Image Vectorization Editing and Optimization via Differentiable Rendering

CS4316

Shanghai Jiao Tong University

Tianyu Mao
523030910086

Abstract

This project explores the domain of vector graphics generation and editing, addressing the scarcity of high-quality training data for downstream AI models. Leveraging Differentiable Vector Graphics (DiffVG), we establish an automated pipeline that transforms raster images into editable Scalable Vector Graphics (SVG). Our work overcomes critical limitations in existing methods, specifically the "masking occlusion" problem and the "over-pruning" of delicate structures. We introduce *Smart Alpha* for dynamic transparency optimization and a multi-metric *Smart Pruning* mechanism that combines color consistency, geometric compactness, and leave-one-out loss analysis. Experimental results demonstrate that our system achieves superior perceptual fidelity and topological cleanliness compared to baseline optimization methods, effectively handling complex tasks such as style transfer and precise object deletion.

January 1, 2026

Contents

1 Introduction

1.1 Background and Motivation

In the rapidly evolving field of AI-Generated Content (AIGC), raster image generation has reached unprecedented levels of quality. However, vector graphics (SVG) generation remains a significant bottleneck. Unlike raster images, which are grids of pixels, SVGs are composed of mathematical paths defined by control points, making them resolution-independent and easily editable. These properties make SVGs indispensable in graphic design, engineering, and typography.

The primary challenge in learning-based SVG generation is the lack of large-scale, high-quality datasets containing paired "source SVG" and "edited SVG" examples. To address this, we aim to build a robust pipeline that can automatically generate such data by optimizing an input SVG to match a target raster image.

1.2 Objective

The objective of this project is to bridge the gap between pixel-based reference images and vector-based SVG representations. We focus on two core editing tasks:

1. **Color Migration (Basic Task):** Transferring the color scheme of a reference image to a source SVG while strictly preserving its original topological structure.
2. **Topology Editing (Advanced Task):** Modifying the SVG structure (adding or removing elements) to match the semantic content of the reference image.

2 Methodology

Our approach is grounded in **DiffVG (Differentiable Vector Graphics)**, a differentiable rendering engine. Let P be the set of SVG parameters (control points, colors, opacity). The renderer R maps these parameters to a raster image $I = R(P)$. The goal is to find optimal parameters P^* that minimize the difference between the rendered image and the target image I_{target}:

$$P^* = \arg\min_P \mathcal{L}(R(P), I_{target}) \tag{1}$$

where $\mathcal{L}$ is a loss function.

2.1 System Architecture

We established a Dockerized environment to ensure reproducibility, resolving dependencies between PyTorch, DiffVG, and CUDA. The pipeline consists of three stages: Preprocessing, Color Optimization, and Topology Pruning.

2.2 Data Preprocessing: The "Clean + Thicken" Strategy

Raw SVGs from the wild are often ill-conditioned for optimization. We implemented ‘clean_svg.py‘ to normalize the data:

- **Coordinate Absolute-ization:** Relative coordinates ('m' commands) accumulate errors during gradient updates. We convert all paths to absolute coordinates ('M' commands).
- **Compound Path Explosion:** Complex paths containing multiple sub-paths are "exploded" into individual elements. This allows the optimizer to assign different colors or opacity values to disjoint parts of the same object.
- **Tactical Thickening:** Thin strokes often vanish because the gradient for "deleting" a misaligned line is often steeper than "moving" it. We pre-thicken strokes to increase their overlap with the target, creating a larger "gradient basin" for successful convergence.

2.3 Color Refinement: Smart Alpha

A major issue identified in early experiments was the "masking effect," where opaque layers obscured underlying details. Standard SVG optimization often locks opacity (α) to 1.0.

We proposed the **Smart Alpha** strategy, treating α as a learnable parameter constrained to $[0, 1]$.

$$C_{rendered} = \alpha \cdot C_{foreground} + (1 - \alpha) \cdot C_{background} \tag{2}$$

By optimizing α, the system automatically identifies occluding layers that contribute to high loss and reduces their opacity, effectively "fading them out" to reveal the correct structure.

2.4 Topology Editing: Smart Pruning

For the task of removing unwanted elements (e.g., removing a fish's mouth while keeping the eyes), naive loss-based pruning often fails, leading to "over-pruning" (removing useful details like eyes or fins).

We developed a multi-metric scoring system $S(path)$ to determine if a path should be pruned:

$$S(path) = w_1 \cdot S_{color} + w_2 \cdot S_{geo} + w_3 \cdot S_{loss} \tag{3}$$

- **Color Consistency (S_{color}):** We sample points along the path and compare their color in the rendered image vs. the target image. High discrepancy implies the path does not belong.
- **Geometric Compactness (S_{geo}):** Defined as the isoperimetric quotient $\frac{4\pi \cdot Area}{Perimeter^2}$. Regular shapes (like eyes) receive high scores and are protected.
- **Leave-One-Out Loss (S_{loss}):** We tentatively hide the path and measure $\Delta Loss$. If $Loss_{new} < Loss_{old}$, the path is harmful and should be removed.

3 Experiments and Results

3.1 Implementation Details

All experiments were conducted on an NVIDIA GPU with CUDA acceleration. The optimization ran for 200 iterations using the Adam optimizer with a learning rate of 1.0 for positions and 0.01 for colors.

3.2 Qualitative Evaluation

We evaluated our pipeline on the provided test cases. Figure 1 (placeholder) illustrates the progression from the source SVG to the final optimized result.

- **Case 1 (Pen):** Successfully transferred the solid blue color while maintaining the outline.
- **Case 3 (Bulb):** The "Clean + Thicken" strategy successfully reconstructed the filament structure which was previously lost.
- **Case 2 (Fish):** The Smart Pruning mechanism precisely removed the mouth (black shape) while retaining the eyes and fins, which were lost in baseline approaches.

Figure 1: Qualitative results. Top: Original SVGs. Middle: Target PNGs. Bottom: Our Optimized Results.

3.3 Loss Function Analysis: MSE vs. LPIPS

We explored the trade-off between Mean Squared Error (MSE) and Learned Perceptual Image Patch Similarity (LPIPS).

- **MSE:** Tends to prioritize large regions of flat color. This often results in accurate fill colors but washed-out strokes, as strokes occupy few pixels.
- **LPIPS:** Focuses on structural similarity and high-frequency details. Using LPIPS improved stroke retention but occasionally introduced color noise in flat regions.

We conclude that a hybrid loss $\mathcal{L} = \lambda_{MSE}\mathcal{L}_{MSE} + \lambda_{LPIPS}\mathcal{L}_{LPIPS}$ is optimal.

4 Discussion and Limitations

4.1 The Mosaic Effect

In the "Spawning" task (adding new objects), we observed a "Mosaic Effect," where the system generates multiple small, fragmented shapes instead of a single coherent object. This is a fundamental limitation of gradient-descent-based vectorization: the optimizer lacks semantic understanding of "objects." It greedily places primitives to cover error regions, resulting in a patch-work appearance.

4.2 Future Work

To address the Mosaic Effect, we propose integrating **Semantic Priors**. By using a segmentation model (e.g., Segment Anything Model, SAM) to generate an initial mask for the new object, we can initialize the SVG with a coherent shape matching that mask, rather than starting from random initialization.

5 Conclusion

This project successfully demonstrates a differentiable rendering pipeline for SVG editing. Through the introduction of Smart Alpha and Smart Pruning, we resolved key artifacts associated with standard optimization methods. Our system can generate high-quality, topologically clean vector graphics that align with raster targets, providing a valuable tool for automated dataset creation in the field of vector graphics generation.