Optimize & Reduce: A Top-Down Approach for Image Vectorization

Or Hirschorn*, **Amir Jevnisek***, **Shai Avidan**

Tel-Aviv University, Israel
orhirschorn@mail.tau.ac.il, amirjevn@mail.tau.ac.il, avidan@eng.tau.ac.il

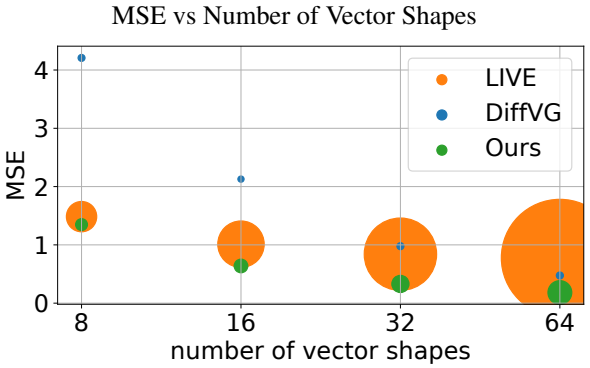
Abstract

Vector image representation is a popular choice when editability and flexibility in resolution are desired. However, most images are only available in raster form, making raster-to-vector image conversion (vectorization) an important task. Classical methods for vectorization are either domain-specific or yield an abundance of shapes which limits editability and interpretability. Learning-based methods, that use differentiable rendering, have revolutionized vectorization, at the cost of poor generalization to out-of-training distribution domains, and optimization-based counterparts are either slow or produce non-editable and redundant shapes. In this work, we propose Optimize & Reduce (O&R), a top-down approach to vectorization that is both fast and domain-agnostic. O&R aims to attain a *compact* representation of input images by iteratively optimizing Bézier curve parameters and significantly reducing the number of shapes, using a devised importance measure. We contribute a benchmark of five datasets comprising images from a broad spectrum of image complexities - from emojis to natural-like images. Through extensive experiments on hundreds of images, we demonstrate that our method is domain agnostic and outperforms existing works in both reconstruction and perceptual quality for a fixed number of shapes. Moreover, we show that our algorithm is $\times 10$ faster than the state-of-the-art optimization-based method. Our code is publicly available: https://github.com/ajevnisek/optimize-and-reduce

Introduction

Vectorization refers to the process of transforming a raster image into a vector image. While it is possible to create a vector image with a shape for each pixel, ensuring a perfect reconstruction, this approach lacks compactness and editability. The question arises: can we achieve compact vectorization without compromising quality?

This paper aims to achieve vectorization while minimizing shape count. We show in Figure 1 the tradeoff between shape count and Mean Squared Error (MSE) for our method and two comparative techniques. Our approach demonstrates better performance compared to the state-of-the-art LIVE method (Ma et al. 2022) in both MSE and runtime aspects. Furthermore, our method surpasses DiffVG (Li et al. 2020) in terms of minimizing MSE.



- Runtime indicated by marker-size (↓ better)

Figure 1: Optimize & Reduce: A fast Top-Down approach for raster to vector image conversion, aiming at a low budget of shapes for vectorization. Our method (green circles) achieves lowest MSE error (y-axis) for all shape counts (x-axis) with a low runtime cost (size of circle).

Vector images are an appealing image representation for those seeking infinite resolution and editability. Vector Images are made up of lines and shapes with clear mathematical formulations. A graphical renderer processes the mathematical formulations and renders the image at any specified resolution. This resolution-free property makes them suitable for cases where the image is presented in multiple resolutions, such as commercial logos. A single vector image can be resized to fit the resolution of a business card, a header image in a formal letter, a T-shirt, or a billboard.

The task of vectorization has been studied extensively for many decades (Jimenez and Navalon 1982). We identify two primary drawbacks of classical algorithms for vectorization. First, they are often tailored to a specific domain of images such as fonts (Itoh and Ohno 1993), line drawings (Chiang, Tue, and Leu 1998; Noris et al. 2013), cartoons (Zhang et al. 2009) or cliparts (Favreau, Lafarge, and Bousseau 2017; Dominici et al. 2020). Second, they yield an abundance of shapes making the representation complex. For instance, Potrace (Selinger 2003) generates thousands of

*These authors contributed equally.

(a) Raster Image

(b) AutoTracer 4579 shapes

(c) **Ours** 32 shapes

(d) **Ours** 2 shapes color edit

Figure 2: Low Shapes Budget Vectorization: Low budget shapes makes editing easier.

shapes for natural-like images.

In contrast, our objective is to achieve a compact vector representation, characterized by a relatively small number of shapes. Intuitively, a low number of shapes enhances the editability of vector images. To demonstrate this we show a comparison between the vectorization of the classical AutoTrace (Weber 2004) and our vectorization approach in Figure 2. Performing a color edit to the shirt is straightforward when the number of shapes is low (32 shapes), whereas, with AutoTrace the task becomes challenging due to the need to augment hundreds of shapes.

Differentiable rasterizers, such as DiffVG (Li et al. 2020), marked a turning point in the field. DiffVG can optimize the vector shape parameters with respect to any raster-based loss function. DiffVG suggested a vanilla vectorization method, directly optimizing the shape parameters. This was extended to two types of vectorization methods: dataset-based and optimization-based approaches. Dataset-based approaches train a neural network to regress the shape parameters and train over some datasets. These methods leverage DiffVG propagation of gradients from the loss function to the shape parameters. Though these methods enjoy a fast inference time, they generalize poorly to out-of-dataset image distribution.

Our focus in this work is to extend Optimization-based methods and provide a method that excels when vectorization runtime and shape compactness are important. While DiffVG (Li et al. 2020) proposes direct optimization for curve parameters, this basic approach is highly sensitive to initialization and tends to generate intricate shapes when optimized using reconstruction or perceptual loss functions. The current State-of-the-art vectorization method in terms of reconstruction error is Layer-wise Image Vectorization (LIVE) (Ma et al. 2022), which employs a bottom-up process to generate a comprehensive set of vectors, achieving layerwise decomposition. However, this advantage comes at the expense of increased runtime, as depicted in Figure1.

Our approach, Optimize & Reduce (O&R), is a top-down method. Given a raster image, we cluster pixel values to create an initial guess of the shapes. Then, in the *Optimize* step, we use DiffVG to optimize all shapes. We use both reconstruction loss, to match the input raster image, as well as a geometric loss to encourage DiffVG to generate "simple" shapes that do not intersect. In the *Reduce* step, we remove shapes according to an importance measure that we design. The remaining shapes serve as the initial guess for the next round of optimization. The process is iterated until the target number of shapes is reached. The combination of a smart, clustering-based initialization and our O&R scheme allows DiffVG to optimize for all shapes *at once* leading to excellent results.

Through extensive experimentation, we demonstrate that our approach achieves a runtime speedup of over $\times 10$ compared to LIVE, all while maintaining reconstruction accuracy. Furthermore, we establish that DiffVG produces reconstructions that are inferior to our method, whether initialized randomly (as shown in Figure 1) or with a more informed approach (as illustrated in Figure 8).

We contribute a benchmark of five datasets covering a wide range of image complexities. Three of these datasets consist of images that are typically of interest for vectorization, such as (two versions of) emojis, NFT-apes, and SVG images converted to raster images. The last dataset we propose consists of natural-like images generated by Midjourney. Natural images have also been shown to be of interest for vectorization in previous works (Li et al. 2020; Ma et al. 2022). Thus, our paper's contribution is threefold: (1) we propose a new vectorization method, (2) through extensive experiments, we demonstrate that our method produces high-quality vector images in terms of reconstruction and perceptual quality, and (3) we provide a benchmark consisting of five diverse image types.

Related Work

Raster To Vector Conversion The conventional image vectorization process typically involves two main steps. Initially, pixels are clustered into regions, either through segmentation or triangulation based on edge detection. Subsequently, these regions are fitted using vector primitives. The optimization in the second step includes both discrete and continuous variables, such as the number and position of primitives. Various exploration mechanisms, like those presented in (Li, Lafarge, and Marlet 2020; Favreau, Lafarge, and Bousseau 2017; Olsen and Gooch 2011) have been proposed for this purpose. Among them, Potrace(Selinger 2003) and Diebel *et al.*(Diebel 2008) utilized an empirical two-stage algorithm to regress segmented components as polygons and bezigons.

In the deep learning era, large, pre-trained models can be employed for pixel clustering, utilizing techniques like semantic segmentation. Moreover, perceptual loss based on deep neural networks can evaluate fitness between image regions and vector primitives. One can categorize vectorization techniques into Dataset-based and Optimization-based.

Some works focus on dataset-level learning, which simplifies the problem to one specific domain of images. Early works used CNNs (Zheng, Jiang, and Huang 2019) or GANs (Balasubramanian, Balasubramanian et al. 2019) to learn a direct conversion between the two representations. Later, Lopes *et al.*(Lopes et al. 2019) and Carlier *et al.*(Carlier et al. 2020) framed the synthesis of vectors as a problem of predicting sequences, whereby the image is represented as a sequence of instructions for drawing, emulating the representation of vector graphics in typical formats.

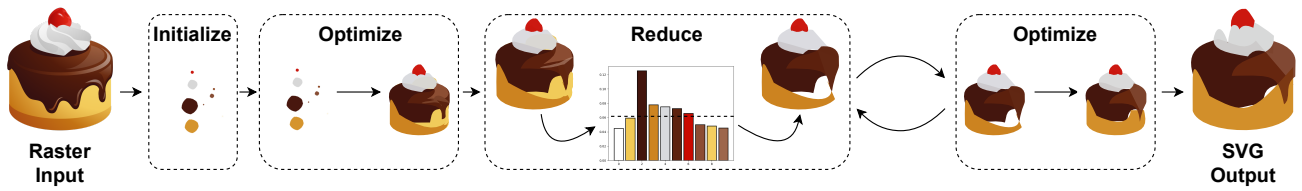


Figure 3: Optimize & Reduce: We start with a raster image and use DBSCAN (Ester et al. 1996) and connected components to initialize a large number of Bézier shapes. We **Optimize** the shapes using DiffVG (Li et al. 2020). Then we **Reduce** the number of shapes by ranking them according to $\mathcal{L}_{reduce}$. That is, the least important k shapes (shapes with scores below the dashed line in the bar graph) are removed. The remaining shapes serve as an initial guess for the next optimization iteration.

Similarly, other works formulate this task as an image-to-image translation using paired data of corresponding images and shape vectors (Yi et al. 2019; Li et al. 2019b,a; Muhammad et al. 2018), which are difficult to acquire. To overcome this supervision problem, there is a popular research trend of integrating differentiable rendering methods (Li et al. 2020; Mihai and Hare 2021) with deep learning models. Reddy *et al.*(Reddy et al. 2021a) suggested a dataset-level method, which utilized differentiable rendering (Li et al. 2020). They used a VAE to encode the input image, and using an RNN network create latent geometric features, which are decoded using a Bézier path decoder and a differentiable rasterizer. However, this method is limited to only a specific domain and performs poorly on complex images. Recently, Jain *et al.*(Jain, Xie, and Abbeel 2022) combined a pre-trained text-to-image diffusion model and a differentiable renderer to create a text-to-vector model. Yet, the topology of the output SVG isn't intuitive, as simple shapes are sometimes represented by multiple overlapping layers.

Meanwhile, optimization-based methods were also suggested, not relying on datasets or lengthy training phases. Vinker *et al.*(Vinker et al. 2022b,a) proposed using semantic Clip objective to optimize Bézier curves for object/scene sketching. Ma *et al.*(Ma et al. 2022) presented a layer-wise image vectorization method, which converts an image to vectors while maintaining its topology, forming shapes that are semantically consistent with the human perspective. They use a differentiable renderer (Li et al. 2020) in a bottom-up approach, optimizing the vector image layer-by-layer. However, this method is extremely slow, as optimizing each layer is compute-intensive. In addition, the output greatly varies according to the initialization method of the shapes, thus becoming a strong prior which fails for complex and texture-rich images. In contrast, our top-down approach is fast, robust to initialization, and can handle both simple and complex images while maintaining editability.

Topology and Editability To ensure easy SVG editing, proper organization, and smooth shapes are crucial. Previous studies have explored image topology for both raster images and vector shapes (Guo et al. 2019; Zhu et al. 2022; Kopf and Lischinski 2011).

Some works study topology, to characterize fonts styles (Wang and Lian 2021; Reddy et al. 2021b; Shimoda et al. 2021), while others use topology as a regularizer for vectorizing sketches (Bessmeltsev and Solomon 2019; Chan, Durand, and Isola 2022; Mo et al. 2021). For images, early works focused on vectorizing cliparts using various boundary-aware methods (Hoshyari et al. 2018; Dominici et al. 2020; Riso, Sforza, and Pellacini 2022). Later, Favreau *et al.*(Favreau, Lafarge, and Bousseau 2017) and Shen *et al.*(Shen and Chen 2021) decomposed bitmap pictures into gradient-filled vectors, creating visual hierarchy through segmentation and merging. Alternative studies like diffusion curves (Xie et al. 2014; Zhao, Durand, and Zheng 2017; Orzan et al. 2008) and meshes (Sun et al. 2007; Xia, Liao, and Yu 2009; Lai, Hu, and Martin 2009) were explored but fell short for complex scenes.

Reddy *et al.*(Reddy et al. 2021a) proposed simultaneous image vectorization and topology extraction but struggled with shape ordering and domain specificity. Later, Ma *et al.*(Ma et al. 2022) introduced a bottom-up SVG construction with geometric constraints, excelling for basic clipart but struggling with complex images. Our method does not require any pre-segmentation or model training and exhibits a satisfying ability to explore image topologies and create easily editable SVGs, even for complex images.

Image Abstraction The process of abstraction is central to creating a new image. To effectively convey desired details, artists must choose which essential visual characteristics of the subject to depict, and which to exclude (often many). Abstract images maintain semantic essence, exhibiting simplicity with fewer details and textures.

Prior works (Gao et al. 2019; Frans, Soros, and Witkowski 2022; Tian and Ha 2022) used contextual loss for style transfer and vector graphics generation. Li *et al.*(Li et al. 2020) demonstrated the usage of a differentiable renderer with a perceptual loss, to convert images into abstract SVGs. Later, Liu *et al.*(Liu et al. 2021) and Zou *et al.*(Zou et al. 2021) used a differential strokes renderer to produce a series of strokes for a given image, creating abstract paintings of the input image. Vinker *et al.*(Vinker et al. 2022b,a) proposed a semantic CLIP loss for image-to-sketch translation, enhancing semantic fidelity while retaining morphology. ClipVG (Song et al. 2023) used differentiable rendering and CLIP-based optimization to edit images using vector graphics instead of pixels. We follow (Vinker et al. 2022a), demonstrating different levels of abstraction for vectorization tasks, by varying the number of shapes in the output image and by incorporating the suggested CLIP loss as a reconstruction objective.

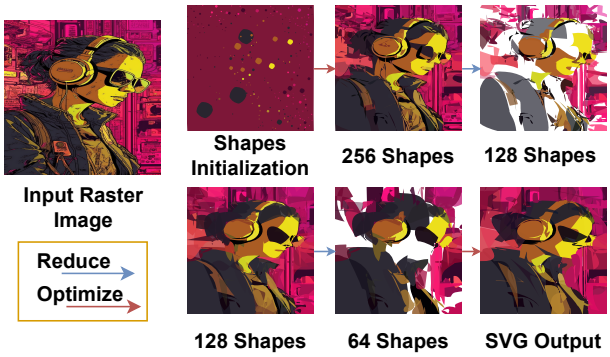


Figure 4: Algorithm Steps: We initialize bézier curves sampled uniformly on a circle with a radius inversely proportional to the DBSCAN clusters. We then iterate between optimization and reduction steps for the following schedule: $256 \rightarrow 128 \rightarrow 64$. The output SVG reconstructs the raster image with a small number of vector shapes.

Method: Optimize & Reduce

Optimize & Reduce (O&R) is a top-down approach to vectorization that is based on the DiffVG (Li et al. 2020) differentiable renderer. Figures 3 and 4 shows the core steps of our method. We start with a raster image, initialize the shapes and then iterate between Optimize and Reduce. For this scheme to work, we need to define how to initialize the shape parameters, how to determine which shapes to remove, and what loss function to use for DiffVG.

Initialization We found that DiffVG is sensitive to the Bézier curves initialization. Therefore, instead of initializing it with a random set of shape parameters, we use DBSCAN (Ester et al. 1996) to cluster image colors and divide the image into regions that correspond to connected components of each cluster. Then, we sort the connected components according to their pixel area in the source image. The N largest connected components' center of mass serves as anchors for the differentiable rasterizer. The first optimization step of our method starts with circle-like shapes with a diameter proportional to the connected component area.

Optimize The Optimize step runs DiffVG with the initial guess provided either by the initialization step or the Reduce step (described next). DiffVG takes as input a raster image as well the initial guess of N shapes and optimizes the parameters of all the shapes *at once* to minimize a loss function between the resulting vector image and input raster image.

DiffVG is flexible with the choice of the optimization loss function $\mathcal{L}_{\text{optimize}}$, and we evaluate a number of loss functions. These can be any classical reconstruction pixel-wise loss, such as L_1 or L_2 (MSE) loss, as well as a contextualized loss, such as CLIP loss (Vinker et al. 2022b). This loss uses different intermediate layers of CLIP-ViT (Dosovitskiy et al. 2021) where shallow layers preserve the geometry of the image and deeper layers preserve semantics. We also devise a new geometric loss that regularizes the behavior of the Bézier curves, explained later in this paper. Combining all losses results in our general loss term:

$$\mathcal{L}_{optimize} = \mathcal{L}_{reconstruction} + \lambda_{geometric}\mathcal{L}_{geometric} \quad (1)$$

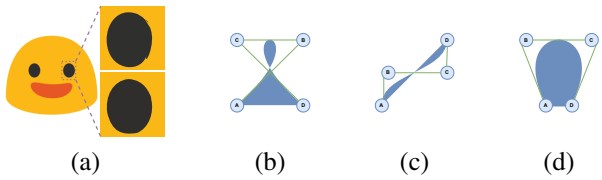


Figure 5: Geometric Loss: Our geometric constraint prevents curve intersections, resulting in the smoother topology of shapes. (a) Input image and SVG output without/with (Top/Bottom) geometric regularization. Our geometric loss term has three components, each one aiming to eliminate intersection cases: (b) Self-intersection occurs when AB intersects CD. In addition, Cross-intersections are prone to happen when: (c) $[A, B, C]$ and $[B, C, D]$ have different orientations. (d) $\angle ABC$ or $\angle BCD$ are acute.

Reduce A shape P is defined via a set of Bézier control points p^k and an RGBA vector c^i. An N-shaped vector image is a set of N shapes: $\left\{P_i = \{p_i^k, c_i^j\}\right\}_{i=1}^{N}$. For each shape P_i, the ranking loss measures the degradation caused by not including that specific shape when the image is rendered. The loss is measured against the original raster image $\mathcal{I}$. Therefore, the loss for the i-th shape, includes rendering all shapes but the i-th shape:

$$\hat{I} = \text{render}(\text{P} \setminus \text{P}_\text{i}) \quad (2)$$

$$\text{rank-score[i]} = \mathcal{L}_{\text{reduce}}(I, \hat{I}) \quad (3)$$

$\mathcal{L}_{\text{reduce}}$ can be any classical reconstruction pixel-wise loss, such as L_1 or MSE loss, a contextualized loss, or a non-differentiable loss, such as histogram matching loss. After scoring all shapes, the lower-ranked half of the shapes are removed from the vector representation of the image. The parameters of the remaining shapes serve as the initial guess for the *Optimization* step.

Geometric Loss Term Another contribution of our paper is a geometric loss term that encourages simple shapes through a differentiable penalty for intersections. An example in Figure 5 shows the effectiveness of the proposed geometric loss in mitigating intersection problems. The constraints on the control points prevent the circle shape from intersecting, demonstrating the intuitive and simple, yet highly effective nature of the proposed objective function.

The mathematical formulation is as follows: assume the control points of a cubic Bézier path are (in order): A, B, C, and D. There are two types of intersections: self-intersections, where a single Bézier curve intersects itself, and cross-intersections, where two Bézier curves intersect each other within the same shape. We observed intersections in three scenarios: (a) Self-intersection exists if the lines $\vec{AB}$ and $\vec{CD}$ intersect (Figure 5b). Furthermore, cross-intersections *are prone to happen* when: (b) ABC and

(a) (b) (c) (d) (e)

Figure 6: Dataset Visualization: We evaluate our algorithm on five datasets: (a) former version of EMOJI (Google 2022) (b) current version of EMOJI (Google 2022) (c) images gathered from free-svg website (d) images from the famous bored apes NFTs (e) images generated using a text-to-image model (Midjourney).

BCD have different orientations (Figure 5c), (c) $\angle ABC$ or $\angle BCD$ are acute (Figure 5d).

To measure self-intersection or orientation change, we compute the orientation (0 is clockwise and 1 is counter-clockwise) of 3 points $[A, B, C]$, where $S(x)$ represents a sigmoid function:

$$O(A,B,C) = S((B_y - A_y)\cdot(C_x - B_x) - (B_x - A_x)\cdot(C_y - B_y)) \quad (4)$$

Using this approximation and simple AND and XOR logic, we can verify the same orientation for $[A, B, C]$ and $[B, C, D]$, or check whether AB intersects CD ($f_{intersect}$ and $f_{orientation}$ defined and detailed in the supplementary). We integrate these functions into our geometric loss:

$$\mathcal{L}_{geom_{ab}} = \lambda_p \cdot (f_{intersect} + f_{orientation}) \quad (5)$$

In addition, The angle loss can be computed using a dot product between the line segments:

$$\mathcal{L}_{geom_c} = ReLU(-\frac{AB \cdot BC}{|AB| \cdot |BC|}) + ReLU(-\frac{BC \cdot CD}{|BC| \cdot |CD|}) \quad (6)$$

Our final geometric loss is:

$$\mathcal{L}_{geometric} = \mathcal{L}_{geom_{ab}} + \mathcal{L}_{geom_c} \quad (7)$$

LIVE (Ma et al. 2022) suggests a different geometric loss, that encourages the angle between the first $\vec{AB}$ and the last $\vec{CD}$ control points connections to be greater than $180°$. Though this rule encourages shapes to be convex, our loss term also avoids both self-intersection and orientation changes.

Experiments

We introduce five new datasets and compare our method against prior art. O&R is a single-image optimization method, therefore we compare it with single-image methods: DiffVG (Li et al. 2020) and LIVE (Ma et al. 2022). We then demonstrate applications based on our method.

Datasets

For evaluation, we collect five datasets to cover a range of image complexities, including emojis, cliparts, and natural-like images. Figure 6 displays a sample image from each dataset. All datasets will be made publicly available.

EMOJI dataset. We take two snapshots of the NotoEmoji project (Google 2022). The first, taken in 2015, follows LIVE's EMOJI's dataset. The second from 2022 introduces color gradients and is available in various resolutions. For evaluation, we resize all of the acquired images to a 240×240 resolution. The supplemental includes a list of all emojis' names and a showcase of the two datasets.

Free-SVG. A collection of 65 images, including complex clipart images. Compared to the Emoji dataset, this dataset is more complex and difficult for image vectorization.

NFT-Apes. A collection of 100 images from the famous Bored Apes Yacht Club NFT. It features profile pictures of cartoon apes that are generated by an algorithm. It includes images with various backgrounds and complex shapes.

Text-To-Image Generated Images. To stress-test vectorization algorithms we introduce a dataset of images generated by Midjourney, a Text-To-Image public generator. We gather 100 such images alongside their prompts.

Implementation Details

Our code is written in PyTorch (Paszke et al. 2017). Hyperparameters and baseline implementations are described thoroughly in the supplementary.

We use three Reduce steps that follow an exponential decay schedule (i.e., the number of shapes is halved in each iteration). For the qualitative experiments, we use MSE for optimization and L_1 for reduction. All quantitative evaluations use L_1 with our geometric loss for optimization. For reduction, we use L_1 and Clip on Midjourney, and for the other datasets, we use only L_1. Further details on the runtime analysis are in the supplementary.

Quantitative Evaluation

Figure 1 shows MSE reconstruction error vs the number of vector shapes for the new emoji dataset. Colors in the graph code DiffVG, LIVE, and our method. As can be seen, DiffVG is the fastest but converges to solutions that are not very accurate (especially when the number of shapes is low). LIVE, on the other hand, works well, but takes about 30 minutes[1] on average to vectorize a 64 shape image. O&R strikes a good balance between the two. It is almost two orders of magnitude faster than LIVE and as accurate. The speedup is a result of our exponential scheduler. For n shapes, LIVE optimizes layer-by-layer which results in $\mathcal{O}(n)$ steps, while we optimize $\mathcal{O}(1)$ steps. For precise runtime figures, please refer to the supplementary materials.

Figure 7 displays the MSE reconstruction error plotted against the number of shapes in the vector representation. Across all datasets, our method consistently outperforms DiffVG with random initialization, when considering the same number of shapes. This observation holds true for the LPIPS score (using VGG (Simonyan and Zisserman 2014)) plotted against the number of shapes, which is presented in the supplemental. It also outperforms DiffVG with our suggested initialization, except for the NFT-Apes dataset, where our DBSCAN initialization introduces a strong in-

[1]Runtime was measured on an NVIDIA RTX A5000 GPU.

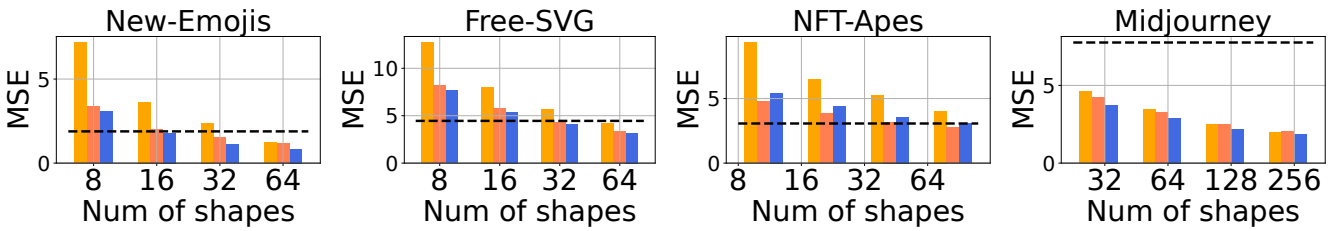


Figure 7: MSE (↓ better) vs Number of shapes: Our method (blue) outperforms DiffVG with random initialization (orange) for all five datasets. Moreover, using our proposed initialization (peach) improves the performance of DiffVG. Dashed line represents the MSE between DBSCAN-clustering and input image. The results for the Old-Emojis dataset are in the supplementary.

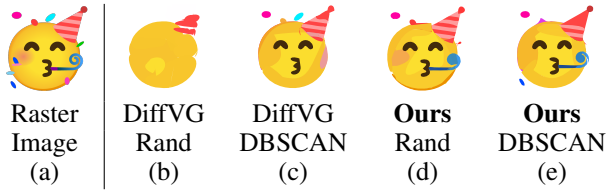


Figure 8: Components Contribution: Results using 16 shapes. DiffVG is remarkably sensitive to the curve's initialization, completely missing the party horn.

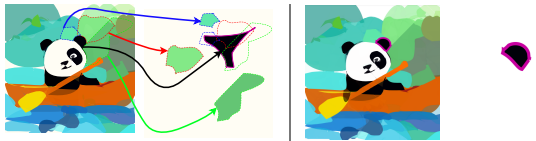


Figure 9: Convex Shapes: VectorFusion vs Ours: For VectorFusion, the right ear is composed of not less than four shapes. In our case, the ear is made of one shape. This demonstrates our compact representation and editability.

ductive bias, resulting in improved performance of DiffVG when using our initialization.

Qualitative Evaluation

The Contribution of DBSCAN and O&R Figure 7 shows that DBSCAN initialization reduces DiffVG's error. However, when a low number of shapes is considered, the introduction of O&R further decreases the error. Figure 8 provides insight into the effectiveness of our components - DBSCAN initialization and O&R optimization scheme. Figures 8(b) and (c) demonstrate DiffVG's sensitivity to initialization, emphasizing its limitations without effective initialization. Furthermore, the contribution of O&R optimization in mitigating local minima is showcased (Figures 8(c) and (d)). The comparison underscores that DBSCAN initialization alone falls short in yielding semantically satisfactory results for DiffVG. Notably, the primary driver of our algorithm's success lies in the O&R top-down approach, as demonstrated in Figures 8(d) and (e).

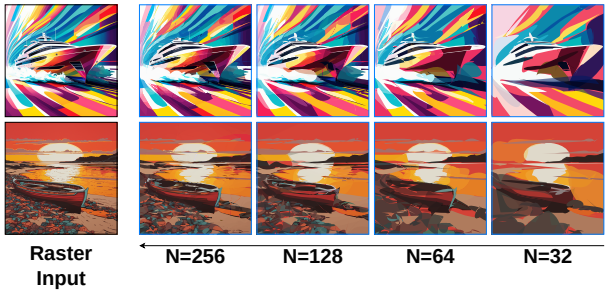


Figure 10: Results Showcase: We showcase more results of our method on a collection of images generated using Midjourney. Each image is vectorized using 256, 128, 64, and 32 shapes.

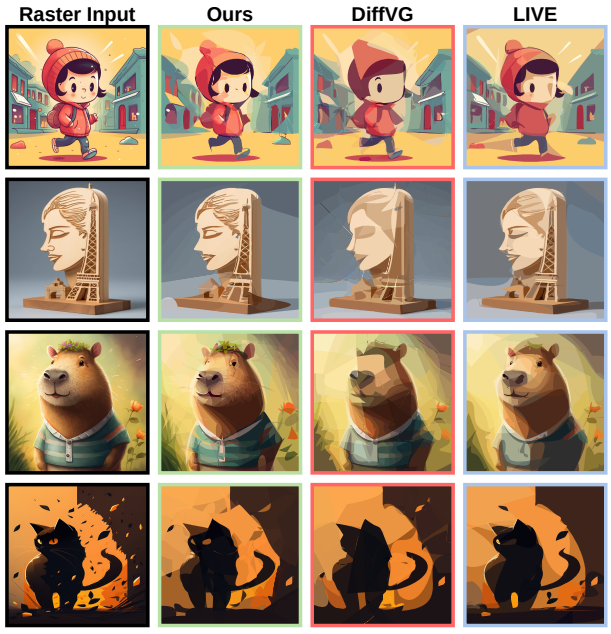


Figure 11: Qualitative Comparison: We showcase the results of DiffVG, LIVE, and Our method on a collection of images generated using Midjourney. Each image is compared using the same number of shapes.

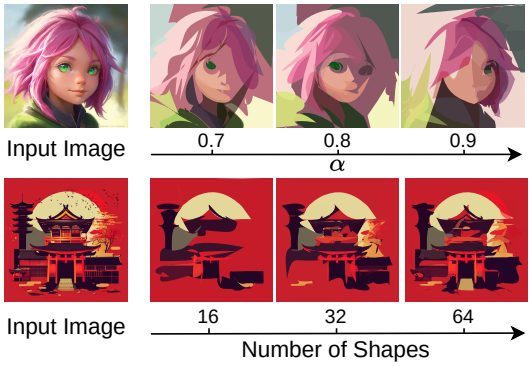


Figure 12: Abstraction Examples: Examples of (Top) different levels of abstraction using 16 shapes and various α values and number of shapes. Lower α means more abstraction (Equation 8). (Bottom) various number of shapes.

Geometric Loss VectorFusion (Jain, Xie, and Abbeel 2022) is a Text-to-SVG algorithm that given an optimizer such as DiffVG (Li et al. 2020) or LIVE (Ma et al. 2022), utilizes a text conditional diffusion model to create an SVG image from a text prompt. Figure 9 shows VectorFusion's output using LIVE's framework with its geometric loss and with SDS (Poole et al. 2022) loss, for "A Panda rowing a boat" prompt. Figure 9a shows LIVE's decomposition of the shapes composing the panda's ear. It takes four shapes to capture the ear's shape. Furthermore, the black shape is non-convex and exhibits a self-intersection. Figure 9b shows the result of taking the image rasterizing it, and vectorizing it with O&R. As highlighted with a pink solid line, we capture the panda's ear using a single *convex* shape. Additional results and comparisons can be seen in Figures 10 and 11.

Applications

Image Abstraction Inspired by CLIPasso (Vinker et al. 2022b), we demonstrate image abstraction with two degrees of freedom: (1) the number of shapes in the vector image representation, and (2) the extent to which we utilize a CLIP loss for optimization. Accompanied by the Midjourney image prompts, we propose employing CLIP loss, incorporating both its image-to-image components (image embeddings and image attention maps) and text-to-image components.

We define a hyperparameter α in Equation 8, which controls the extent to which a reconstruction loss (L_1 loss) is preferred over a contextualized loss ($\mathcal{L}_{CLIP}$) when optimizing for shapes parameters. The top row of Figure 12 demonstrates the effect of α on O&R for different values of α. For small values of α the image retains semantic features, whereas for larger values details are neglected, reflecting the effect of L_1 loss.

$$\mathcal{L} = \alpha L_1 + (1 - \alpha)\mathcal{L}_{CLIP} \tag{8}$$

Alternatively, we achieve abstraction with a lower number of shapes as demonstrated in the bottom row of Figure 12. Despite the significant variation in abstraction level, the output SVG remains easily identifiable as depictions of the same input image.

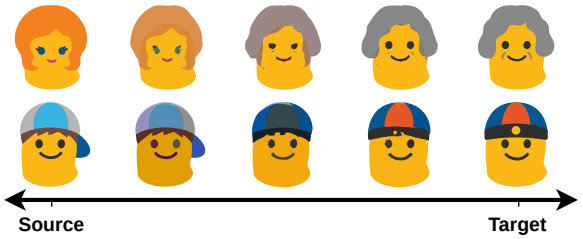


Figure 13: Interpolation: We perform an interpolation of emoji images from the rightmost column to the leftmost column. The intermediate images shown in between represent the optimization process's incremental results.

Interpolation Previous works, such as Im2Vec (Reddy et al. 2021a) and LIVE (Ma et al. 2022), have proposed interpolating emojis and character images in latent space, producing an interpolated latent representation that serves as input to a GAN, which generates a raster image. This raster image is then vectorized to create the vector output. In contrast, our approach does not rely on a GAN to generate interpolated images; instead, it performs interpolation directly in the shapes domain between the source and target images. Our method is preferable as it reflects a realistic scenario where a user wants to interpolate between two emoji images they possess. Our results are achieved without depending on a GAN trained to generate the raster images.

Figure 13 demonstrates an image interpolation application between two emoji images. Interpolating vector images imposes a double correspondence problem: (1) Resolving shape correspondence and (2) Bézier control point correspondence. To overcome this problem, we propose a new mechanism. We start with vectorizing the first image using O&R. Then, we use the vectorized image as the initial approximation for the shape structure and color and optimize the shapes, setting the second image as the target. During optimization, we obtain interpolated states between the two images. We leverage CLIP loss as an optimization objective for O&R to semantically interpolate the two images.

Conclusions

Optimize & Reduce is a top-down approach for vectorization, which excels when the budget of the number of shapes is low. The combination of O&R steps help the algorithm avoid local minima and converge to a solution that minimizes the reconstruction loss. Since O&R is a top-down approach, its' runtime is much faster than bottom-up approaches. Furthermore, our DBSCAN-based initialization and geometric loss term results in fast and editable results.

As part of our contribution, we suggest a benchmark dataset that facilitates the standardization of vectorization methods. We supplement our demonstrated reconstruction capabilities on this benchmark with applications of image abstraction and emoji interpolation.

Acknowledgments

Parts of this research were supported by ISF grant 1549/19.

References

Balasubramanian, S.; Balasubramanian, V. N.; et al. 2019. Teaching gans to sketch in vector format. *arXiv preprint arXiv:1904.03620*.

Bessmeltsev, M.; and Solomon, J. 2019. Vectorization of line drawings via polyvector fields. *ACM Transactions on Graphics (TOG)*, 38(1): 1–12.

Carlier, A.; Danelljan, M.; Alahi, A.; and Timofte, R. 2020. Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems*, 33: 16351–16361.

Chan, C.; Durand, F.; and Isola, P. 2022. Learning to generate line drawings that convey geometry and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7915–7925.

Chiang, J. Y.; Tue, S.; and Leu, Y. 1998. A new algorithm for line image vectorization. *Pattern recognition*, 31(10): 1541–1549.

Diebel, J. 2008. *Bayesian Image Vectorization: The Probabilistic Inversion of Vector Image Rasterization*. Stanford University.

Dominici, E. A.; Schertler, N.; Griffin, J.; Hoshyari, S.; Sigal, L.; and Sheffer, A. 2020. Polyfit: Perception-aligned vectorization of raster clip-art via intermediate polygonal fitting. *ACM Transactions on Graphics (TOG)*, 39(4): 77–1.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929.

Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X.; et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, 226–231.

Favreau, J.-D.; Lafarge, F.; and Bousseau, A. 2017. Photo2clipart: Image abstraction and vectorization using layered linear gradients. *ACM Transactions on Graphics (TOG)*, 36(6): 1–11.

Frans, K.; Soros, L.; and Witkowski, O. 2022. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *Advances in Neural Information Processing Systems*, 35: 5207–5218.

Gao, Y.; Guo, Y.; Lian, Z.; Tang, Y.; and Xiao, J. 2019. Artistic glyph image synthesis via one-stage few-shot learning. *ACM Transactions on Graphics (TOG)*, 38(6): 1–12.

Google. 2022. Noto Emoji.

Guo, Y.; Zhang, Z.; Han, C.; Hu, W.; Li, C.; and Wong, T.-T. 2019. Deep line drawing vectorization via line subdivision and topology reconstruction. In *Computer Graphics Forum*, volume 38, 81–90. Wiley Online Library.

Hoshyari, S.; Alberto Dominici, E.; Sheffer, A.; Carr, N.; Ceylan, D.; Wang, Z.; and Shen, I.-C. 2018. Perception-Driven Semi-Structured Boundary Vectorization. *ACM Transaction on Graphics*, 37(4).

Itoh, K.; and Ohno, Y. 1993. A curve fitting algorithm for character fonts. *Electronic publishing*, 6(3): 195–205.

Jain, A.; Xie, A.; and Abbeel, P. 2022. VectorFusion: Text-to-SVG by Abstracting Pixel-Based Diffusion Models. arXiv:2211.11319.

Jimenez, J.; and Navalon, J. L. 1982. Some experiments in image vectorization. *IBM Journal of research and Development*, 26(6): 724–734.

Kopf, J.; and Lischinski, D. 2011. Depixelizing pixel art. In *ACM SIGGRAPH 2011 papers*, 1–8.

Lai, Y.-K.; Hu, S.-M.; and Martin, R. R. 2009. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Transactions on Graphics (TOG)*, 28(3): 1–8.

Li, M.; Lafarge, F.; and Marlet, R. 2020. Approximating shapes in images with low-complexity polygons. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8633–8641.

Li, M.; Lin, Z.; Mech, R.; Yumer, E.; and Ramanan, D. 2019a. Photo-sketching: Inferring contour drawings from images. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1403–1412. IEEE.

Li, T.-M.; Lukáč, M.; Gharbi, M.; and Ragan-Kelley, J. 2020. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39(6): 1–15.

Li, Y.; Fang, C.; Hertzmann, A.; Shechtman, E.; and Yang, M.-H. 2019b. Im2pencil: Controllable pencil illustration from photographs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1525–1534.

Liu, S.; Lin, T.; He, D.; Li, F.; Deng, R.; Li, X.; Ding, E.; and Wang, H. 2021. Paint transformer: Feed forward neural painting with stroke prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, 6598–6607.

Lopes, R. G.; Ha, D.; Eck, D.; and Shlens, J. 2019. A learned representation for scalable vector graphics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 7930–7939.

Ma, X.; Zhou, Y.; Xu, X.; Sun, B.; Filev, V.; Orlov, N.; Fu, Y.; and Shi, H. 2022. Towards layer-wise image vectorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16314–16323.

Mihai, D.; and Hare, J. 2021. Differentiable drawing and sketching. *arXiv preprint arXiv:2103.16194*.

Mo, H.; Simo-Serra, E.; Gao, C.; Zou, C.; and Wang, R. 2021. General virtual sketching framework for vector line art. *ACM Transactions on Graphics (TOG)*, 40(4): 1–14.

Muhammad, U. R.; Yang, Y.; Song, Y.-Z.; Xiang, T.; and Hospedales, T. M. 2018. Learning deep sketch abstraction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8014–8023.

Noris, G.; Hornung, A.; Sumner, R. W.; Simmons, M.; and Gross, M. 2013. Topology-driven vectorization of clean line drawings. *ACM Transactions on Graphics (TOG)*, 32(1): 1–11.

Olsen, S.; and Gooch, B. 2011. Image simplification and vectorization. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, 65–74.

Orzan, A.; Bousseau, A.; Winnemöller, H.; Barla, P.; Thollot, J.; and Salesin, D. 2008. Diffusion curves: a vector representation for smooth-shaded images. *ACM Transactions on Graphics (TOG)*, 27(3): 1–8.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

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

Reddy, P.; Gharbi, M.; Lukac, M.; and Mitra, N. J. 2021a. Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7342–7351.

Reddy, P.; Zhang, Z.; Wang, Z.; Fisher, M.; Jin, H.; and Mitra, N. 2021b. A multi-implicit neural representation for fonts. *Advances in Neural Information Processing Systems*, 34: 12637–12647.

Riso, M.; Sforza, D.; and Pellacini, F. 2022. pOp: Parameter Optimization of Differentiable Vector Patterns. In *Computer Graphics Forum*, volume 41, 161–168. Wiley Online Library.

Selinger, P. 2003. Potrace: a polygon-based tracing algorithm.

Shen, I.-C.; and Chen, B.-Y. 2021. Clipgen: A deep generative model for clipart vectorization and synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 28(12): 4211–4224.

Shimoda, W.; Haraguchi, D.; Uchida, S.; and Yamaguchi, K. 2021. De-rendering stylized texts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1076–1085.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Song, Y.; Shao, X.; Chen, K.; Zhang, W.; Jing, Z.; and Li, M. 2023. CLIPVG: Text-Guided Image Manipulation Using Differentiable Vector Graphics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 2312–2320.

Sun, J.; Liang, L.; Wen, F.; and Shum, H.-Y. 2007. Image vectorization using optimized gradient meshes. *ACM Transactions on Graphics (TOG)*, 26(3): 11–es.

Tian, Y.; and Ha, D. 2022. Modern evolution strategies for creativity: Fitting concrete images and abstract concepts. In *Artificial Intelligence in Music, Sound, Art and Design: 11th International Conference, EvoMUSART 2022, Held as Part of EvoStar 2022, Madrid, Spain, April 20–22, 2022, Proceedings*, 275–291. Springer.

Vinker, Y.; Alaluf, Y.; Cohen-Or, D.; and Shamir, A. 2022a. CLIPascene: Scene Sketching with Different Types and Levels of Abstraction. *arXiv preprint arXiv:2211.17256*.

Vinker, Y.; Pajouheshgar, E.; Bo, J. Y.; Bachmann, R. C.; Bermano, A. H.; Cohen-Or, D.; Zamir, A.; and Shamir, A. 2022b. Clipasso: Semantically-aware object sketching. *ACM Transactions on Graphics (TOG)*, 41(4): 1–11.

Wang, Y.; and Lian, Z. 2021. DeepVecFont: Synthesizing high-quality vector fonts via dual-modality learning. *ACM Transactions on Graphics (TOG)*, 40(6): 1–15.

Weber, M. 2004. Autotrace-converts bitmap to vector graphics.

Xia, T.; Liao, B.; and Yu, Y. 2009. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Transactions on Graphics (TOG)*, 28(5): 1–10.

Xie, G.; Sun, X.; Tong, X.; and Nowrouzezahrai, D. 2014. Hierarchical diffusion curves for accurate automatic image vectorization. *ACM Transactions on Graphics (TOG)*, 33(6): 1–11.

Yi, R.; Liu, Y.-J.; Lai, Y.-K.; and Rosin, P. L. 2019. Apdrawinggan: Generating artistic portrait drawings from face photos with hierarchical gans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10743–10752.

Zhang, S.-H.; Chen, T.; Zhang, Y.-F.; Hu, S.-M.; and Martin, R. R. 2009. Vectorizing cartoon animations. *IEEE Transactions on Visualization and Computer Graphics*, 15(4): 618–629.

Zhao, S.; Durand, F.; and Zheng, C. 2017. Inverse diffusion curves using shape optimization. *IEEE transactions on visualization and computer graphics*, 24(7): 2153–2166.

Zheng, N.; Jiang, Y.; and Huang, D. 2019. Strokenet: A neural painting environment. In *International Conference on Learning Representations*.

Zhu, H.; Cao, J.; Xiao, Y.; Chen, Z.; Zhong, Z.; and Zhang, Y. J. 2022. TCB-spline-based Image Vectorization. *ACM Transactions on Graphics (TOG)*, 41(3): 1–17.

Zou, Z.; Shi, T.; Qiu, S.; Yuan, Y.; and Shi, Z. 2021. Stylized neural painting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15689–15698.