

# Layerwise Image Vectorization via Bayesian-Optimized Contour

Ghfran Jabour<sup>a</sup>, Sergey Muravyov<sup>b</sup> and Valeria Efimova<sup>c</sup>

ITMO University, Kronverksky Pr. 49, bldg. A, St. Petersburg, 197101, Russia  
fi

**Keywords:** Image Vectorization, Contour-Based Initialization, Bayesian Optimization, Scalable Vector Graphics, Computational Efficiency, Reconstruction Fidelity, Path Optimization, Digital Content Creation, Semantic Simplification, Superpixel-Based Vectorization.

**Abstract:** This work presents a novel method LIVBOC for complex image vectorization that addresses key challenges in path initialization, color assignment, and optimization. Unlike existing approaches such as LIVE, our method generates Bayesian-optimized contour for path initialization, which is then optimized using a customized loss function to align it better with the target shape in the image. In our method, adaptive selection of points and parameters for efficient and accurate vectorization is enabled to reduce unnecessary iterations and computational overhead. LIVBOC achieves superior reconstruction fidelity with fewer paths, and that is due to the path initialization technique, which initializes paths as contours that approximate target shapes in the image, reducing redundancy in points and paths. The experimental evaluation indicates that LIVBOC outperforms LIVE in all key metrics, including a significant reduction in  $L_2$  loss, processing time, and file size. LIVBOC achieves comparable results with just 100 iterations, compared to LIVE's 500 iterations, while preserving finer details and generating smoother, more coherent paths. These improvements make LIVBOC more suitable for applications that require scalable, compact vector graphics, and computational efficiency. By achieving both accuracy and efficiency, LIVBOC offers a new robust alternative for image vectorization tasks. The LIVBOC code is available at <https://github.com/CTLab-ITMO/LIVBOC>.

## 1 INTRODUCTION

Image vectorization, the process of converting raster images into scalable vector graphics (SVG), has become an essential tool in various fields, including graphic design, web development, and scientific visualization. Vector graphics offer distinct advantages over raster images, such as infinite scalability, smaller file sizes, and efficient editing capabilities. Despite the importance of vector graphics, this process comes with multiple challenges. Obtaining high-quality vectorization, especially for complex images with important details, is not an easy task (Dziuba et al., 2023), as we will need to trade off many metrics listed above. Image vectorization methods must balance the fidelity of reconstruction, computational efficiency, and compactness while preserving the key features of the input image.

Many traditional vectorization methods often use fixed strategies for path initialization and param-

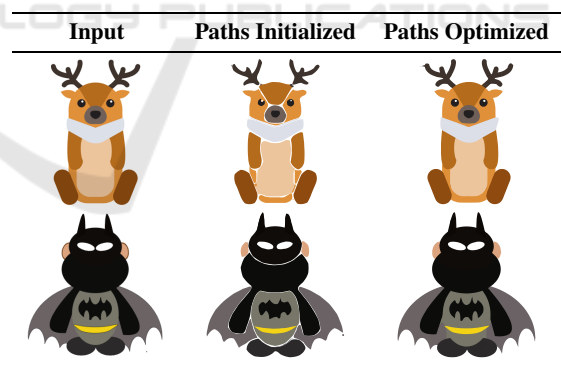


Figure 1: Example showing the output of each stage of the LIVBOC method (path initialization and path optimization).

ter settings, which limit their ability to adapt to diverse image contents. As an example, SAMVG (Zhu et al., 2023) utilizes segmentation models for vectorization, but highly depends on the quality of segmentation maps, which may struggle with overlapping objects. Another example is SuperSVG (Hu et al., 2024), which simplifies vectorization using superpixels but may sacrifice fine details in highly textured areas. Recent advances, such as LIVSS (Wang et al.,

<sup>a</sup> <https://orcid.org/0009-0003-8092-9804>

<sup>b</sup> <https://orcid.org/0000-0002-4251-1744>

<sup>c</sup> <https://orcid.org/0000-0002-5309-2207>

2024), incorporate semantic segmentation to prioritize meaningful regions but are computationally intensive due to their iterative processing.

To achieve a high-quality low-sized SVGs, we propose LIVBOC (Layer-wise Image Vectorization with Bayesian-Optimized Contours), an image vectorization method of two main stages, a Bayesian-optimized contour-based path initialization stage and a path optimization stage. Contour-based techniques have been shown to be effective in capturing shape boundaries while minimizing redundancy (Arbeláez et al., 2011; Polewski et al., 2024). Furthermore, Bayesian optimization (Frazier, 2018; Snoek et al., 2012) have found widespread use in machine learning for hyperparameter optimization. Building on those two well-established techniques, the first stage of our method utilizes the Bayesian optimization algorithm to determine the optimal values for the parameters used in identifying the optimal contour used in path initialization. This stage gives the method the ability to adapt to the complexity and variations of colors in the input image. In the second stage, after path initialization, we optimize the initialized path with a custom loss function composed of three loss functions: the reconstruction loss to preserve information and guarantee a correct direction of the path during the optimization process, the Laplacian smoothness loss based on (Sorkine et al., 2004) to ensure smoothness of paths, and the overlap loss to prevent shapes from overlapping with other correct shapes. With these two stages, LIVBOC ensures not only achieving superior reconstruction fidelity but also reducing the number of required paths and computational time. Unlike previous methods, which often require a fixed number of iterations for convergence, LIVBOC achieves high-quality results with few iterations by integrating robust initialization with efficient optimization. Thus, our method fits for applications demanding scalable and high-fidelity vector graphics, such as web graphics, logo design, and digital content creation.

In our experiments, we compare our LIVBOC method with LIVE (Ma et al., 2022) and demonstrate LIVBOC strength. Through a multi-faceted comparison, we highlight the efficiency of our method in generating high-quality vector image with optimal primitives (optimal SVG file size).

Key contributions of our paper are as follows:

- We propose LIVBOC, a novel image vectorization method that generates minimized SVG files by using an adaptive number of points for different paths.
- The resulting vector primitives are optimal or near-optimal, allowing the user to better manipulate the vectors.

- The proposed LIVBOC method is 3-5 times faster than current state-of-the-art machine learning-based vectorization.

## 2 RELATED WORK

### 2.1 Layered Image Vectorization via Semantic Simplification (LIVSS)

The LIVSS method (Wang et al., 2024) is a novel approach to image vectorization that uses semantic information to guide the vectorization process. In this method, Score Distillation Sampling (SDS) and semantic segmentation are calculated to iteratively simplify an input image while preserving essential features. By doing that, LIVSS creates a series of simplified layers, each of these layers contains varying levels of detail. Then these layers are optimized to create a high-quality vector representations with adjustable levels of fidelity.

The main strength of LIVSS is the ability to balance detail preservation and vectorization compactness. Using semantic segmentation, the method prioritizes meaningful regions first, with the possibility of ignoring redundant details in the background. This technique makes LIVSS effective for applications of vectorizing images containing semantic meaningful shapes.

Despite that, the performance of LIVSS is highly dependent on the quality of its semantic segmentation. Therefore, if segmentation fails to accurately delineate complex objects or overlapping structures, the resulting vectorized output may lose critical details or introduce artifacts. Another limitation that might appear in this method is that it might be unable to segment unnecessary shapes, which can lead to low quality images. It would be valuable to compare our method with LIVSS, but at the time of writing this paper, the LIVSS code is not publicly available.

### 2.2 SuperSVG

SuperSVG (Hu et al., 2024) represents a significant advance in the field of image vectorization in 2024. This method mainly utilize a superpixel-based framework to decompose raster images into regions with uniform colors and textures, thereby enabling efficient vectorization. SuperSVG is the two-stage self-training framework. In the first stage, the model reconstructs the primary structure of the image using superpixels, focusing on large, uniform areas in the image. In the second stage, it improves the output by

enriching or adding fine-grained details, such as regions with complicated shapes and textures.

One of the key strengths of SuperSVG is that it balances computational efficiency and precision. Because by focusing on superpixels, the method reduces the complexity of processing high-resolution images while keeping an acceptable levels of detail in the output.

SuperSVG also has some limitations, for example, the reliance on superpixels can oversimplify complex textures, which leads to less accurate reconstructions in situations where the image is highly detailed. Another point to consider is that the two-stage can cause an additional computational complexity that may not be ideal tasks that prioritize computational efficiency.

### 2.3 SAMVG

SAMVG (Segment-Anything Model for Vector Graphics) (Zhu et al., 2023), a multi-staged method that uses the Segment-Anything Model (SAM) to perform general image segmentation. The first stage is the process of retrieving Segmentation Masks, where image is divided into a dense set of regions for capturing complex boundaries and features. Next stages are for approximation and optimization of the traced shapes in the masks. In the final stage, these segments are converted into a structured Scalable Vector Graphics (SVG) format that incorporates detailed components to enhance the quality of the output.

SAMVG also provides the ability to handle a wide variety of image domains effectively, thanks to the robustness of SAM in segmenting diverse and complex images. It is also computationally efficient as its filtering and conversion steps are optimized to minimize processing overhead without compromising the quality of the vectorized result. Therefore, SAMVG is useful for applications requiring computational efficiency and high quality output. However, during the high dependence on the Segmentation model SAM, its accuracy directly impacts the quality of the final vector output. Therefore, in cases where segmentation struggles with overlapping or unclear objects, the vector output may contain errors or inconsistencies.

### 2.4 EvoVec

Evolutionary Image Vectorization (Bazhenov et al., 2024) is an evolutionary algorithm for image vectorization, it introduces a method that addresses very common challenges in existing deterministic and machine learning-based methods. For example, a very common challenge for deterministic algorithms is handling color gradient. While for machine learn-

ing approaches, common challenges can be defining curve numbers, computational complexity, and also the struggle with gradient representations. EvoVec faces these challenges by including mutations and crossovers to iteratively refine vectorized images. It also adaptively determines the number of curves needed to accurately represent the input image. Another valuable feature of this method, is that it can merge gradient-based path to give a better presentation of color transitions in the image. Experiments of EvoVec demonstrated 15% improvement in pixel-by-pixel mean square error (MSE) compared to other state-of-the-art methods, with a competitive computational efficiency.

## 3 METHOD

The proposed method introduces a two-staged approach for complex images vectorization that aims to address key challenges in path initialization, color assignment, and optimization. Unlike existing methods, our approach ensures adaptive and accurate path generation with a consistent color representation, reducing unnecessary iterations, and improving overall fidelity. The process is clearly divided into two main stages: Path Initialization and Path Optimization, where in the first stage we initialize an accurate or close to accurate path, remaining the need for only few adjustments in the second stage to ensure a better adjustment and alignment for the generated path.

### 3.1 Path Initialization

The path initialization stage is the foundation for efficient and accurate vectorization. A path (or a vector) is represented by a set of points that contours the target shape in the input image, and a color assigned to the points contained inside that contour. In a layerwise vectorization technique, we define a set of layers, each containing a set of paths, these layers are stacked hierarchically, where each layer introduces new information. From that definition, we can notice that each layer depends on previous set of layers, and the first layer doesn't depend on any layer (or we can say depends on an empty layer). In order to extract contours of each layer, a difference map is calculated as the  $L_2$  between the ground truth and the stack of previous layers is calculated to define key regions of variation. In order to enhance the quality of contours extracted from these key regions, we use the Bayesian optimization algorithm to fine-tune important parameters of operations applied to the difference map, such as thresholding, quantization, and morphological re-

finement. The color of each path is extracted using the k-means clustering method (Frackiewicz et al., 2019). Therefore, the output of this stage is an optimized contour (a set of points) and its color, forming by that a path that will be added to the current layer.

### 3.1.1 Contour Extraction

The initialization phase begins by generating a difference map between the ground truth (GT) image and an initial predicted image (uniformly white). The following steps are applied to extract an efficient contour:

1. **Thresholding.** Intensities in the difference map below the threshold value are set to zero in order to remove noise of small differences therefore to isolate significant regions of variation.
2. **Quantization.** After thresholding, the remaining pixel intensities are quantized to produce discrete levels from the difference map. Quantization relies on a range of intervals derived from the flattened difference map, with keeping only central quantized values to maintain optimal granularity.
3. **Morphological Refinement.** Morphological operations (erosion and dilation) are used to refine the quantized difference map (applying kernels to the difference map to clear/close partial lines), producing by that a well-defined path boundaries.
4. **Component Analysis.** Connected-component analysis (connected components algorithm used by OpenCV) identifies distinct regions in the refined map and the contour of the largest connected component is extracted and designated as the initial path.

### 3.1.2 Color Selection

After extracting the contour of the largest component, we utilize the K-means clustering algorithm to extract the color of that component in the ground truth (GT) image. Color extraction steps are as follows:

- The region in the GT image corresponding to the largest component in the difference map is extracted.
- In the extracted area from the GT image, only pixels close to boundary are considered, minimizing interference from adjacent shapes.
- The K-means clustering algorithm is applied to the extracted set of pixels (Frackiewicz et al., 2019), and the dominant color (the centroid of the largest cluster) is selected as the color assigned to the path being initialized.

This method ensures that the assigned color accurately reflects the shape's identity while being unaffected by neighboring regions.

### 3.1.3 Bayesian Optimization for Parameter Tuning

Bayesian optimization is used to fine-tune key parameters (Nguyen, 2019) used in extracting contours from the difference map: 1) The quantile interval parameter used for determining the segmentation granularity; 2) The thresholding parameter used for filtering out minor variations and noises for a better focus on significant regions of interest; and 3) kernel size used in morphological operations (erosion and dilation) that aims for removing noises or closing partial paths. Figure 2 illustrates how different values of quantile interval and thresholding parameters affect generated mask and contour initialization.

### 3.1.4 Loss Function Design

The objective function used in the Bayesian optimization algorithm is a composite loss function that balances between reconstruction accuracy, overlap minimization, and path efficiency. The key components of the loss function are:

- **Reconstruction Loss**, which ensures accuracy and fidelity by penalizing differences between the ground truth (GT) and predicted images:

$$\mathcal{L}_{\text{Reconstruction}} = \sum (I_{\text{GT}} - I_{\text{Pred}})^2 \quad (1)$$

- **Overlap Loss** to discourage paths that overlap with regions that already had a better prediction:

$$\mathcal{L}_{\text{Overlap}} = \sum_{i,j} (\mathcal{L}_{i,j} > \mathcal{L}_{\text{prev},i,j} \wedge \mathcal{L}_{\text{prev},i,j} < \epsilon) \quad (2)$$

Where  $\mathcal{L}_{i,j}$  is the current reconstruction loss of the pixel  $i,j$  of the image after adding the current path, and  $\mathcal{L}_{\text{prev},i,j}$  is the previous reconstruction loss for this pixel before adding the current path.

- **Coverage Loss** to encourage Bayesian optimization process to select larger shapes to contour. Because if small shapes inside a bigger one were contoured first, it will complicate the process of contouring the large one later. Let  $S$  donate the shape inside the predicted path, the coverage loss formula is defined as follows:

$$\mathcal{L}_{\text{Coverage}} = \sum_{(i,j) \notin S} 1_{(i,i)} \quad (3)$$

Therefore, the final loss function used as objective function for the Bayesian Optimization process is the weighted sum of the three loss functions  $\zeta_{\text{Bayesian}}$ :

$$\mathcal{L}_{\text{Bayesian}} = \mathcal{L}_{\text{Reconstruction}} + \lambda \mathcal{L}_{\text{Overlap}} + \gamma \mathcal{L}_{\text{Coverage}} \quad (4)$$

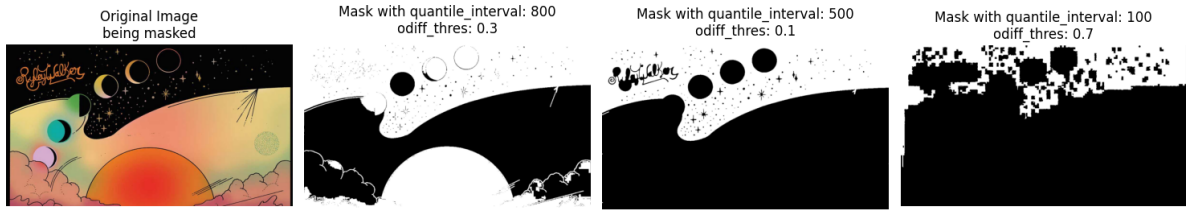


Figure 2: The plot of the largest quantile in the difference map after applying thresholding and quantization with different values for parameters: quantile interval and thresholding. Showing their effect on contour extraction process.

However, the newly initialized path can result in a higher reconstruction error (the  $L_2$  loss is larger than before applying the new path), then the loss is set to infinity, effectively rejecting the path during Bayesian optimization.

### 3.2 Path Optimization

The goal of this stage is mainly to ensure that the path resulted from the previous stage (represented as the target shape contour and its color) is well aligned with its corresponding shape in the target image. Therefore, we define a composite loss function that ensures a correct alignment, and smoothness of the path.

1. **Reconstruction Loss:** penalizes the deviation between the ground truth (GT) image and the predicted image.

$$\mathcal{L}_{reconstruction} = \sum_{i,j,c \in \{R,G,B\}} (I_{GT,k,c} - I_{pred,k,c})^2 \quad (5)$$

2. **Laplacian Smoothness Loss:** is designed to ensure the geometric smoothness of the path (Vartziotis and Himpel, 2014) during the optimization process. It prevents the points from being optimized in random directions.

$$\mathcal{L}_{smooth} = \sum_i \left\| P_i - \frac{1}{2}(P_{i-1} + P_{i+1}) \right\|^2 \quad (6)$$

Where  $p_i$  is the position of point  $i$  of the path predicted.

Therefore, the Laplacian Smoothness Loss is added to encourage paths to adopt smooth, continuous shapes by reducing sharp angles or irregularities.

3. **Overlap Loss:** is similar to the overlap loss used in the first stage 3.1, but instead of comparing the current path with the currently predicted image, we compare it with a 3D array that has the same shape as the target image, and preserves the best color assigned to each pixel across all previous iterations. The overlap loss is described in the following equation:

$$\mathcal{L}_{overlap} = \sum_{i,j} (\mathcal{L}_{i,j} > \mathcal{L}_{previ,j}) \quad (7)$$

Where  $\mathcal{L}_{i,j}$  is the current reconstruction loss of the pixel  $i,j$  of the image, and  $\mathcal{L}_{previ,j}$  is the lowest reconstruction loss recorded for this pixel through all previous iterations.

And finally, the total loss for the path optimization stage is the weighted sum of these three losses, and given by the following equation.

$$\mathcal{L}_{total} = \mathcal{L}_{reconstruction} + \alpha \mathcal{L}_{smooth} + \beta \mathcal{L}_{overlap} \quad (8)$$

This multi-objective optimization ensures that the refined paths not only align accurately with the GT image but also maintain smooth, visually appealing geometries.

This structured process ensures precise and efficient path initialization, requiring minimal refinement during the Path Optimization stage.

## 4 EXPERIMENTS AND RESULTS

In this section, we explore the behavior of our method (LIVBOC) and compare it with the LIVE method (Ma et al., 2022) from different aspects on a set of 30 images, varying from simple (it may need about 16 to 32 paths for the LIVE method) to complex (it may need about 256 to 512 paths), while for LIVBOC method, it can use half of this number of paths and maybe even less.

### 4.1 Experimental Setup

The experiments were carried out on an NVIDIA GeForce MX110 GPU, using Python as the programming language. The Adam optimizer is used for optimizing paths in both methods under evaluation: LIVBOC and LIVE. Optimization hyperparameters such as learning rates, iteration counts, and LIVBOC initialization strategies are different from the LIVE hyperparameters.

**Hyperparameter Optimization.** For LIVBOC, we set the number of clusters in the K-means algorithm to 3, and the number of segments of each path to 4, with the total number of points adaptively adjusted

based on the shape contours. The LIVE method algorithm generally sets a fixed total of 12 points for each path, and these points are evenly divided over the same number of segments. Both methods were configured without stroke training (the stroke width is set to zero). In terms of loss functions, the overlap loss function is weighted as 1.0, and the smoothness loss weight is set to 0.05, while the LIVE method used a Xing loss weight of 0.01.

In terms of learning rates, in experiments of our method, the high efficiency of our loss functions gives us the confidence to set the point learning rate to 2.0, while for LIVE it is set to 1.0 to avoid overshooting. The fill color learning rate in LIVE is set to 0.01, while for LIVBOC, it is set to 0.0. The number of iterations is set to 100 for our method, while for the LIVE method it is set to 500 iterations.

We conducted experiments on both methods using the same 30 images, which provide a range of color variations and shape contours suitable for vectorization tasks.

## 4.2 Results and Discussion

**Quantitative Comparison.** In order to make a quantitative comparison, We compare calculate the value of 6 metrics for both methods on the same images, metrics are: 1)  $L_2$  loss, to compare information preservation; 2) time consumed to compare computational efficiency; 3) file size to compare vectorization compactness and efficiency in points generation; 4) path number to compare redundancy of information; 5) PSNR, to assess the quality of reconstruction and signal fidelity; and 6) SSIM, to evaluate the structural similarity and perceptual quality of the results (Horé and Ziou, 2010).

As shown in Table 1, our method outperforms the LIVE method, achieving a lower mean and standard deviation for  $L_2$  loss, which means that not only LIVBOC gives better reconstruction fidelity, but also it gives a more consistent and reliable reconstruction than LIVE. SSIM standard deviation for LIVE is slightly less than for LIVBOC, but the good difference between the mean of SSIM values registered by LIVBOC and by LIVE still highlights the LIVBOC's superiority over LIVE in vectorizing images with better perceptual quality.

The LIVBOC method also generates smaller SVG file sizes and reduces the average number of paths, which indicates the improvement of vectorization efficiency while avoiding information redundancy, and at the same time, without compromising accuracy. LIVBOC generates contour-based paths that are represented by a non-fixed number of points depending

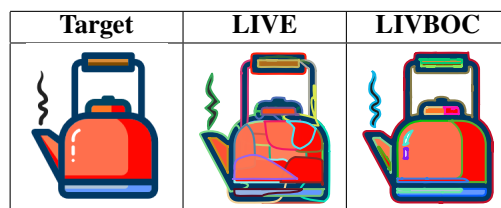


Figure 3: Comparison of path efficiency between LIVE and LIVBOC methods. LIVBOC assigns a single path per shape, while LIVE generates multiple paths for the same shape.

on the size of the shape contoured. This feature helps LIVBOC to represent each shape with a single path, even if this target shape is partially covered by other shapes in the ground-truth image. By minimizing information duplication—where a single shape would otherwise require multiple paths for representation (as is the case with LIVE), LIVBOC reduces the size of SVG output files, as shown in Figure 3 and evidenced in Table 1. The generation of contour-based paths also helps in reducing time needed for reaching the target shape, therefore, reducing overall time of the vectorization process.

**Visual Comparison.** In Figure 4, we show the vectorization results for samples of different levels of complexity.

In **simple** samples, while both methods give accurate vectorization, LIVE still misses some important paths, resulting in not only a lower perceptual quality of the output image, but also in consuming a higher number of paths and a larger storage for the output file due to information duplication.

**Medium** samples show a clear superiority of the LIVBOC performance where we can notice how LIVE is generating overlapping shapes, where vectors are newer covering previous correct ones, while our method avoids that mistake thanks to the usage of the overlap loss function.

**Complex** samples show the core point of power of our method. One of the limitations of the LIVE method observed during experiments is its tendency to generate random paths when no further shapes are available for vectorization. This behavior can decrease reconstruction fidelity and lead to the loss of perceptual quality. In contrast, the Bayesian optimization step in LIVBOC enables it to consistently identify missing paths, as demonstrated in complex samples with fine details.

As can be seen in Table 1, our LIVBOC method significantly reduces processing time compared to the LIVE baseline, while at the same time, achieving similar or better results in just 100 iterations versus the 500 iterations required by LIVE. This efficiency is due to LIVBOC's contour-based initialization, which

Table 1: Comparison between LIVE and our LIVBOC across 30 images of different paths' complexity.

Metric	$L_2$ (Mean $\pm$ SD)	$SSIM$ (Mean $\pm$ SD)	Time, seconds	File Size (KB)	Path Number
LIVE	$0.0107 \pm 0.0069$	$0.8943 \pm 0.0784$	22470	62.5	99.34
LIVBOC (ours)	$0.0056 \pm 0.0038$	$0.9334 \pm 0.0788$	7802	56.48	46.05

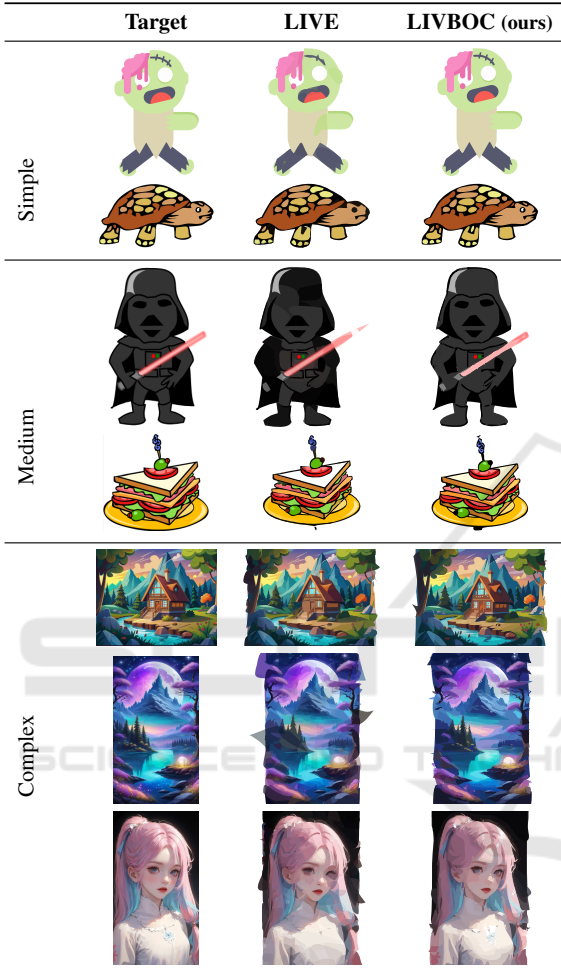


Figure 4: Samples of LIVE and our method results on images of various complexities.

is optimized using Bayesian methods. In contrast, LIVE initializes small B-spline paths using fixed, non-optimized parameter values, resulting in the need for more refinement for paths during optimization.

## 5 CONCLUSION

In this work, we have introduced LIVBOC, a novel method for image vectorization that addresses key challenges in path initialization, color assignment, and optimization. With the Bayesian-contour-based initialization strategy, LIVBOC achieves a significant

reduction in computational overhead while maintaining a better preservation of information and reconstruction fidelity. The method adaptively sets the number of points required for each shape, providing efficient path generation that minimizes redundancy and file size.

The experimental results indicate that LIVBOC outperforms the baseline LIVE method in all key metrics. LIVBOC achieves a lower  $L_2$  loss, faster processing times, and more compact SVG representations, all while requiring fewer paths to represent complex images. Qualitative comparisons further highlight LIVBOC's ability to preserve intricate details and avoid artifacts, resulting in smoother and more accurate vectorizations, thus, a user can manipulate with vectors more easily.

These results establish LIVBOC as a robust and efficient alternative to existing methods, with applications in scalable vector graphics, digital design, and computational graphics. Future work will focus on enhancing the LIVBOC coloring capabilities. Specifically, our goal is to implement and optimize more advanced coloring techniques, such as gradients and pattern-based fills, in addition to solid fill.

## ACKNOWLEDGMENTS

The research was supported by the ITMO University, project 623097 "Development of libraries containing perspective machine learning methods".

## REFERENCES

- Arbeláez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916.
- Bazhenov, E., Jarsky, I., Efimova, V., and Muravyov, S. (2024). Evovec: Evolutionary image vectorization with adaptive curve number and color gradients. In Affenzeller, M., Winkler, S. M., Kononova, A. V., Trautmann, H., Tušar, T., Machado, P., and Bäck, T., editors, *Parallel Problem Solving from Nature – PPSN XVIII*, pages 383–397, Cham. Springer Nature Switzerland.
- Dziuba, M., Jarsky, I., Efimova, V., and Filchenkov, A. (2023). Image vectorization: a review.

- Frackiewicz, M., Mandrella, A., and Palus, H. (2019). Fast color quantization by k-means clustering combined with image sampling. *Symmetry*, 11.
- Frazier, P. I. (2018). A tutorial on bayesian optimization.
- Horé, A. and Ziou, D. (2010). Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369.
- Hu, T., Yi, R., Qian, B., Zhang, J., Rosin, P. L., and Lai, Y.-K. (2024). Supersvg: Superpixel-based scalable vector graphics synthesis.
- Ma, X., Zhou, Y., Xu, X., Sun, B., Filev, V., Orlov, N., Fu, Y., and Shi, H. (2022). Towards layer-wise image vectorization.
- Nguyen, V. (2019). Bayesian optimization for accelerating hyper-parameter tuning. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 302–305.
- Polewski, P., Shelton, J., Yao, W., and Heurich, M. (2024). Segmenting objects with bayesian fusion of active contour models and convnet priors.
- Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms.
- Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., and Seidel, H.-P. (2004). Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184.
- Vartziotis, D. and Himpel, B. (2014). Laplacian smoothing revisited.
- Wang, Z., Huang, J., Sun, Z., Cohen-Or, D., and Lu, M. (2024). Layered image vectorization via semantic simplification.
- Zhu, H., Chong, J. I., Hu, T., Yi, R., Lai, Y.-K., and Rosin, P. L. (2023). Samvg: A multi-stage image vectorization model with the segment-anything model.