructed as follows: For all coordinate points $\mathbf{p}$ in the canvas, we first apply the background color c_0, then overlay each layer in order, with the i-th layer contributing color c_i with intensity m_i. Here, an intensity of 0 means no color is applied, while an intensity of 1 means full color is applied, resulting in the RGB image $C(f)$.

The first stage optimizes the LINR f to reconstruct the input SVG s with the following objectives:

- Layered reconstruction loss ($\mathcal{L}_{\text{MSE}}$): Measures the mean squared error between $C(f)$ and the raster image $R(s)$ rendered from s among every position $\mathbf{p}$ and layer i, leading to $\mathcal{L}_{\text{MSE}} = \text{MSE}(R(s), C(f)) = \mathbb{E}_{\mathbf{p},i}[(s(\mathbf{p})_i - f(\mathbf{p})_i)^2]$ where $s(\mathbf{p})_i \in \{0, 1\}$ denotes if position $\mathbf{p}$ in layer i of s is colored, and $f(\mathbf{p})_i$ denotes the color intensity m_i at position $\mathbf{p}$ and layer i. We use differentiable

rasterizer DiffVG [13] to obtain $R(s)$.

- Binarization loss ($\mathcal{L}_{\text{bi}}$): To maintain the sharp shapes of SVG, we would like the color intensity m_i to approach the extreme values of 0 or 1. We penalize $B(m_i) = min\{(m_i - 0)^{1.1}, (1 - m_i)^{1.1}\}$ at each position $\mathbf{p}$ and layer i, averaged over all positions and layers for f, leading to $\mathcal{L}_{\text{bi}} = \mathbb{E}_{\mathbf{p},i}[B(f(\mathbf{p})_i)]$.
- Regularization loss ($\mathcal{L}_{\text{reg}}$): Applied to network parameters to prevent gradient explosion with $\mathcal{L}_{\text{reg}} = L_2(f)$.

For reconstruction, we aim to optimize f to obtain $f^* = \arg\min_f \mathcal{L}_{\text{rec}}$ with

$$\mathcal{L}_{\text{rec}} = \mathcal{L}_{\text{MSE}} + \lambda_1 \mathcal{L}_{\text{bi}} + \lambda_2 \mathcal{L}_{\text{reg}}, \tag{1}$$

where λ_1 and λ_2 are weight parameters.

B. Coarse Animation Generation

Video score distillation sampling (VSDS) originates from score distillation sampling (SDS), which was first used to generate 3D models based on a text-to-image (T2I) diffusion model [8], [11], [14] and can be generalized to SVG [10], [15]. The core idea is to extract prior information from the T2I diffusion model by comparing the difference between the predicted noise and the added noise. Later, researchers extended T2I to T2V, proposing VSDS for video generation [6], [7], [16]. However, the large domain gap between SVG and diffusion models makes it extremely hard to directly optimize. Thus, our key idea is to use LINR to bridge the gap.

We replicate f^* K times to get $\mathbf{f} = (f_0, f_1, \ldots, f_{K-1})$ which constructs a K-frame video $C(\mathbf{f})$. Next, we optimize $\mathbf{f}$ to produce a coarse LINR animation $\mathbf{f}^*$ with:

- VSDS loss ($\mathcal{L}_{\text{VSDS}}$): To ensure visual stability, we propose a new *stability extension* method to extend $C(\mathbf{f})$ with a fixed frame of the original image at the beginning, resulting in a $(K+1)$-frame video $V(\mathbf{f}) = \text{concat}(R(s), C(\mathbf{f}))$. This fixed frame serves as an anchor to constrain the appearance of the remaining frames, which effectively ensures the visual stability. We then add noise ϵ to these frames at time step t to get $V'(\mathbf{f}) = \alpha_t V(\mathbf{f}) + \sigma_t \epsilon$ where α_t and σ_t are pre-defined parameters depending on t and use T2V model ϵ_ϕ with text prompt c to predict noise $\epsilon_\phi(V'(\mathbf{f}), t, c)$. The VSDS loss is the prediction error: $\mathcal{L}_{\text{VSDS}} = \mathbb{E}_t[w(t)(\epsilon_\phi(V'(\mathbf{f}), t, c) - \epsilon)]$ where $w(t)$ is a constant depending on α_t.
- Binarization loss ($\mathcal{L}_{\text{bi}}$): During the animation process, maintaining sharp edges of shapes is still important. We adopt binarization loss in Eq. (1) by taking average among K frames.
- Regularization loss ($\mathcal{L}_{\text{reg}}$): Applied to network parameters to prevent gradient explosion. We adopt regularization loss in Eq. (1) by taking average among K frames.

For animation, our goal is to find $\mathbf{f}^* = \arg\min_{\mathbf{f}} \mathcal{L}_{\text{ani}}$ with

$$\mathcal{L}_{\text{ani}} = \mathcal{L}_{\text{VSDS}} + \lambda_3 \mathcal{L}_{\text{bi}} + \lambda_4 \mathcal{L}_{\text{reg}}, \tag{2}$$

where λ_3 and λ_4 are weight parameters, respectively.

C. Animation Refinement

After obtaining the LINR animation $\mathbf{f}^*$, we warp the original SVG parameters to match $C(\mathbf{f}^*)$. We first calculate the optical flows from $R(s)$ to each frame in $C(\mathbf{f}^*)$ using Farneback method [17]. Then we apply the optical flows to every point parameters in s to produce a smooth SVG-based animation $\mathbf{s}$.

For scenarios that require more flexibility and freedom, such as multi-layer ones, we provide an alternative approach: directly optimizing the point parameters of s using MSE loss with each frame of LINR animation $C(\mathbf{f}^*)$ to produce $\mathbf{s}$.

III. Experiments

A. Experiment Setup

The SIREN network [12] has 4 layers with a hidden dimension of 64 and is activated by a clip-sigmoid function where $y = \text{clamp}_{[0,1]}\left((\text{sigmoid}(x) - 0.5) \times (1 + 10^{-7}) + 0.5\right)$ to prevent gradient explosion since a sigmoid function is not able to approach the extreme values of 0 or 1. The rendered image size is 256×256 pixels.

During **Vector Graphics Reconstruction**, λ_1 increases with optimization steps, linearly from 0 to reach a maximum of 2×10^0 since in the first half of the reconstruction, we aim for overall consistency, while in the second half, we expect clear edges. λ_2 is set to 1×10^{-6}. The Adam optimizer [18] is used with a learning rate of 1×10^{-3} and runs for 10,000 iterations.

For **Coarse Animation Generation**, we render a 12-frame video and the T2V model used is ModelScope text-to-video [2], with timestep t randomly chosen from 200 to 400 every iteration. λ_3 is set to 2×10^2, and λ_4 is set to 1×10^{-6}. The Adam optimizer is used with a learning rate of 3×10^{-5} and runs for 8,000 iterations.

B. Metrics

We collected approximately 20 SVG vector graphics, animated them, and measured the animation results in terms of **Appearance Consistency** and **Motion-Prompt Alignment**. The results are shown in Table I. For **Appearance Consistency**, we measured the CLIP [19] cosine similarity between the generated frames and the original image. For **Motion-Prompt Alignment**, we measured the X-CLIP [20] similarity score between the generated video and the input text prompt.

C. Comparison with State-of-the-Art Methods

Given the scarcity of automated vector graphic animation methods, we compare our approach with the most realted two methods. Example results are shown in Fig. 2.

- Comparison with LiveSketch [6]: LiveSketch is designed for animating sketch SVGs. It focuses on global and local optimization by inputting the SVG's parameter points into an MLP to obtain optimized values, which are then animated using VSDS. However, this direct parameter point optimization leads to problems like shape alterations, unsmooth lines, and excessive freedom. The results are shown in Fig. 2(a).

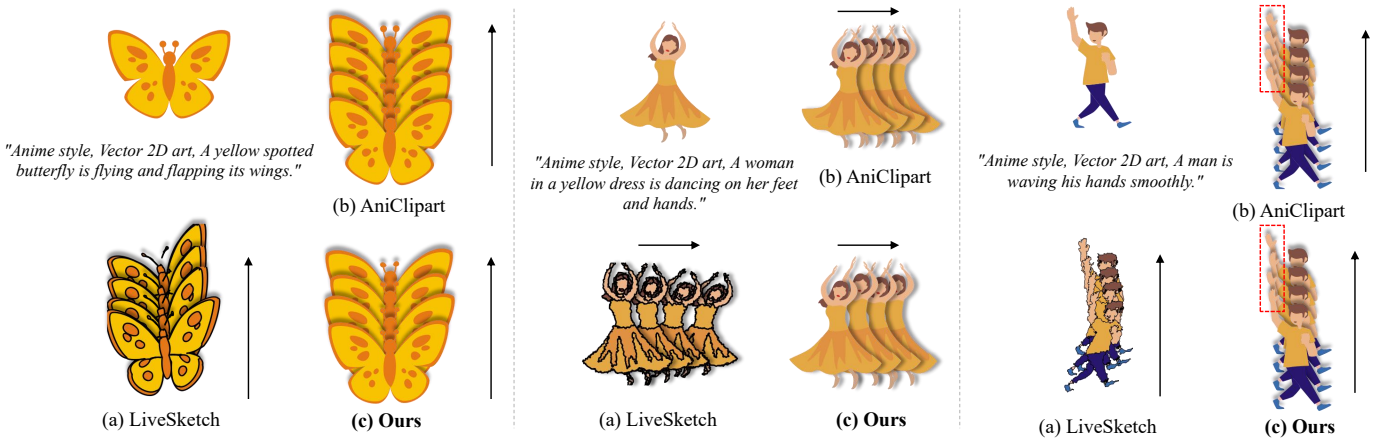

Fig. 2. The results of *LINR Bridge* and comparisons to other methods. We show consecutive frames of SVG animation. (a) LiveSketch [6] optimizes global transforms and local point parameters, unsuitable for common vector graphics other than sketches. (b) AniClipart [7] optimizes an ARAP deformation on vector graphics and offers excessively strict restrictions on shapes. (c) Our method can generate smooth, highly flexible, and well-structured animation results. Please see full animation in our project page: https://gaowenshuo.github.io/LINR-bridge/.

- Comparison with AniClipart [7]: AniClipart extracts key points and skeletons from SVGs, performs As-Rigid-As-Possible (ARAP) deformations based on these key points, and animates using VSDS. Key point extraction methods vary by SVG type (*e.g.*, human skeletons for human SVGs), and the number of keypoints is crucial. SVG motion is constrained by skeleton movements, leading to limited flexibility and excessive joint motion. Manual layering is required for layered motion, otherwise different layers would always stick together. The results are shown in Fig. 2(b).

In contrast, our method animates within the LINR domain, ensuring better consistency in color and shape, while providing greater smoothness, freedom of motion, and separation over layers. The results are shown in Fig. 2(c).

TABLE I
COMPARISONS TO LIVESKETCH AND ANICLIPART

Method	Appearance Consistency ↑	Motion-Prompt Alignment ↑
LiveSketch	0.8826	19.2335
AniClipart	0.9595	19.5870
Ours	**0.9727**	**21.2927**

D. Ablation Study

- Layered INR: The layered structure preserves each layer's shape simplicity and color consistency. If INR outputs (R, G, B) values, significant color deviations and shape changes may occur during VSDS. See Fig. 3(a).
- Binarization Constraint: The binarization loss ensures clear edges in each layer of the LINR-generated image. Without this loss, LINR may produce blurry images and excessive shape variations during animation. See Fig. 3(b).
- Stability Extension: The frame extended maintains the appearance consistency between the optimized LINR and the original SVG. Without it, animation might gradually deviate from the shape of the original during continuous optimization. See Fig. 3(c).

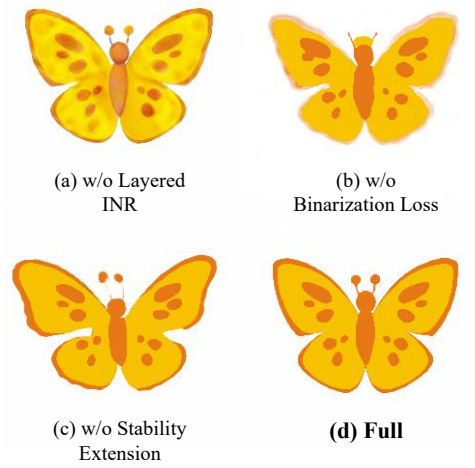

Fig. 3. Ablation study. We show one frame from coarse LINR animations. (a) Without the layered structure of INR, the colors will become uncontrollable, leading to significant color deviations. (b) Without binarization loss, the rendered image will lack clear boundaries, resulting in blurred transition areas. (c) Without the stability extension, the content of the image will not be constrained, leading to a gradual deviation in appearance from the original.

IV. CONCLUSION

In this paper, we propose an automated method for vector graphics animation using LINR and VSDS. Our approach first reconstructs the input SVG using LINR and then generates animations by leveraging prior knowledge from T2V diffusion models through VSDS. Our method maintains consistency in shape and color while offering a high degree of freedom in motion, resulting in stable and smooth animations. However, our method has certain limitations. Due to the complexity and unpredictability of the INR neural network, reconstruction and animation may occasionally produce unexpected and uncontrollable results, particularly with complex SVGs.

REFERENCES

[1] O. Andersson, R. Berjon, E. Dahlström, A. Emmons, J. Ferraiolo, A. Grasso, V. Hardy, S. Hayman, D. J. W3C, C. L. W3C *et al.*, "Scalable vector graphics (svg) tiny 1.2 specification," *World Wide Web Consortium (W3C) recommendation*, vol. 22, 2008.

[2] J. Wang, H. Yuan, D. Chen, Y. Zhang, X. Wang, and S. Zhang, "Modelscope text-to-video technical report," *arXiv preprint arXiv:2308.06571*, 2023.

[3] H. Chen, Y. Zhang, X. Cun, M. Xia, X. Wang, C. Weng, and Y. Shan, "Videocrafter2: Overcoming data limitations for high-quality video diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 7310–7320.

[4] J. Xing, M. Xia, Y. Zhang, H. Chen, X. Wang, T.-T. Wong, and Y. Shan, "Dynamicrafter: Animating open-domain images with video diffusion priors," *arXiv preprint arXiv:2310.12190*, 2023.

[5] S. Zhang, J. Wang, Y. Zhang, K. Zhao, H. Yuan, Z. Qin, X. Wang, D. Zhao, and J. Zhou, "I2vgen-xl: High-quality image-to-video synthesis via cascaded diffusion models," *arXiv preprint arXiv:2311.04145*, 2023.

[6] R. Gal, Y. Vinker, Y. Alaluf, A. Bermano, D. Cohen-Or, A. Shamir, and G. Chechik, "Breathing life into sketches using text-to-video priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 4325–4336.

[7] R. Wu, W. Su, K. Ma, and J. Liao, "Aniclipart: Clipart animation with text-to-video priors," *arXiv preprint arXiv:2404.12347*, 2024.

[8] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," *arXiv preprint arXiv:2209.14988*, 2022.

[9] Y. Chen, S. Liu, and X. Wang, "Learning continuous image representation with local implicit image function," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 8628–8638.

[10] V. Thamizharasan, D. Liu, M. Fisher, N. Zhao, E. Kalogerakis, and M. Lukac, "Nivel: Neural implicit vector layers for text-to-vector generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 4589–4597.

[11] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[12] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," vol. 33, pp. 7462–7473, 2020.

[13] T.-M. Li, M. Lukáč, M. Gharbi, and J. Ragan-Kelley, "Differentiable vector graphics rasterization for editing and learning," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.

[14] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[15] S. Iluz, Y. Vinker, A. Hertz, D. Berio, D. Cohen-Or, and A. Shamir, "Word-as-image for semantic typography," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–11, 2023.

[16] Z. Liu, Y. Meng, H. Ouyang, Y. Yu, B. Zhao, D. Cohen-Or, and H. Qu, "Dynamic typography: Bringing text to life via video diffusion prior," *arXiv e-prints*, pp. arXiv–2404, 2024.

[17] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13*. Springer, 2003, pp. 363–370.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *Proc. IEEE Int'l Conf. Machine Learning*. PMLR, 2021, pp. 8748–8763.

[20] B. Ni, H. Peng, M. Chen, S. Zhang, G. Meng, J. Fu, S. Xiang, and H. Ling, "Expanding language-image pretrained models for general video recognition," in *European Conference on Computer Vision*. Springer, 2022, pp. 1–18.