SAMVG: A MULTI-STAGE IMAGE VECTORIZATION MODEL WITH THE SEGMENT-ANYTHING MODEL

Haokun Zhu[1,♯], *Juang Ian Chong*[1,♯], *Teng Hu*[1], *Ran Yi*[1,*], *Yu-Kun Lai*[2], *Paul L. Rosin*[2]

[1]Shanghai Jiao Tong University, [2]Cardiff University

{zhuhaokun,ianchong,hu-teng,ranyi}@sjtu.edu.cn, {LaiY4,RosinPL}@cardiff.ac.uk

ABSTRACT

Vector graphics are widely used in graphical designs and have received more and more attention. However, unlike raster images which can be easily obtained, acquiring high-quality vector graphics, typically through automatically converting from raster images, remains a significant challenge, especially for more complex images such as photos or artworks. In this paper, we propose SAMVG, a multi-stage model to vectorize raster images into SVG (Scalable Vector Graphics). Firstly, SAMVG uses general image segmentation provided by the Segment-Anything Model and uses a novel filtering method to identify the best dense segmentation map for the entire image. Secondly, SAMVG then identifies missing components and adds more detailed components to the SVG. Through a series of extensive experiments, we demonstrate that SAMVG can produce high quality SVGs in any domain while requiring less computation time and complexity compared to previous state-of-the-art methods.

Index Terms— Image Vectorization, Vector Graphics, Image Segmentation, Computer Graphics, Segment-Anything Model

1. INTRODUCTION

While raster images are commonly used for their adaptability and detailed representation, vector graphics have unique advantages. They represent images as mathematical equations rather than pixels, making them ideal for resizing without quality loss, which is especially useful for logos and icons.

Generating vector graphics, especially for complex images, is a challenging task compared to raster images which can be easily obtained. Existing methods [1, 2, 3] generate vector graphics capable of faithfully reconstructing original images. But the produced representations tend to contain too many unnecessary parameters and fail to preserve essential topological features. Recent advances in deep learning and differentiable rendering have inspired new approaches [4, 5, 6, 7, 8, 9, 10, 11] aiming to create visually similar vector graphics while also preserving topological features. However,

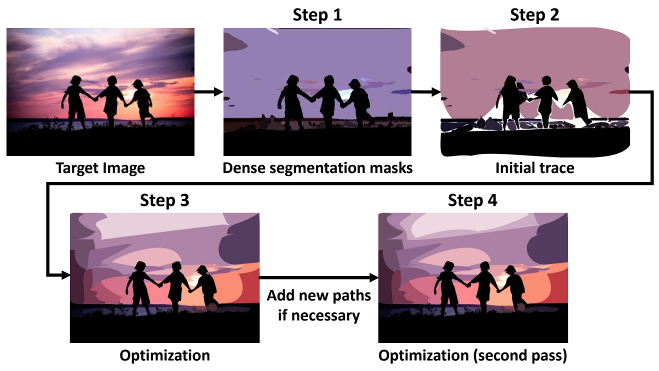

Fig. 1. High-level overview of SAMVG.

deep generative models [6, 7, 8, 9, 10, 12] face limitations related to high-quality training data availability and computationally intensive rendering-based training. Direct optimization algorithms [4, 5, 13] are more versatile but require good initialization for optimal results. The recent method LIVE [4] addresses this by incrementally adding shape primitives, but can be computationally expensive for batch processing.

In this paper, we aim to solve the initialization problem in direct optimization-based image vectorization methods in order to efficiently achieve high-quality results. A common image vectorization method involves segmenting images into their color components, as shown in previous research [14, 15]. However, as images become more complex, accurately identifying meaningful components becomes challenging. To address this problem, we propose SAMVG, a multi-stage image vectorization model for converting raster images into high-quality SVGs with reasonable runtime.

SAMVG utilizes Segment-Anything Model (SAM) [16] to create segmentation masks for the entire image, which are then refined using a novel filter technique. Missed regions are addressed by instructing SAM to generate additional masks. These masks are used to trace each component into Bézier curves, forming an initial SVG. The SVG is further enhanced via differentiable rendering [5] to align with the target image. Lastly, regions needing more shape primitives are identified through error map convolution. Their centers are used as prompts for SAM to generate another set of masks, which are added to the SVG before a final round of optimization.

We assessed the effectiveness of SAMVG through ex-

[♯] Both authors contributed equally to this research.
[*] Corresponding author.

periments, testing its generalization ability across various domains. In our quantitative evaluation against state-of-the-art methods, SAMVG consistently outperformed previous approaches across multiple image evaluation metrics, simultaneously demonstrating significantly improved runtime efficiency. Furthermore, qualitative comparison between outputs of SAMVG and those of other methods highlights that SAMVG excels in generating shapes that better align with the semantic information of target images.

In summary, our contributions are three fold:

- We propose SAMVG, a multi-stage image vectorization model for converting raster images into high-quality SVGs with reasonable runtime.
- We introduce a novel filter method to identify the best dense segmentation map for the entire image and propose to employ a novel convolution-based approach to identify regions requiring additional shape primitives for representation.
- Extensive experiments show our superior performance over previous methods. Qualitative analysis further shows the ability of SAMVG to generate shapes that closely match the semantic content of target images.

2. METHOD

SAMVG comprises four key stages: segmentation, filtering, tracing, and optimization, as shown in Fig. 2. These stages are integral to the image vectorization process, and we provide a brief overview of each:

(1) Retrieving Segmentation Masks. Given a target image $I \in \mathbb{R}^{3\times w\times h}$, SAM generates a list of segmentation masks $m_1, ..., m_n \in \mathbb{R}^{w\times h}$. To boost segmentation quality, we locate the centers of missing components for a second round of prompts. We then filter out redundant masks and sort the remaining ones by area. **(2) Approximate Tracing.** For each component in the masks, the approximate shapes of the mask are traced with Bézier curves to produce an initial SVG denoted by S. **(3) Optimization.** We denote the rasterized image of S as $I' = R(S)$. S is optimized through a specified number of iterations, utilizing mean square error (MSE) loss with respect to the target image I and incorporating additional regularization losses to enhance shape smoothness and minimize artifacts. **(4) Identifying Missing Components.** We use convolution with a circular kernel on the difference map between I' and I to detect any missing components. If such components are found, we add them to S by repeating steps 1 and 2. Finally, the resulting SVG undergoes optimization to produce the ultimate result.

Fig. 1 visually shows the intermediate results of each step in the SAMVG image vectorization process. Subsequent sections offer detailed explanations of each step, providing a comprehensive overview of the SAMVG and its components.

2.1. Stage 1: Retrieving Segmentation Masks

In the initial segmentation stage of SAMVG, we use the Automatic Masks Generator (AMG) from SAM. AMG is

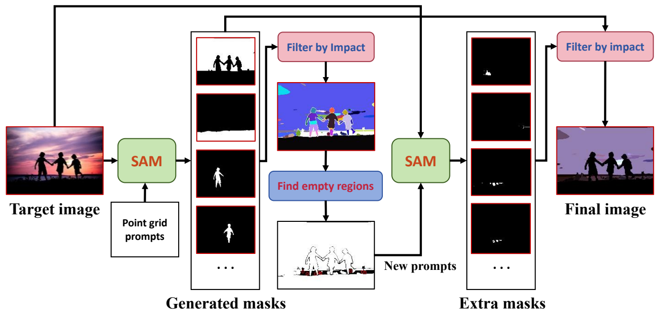

Fig. 2. Flow diagram of the first stage of SAMVG to retrieve high quality segmentation masks. We filter the masks by testing its impact on the image render, and prompt the model twice for any components missed.

prompted with a customizable 32×32 grid of points, along with SAM's test-time augmentation stage in which multiple overlapping zoomed-in image crops are included for comprehensive segmentation. It produces a list of masks, which are then filtered based on the confidence score and intersection over union (IoU). Post-processing removes small components and holes to aid path tracing in later stages.

The list of masks generated by AMG serves as a starting point, but it may not be sufficient for immediate tracing. Depending on the filter threshold and image complexity, AMG can predict either too few masks, leaving large areas uncovered or too many masks leading to redundancy (see Fig. 3). To address this, we introduce a filtering method called "Filter by Impact" in addition to existing filters to ensure that all retained masks are both correct and significant. Additionally, we employ convolution with a circular kernel to identify potential center points in large uncovered regions, which are then used as prompts to generate additional segmentations.

In the following sub-sections, we provide detailed explanations of the filtering masks and the process of finding uncovered regions for prompts.

2.1.1. Filter by Impact

To filter undesirable masks, we assess their impact by rendering them on a canvas. We begin with a blank canvas $C_0 \in \mathbb{R}^{3\times w\times h}$ and sort masks $m_1, ..., m_n \in \{0, 1\}^{w\times h}$ by area in descending order. Each mask m_i is assigned a color $c_i \in \mathbb{R}^3$ by averaging the regions it covers on I. We sequentially add mask m_i to the canvas as follows:

$$C_i = f(C_{i-1}, m_i, c_i), \tag{1}$$

where $f\colon \mathbb{R}^{3\times w\times h} \to \mathbb{R}^{3\times w\times h}$ sets the color of pixels in C_{i-1} covered by m_i as c_i. Then we calculate the normalized mean square error. We ensure that pixels not covered by any masks on C have the maximum error to prevent bias toward bright pixels. The impact γ_i of m_i is defined as the difference between the new and previous error:

$$\begin{gathered} e_i = \tfrac{||I - C_i||_2}{MaskArea}, \\ \gamma_i = e_i - e_{i-1}. \end{gathered} \tag{2}$$

If γ_i falls below a predefined threshold, we discard m_i and revert C_i to the previous state C_{i-1}.

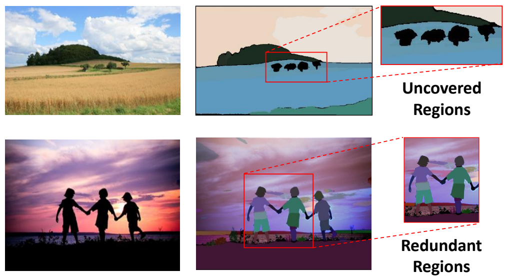

Fig. 3. Examples of failure cases from the Automatic Masks Generator provided in SAM.

The “Filter by Impact” method in SAMVG offers several advantages. It retains masks representing sub-parts of objects with significant intersection over union (IoU) with parent object masks while filtering out small or incorrect masks. This ensures that all generated masks contribute significantly to the image representation. The filtering process is applied iteratively whenever SAM generates new masks, minimizing redundancy and enhancing vectorization efficiency by maintaining a list of relevant masks.

2.1.2. Locating Uncovered Regions

To identify large uncovered regions in C, we first generate an alpha mask $C_\alpha \in \mathbb{R}^{w\times h} = \vee_{i=1}^{n} m_i$, where $\vee$ stands for the bitwise-OR-operator. Convolution is then performed on C_α with a fixed circular kernel $k \in \{0,1\}^{r\times r}$ to get C'_α, where

$$k_{i,j} = \begin{cases} 1, & \text{if } \sqrt{\left(i-\frac{r}{2}\right)^2 + \left(j-\frac{r}{2}\right)^2} \leq r \\ 0, & \text{otherwise} \end{cases} . \qquad (3)$$

We identify candidate points by selecting the coordinates in C'_α with zero values. Convolution identifies points belonging to large uncovered regions within a circle, whose radius is determined as a fraction of the image size. Subsequently, we employ mean shift clustering [17] on these candidate points to generate prompts for SAM. After filtering, we obtain a final list of masks $m_1, ..., m_n$ ready for tracing.

2.2. Stage 2&3: Approximate Tracing and Optimization

Our tracing algorithm is similar to [18]. However, instead of choosing corner points based on local maxima, we select the global maximum as the first corner point. We then remove nearby points from potential corner candidates and search for the next corner point among the remaining candidates. This process is repeated until a predetermined number of corner points is reached. This modification aims to maintain a fixed number of segments in each path for simplicity and comparability with baselines that have a fixed segment count.

Following the tracing stage, we generate an initial SVG, providing an excellent starting point for optimization. Leveraging differentiable rendering [5], we directly optimize the SVG parameters. The primary loss function employed is the MSE loss and LPIPS loss [19] calculated between the rendered and target image. Additionally, we incorporate the Xing loss introduced by [4], which penalizes shapes prone to self-interaction and encourages topologically sound shapes.

Metrics	LIVE [4]	DIFFVG [5]	SAMVG
MSE$^{\times 10^{-3}}$ $\downarrow$	5.75	16.1	**4.89**
LPIPS $\downarrow$	0.259	0.318	**0.243**
FID $\downarrow$	188.54	209.25	**184.24**
Complexity $\downarrow$	11.53	12.17	**11.32**
Paths $\downarrow$	59.96	59.96	**57.54**
Num of Parameters $\downarrow$	2038	2038	**1956**
Time (s) $\downarrow$	2609.00	139.57	**139.13**

Table 1. Quantitative results on the self-collected dataset, Complexity is proposed by [20] and is calculated on how well images can be compressed by JPEG [21] standard.

The total loss function can be formulated as:

$$L = L_{MSE} + \lambda_{Xing}L_{Xing} + \lambda_{LPIPS}L_{LPIPS}, \qquad (4)$$

where $\lambda_{MSE}, \lambda_{Xing}, \lambda_{LPIPS}$ are hyper-parameters.

2.3. Stage 4: Identifying Missing Components

After the first phase of optimization, SAMVG may still miss some semantically significant components if their size is too small. These components may carry semantic meanings that are crucial for human perception of the image. For example, SAMVG occasionally fails to display eyes when vectorizing portrait images due to their small size. However, when prompted at the correct location, SAM can provide appropriate segmentations for these components.

To address these missed components at the end of the first optimization phase, we detect them by convolving the difference map. We compute the difference map D by summing across color channels using the target image I and the current render I'. We then apply convolution to D with a fixed circular kernel, as described in Sec 2.1.2, to eliminate noise, resulting in D_1. Subsequently, we apply thresholding to D_1, ensuring that:

$$\boldsymbol{D}_{2_{x,y}} = \begin{cases} 1, & \text{if } \boldsymbol{D}_{1_{x,y}} \geq \omega \\ 0, & \text{if } \boldsymbol{D}_{1_{x,y}} < \omega \end{cases}, \qquad (5)$$

where ω is the threshold value experimentally determined to be 0.784. Following this, we use the centers of components, whose value equals 1, in D_2 as prompts to SAM to retrieve the masks. Finally, the masks are filtered by impact with respect to I' as the starting canvas, then traced and optimized for another 500 iterations to achieve the final SVG.

3. EXPERIMENT

3.1. Experiment Setup

We use the Adam optimizer for all experiments, conducted on an Nvidia GeForce RTX 3090 GPU. The learning rates are 0.01 for color parameters and 1 for point parameters in all algorithms. All algorithms run for 1000 iterations. To ensure fairness, we optimize the parameters for 1000 iterations for all methods. We assess methods on a self-collected dataset, comprising 120 images from various categories.

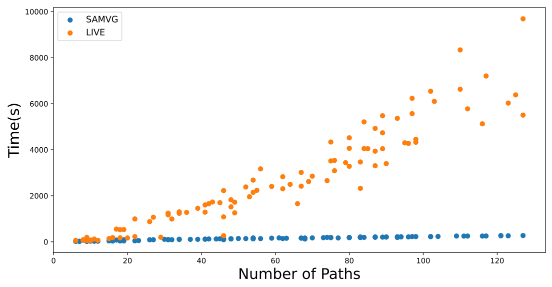

Fig. 4. Time taken by each algorithm vs. number of paths.

Metrics	SAMVG	SAMVG w/o Filter
$\text{MSE}^{\times 10^{-3}} \downarrow$	4.89	**4.77**
LPIPS $\downarrow$	0.243	**0.238**
FID $\downarrow$	**184.24**	230.64
Complexity $\downarrow$	**11.32**	11.38
Paths $\downarrow$	**57.54**	202.95
Num of Parameters $\downarrow$	**1956**	6696
Time (s) $\downarrow$	**142.00**	563.96

Table 2. Ablation Experiment on Filter by Impact.

3.2. Quantitative Results

From Tab. 1, we can conclude that SAMVG outperforms LIVE and DIFFVG in all vectorization quality measures, including MSE, LPIPS, and FID. Besides, SAMVG is considerably faster than LIVE and offers significantly better vectorization quality than DIFFVG. Notably, the speed advantage of SAMVG becomes more pronounced for larger and more complex images, as demonstrated in Fig. 4, making it suitable for applications prioritizing fast and efficient vectorization.

3.3. Qualitative Analysis

Fig. 5 reveals that SAMVG generates vector graphics that closely resemble the target image. LIVE produces cleaner graphics with fewer extraneous shapes, but DiffVG generates patchy colors due to random initial path distribution. While the difference in vectorization quality between LIVE and SAMVG is subtle and subjective, SAMVG excels in terms of speed and the retrieval of semantic features from the target image. This ability to extract semantic features is a key advantage of SAMVG, enabling more accurate and meaningful image representations compared to other methods.

3.4. Ablation Study

To evaluate the efficacy of the proposed filter technique, **Filter by Impact**, we proceed to remove it from all stages of SAMVG and conduct an experiment using the same dataset. As observed in Table 2, it becomes evident that the removal of Filter by Impact results in a substantial increase in path count, number of SVG parameters, and the processing time. Notably, this substantial change in metrics does not yield a commensurate enhancement in image quality, as discerned from

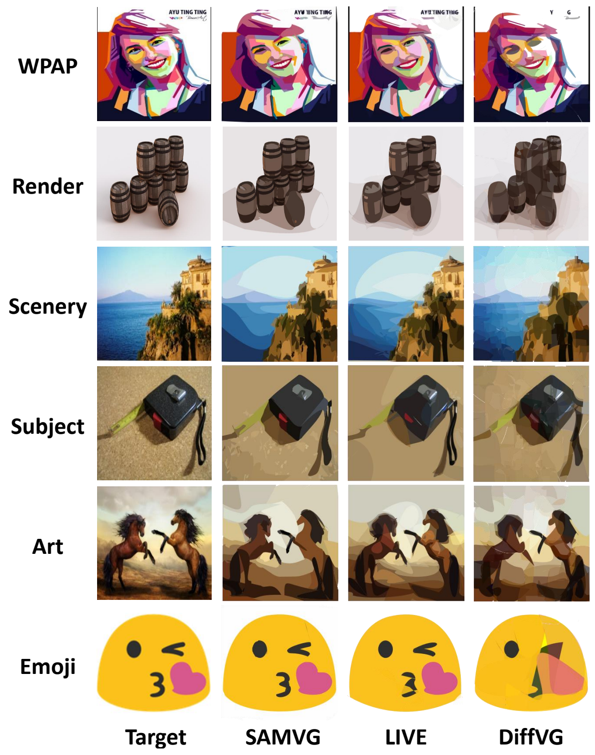

Fig. 5. Qualitative Comparisons between SAMVG, DIFFVG and LIVE.

the metrics reflecting image quality, namely MSE, LPIPS, FID and Complexity, which exhibit minimal change. In light of these findings, we can infer that the proposed Filter by Impact method effectively improve the efficiency of the image vectorization process without compromising image quality.

4. CONCLUSION

In this work, we present a novel multi-stage image vectorization model, SAMVG, which combines deep learning segmentation technology with traditional image vectorization. With the novel filter method, Filter by Impact, and a coarse-to-fine framework, SAMVG exhibits the remarkable capability to produce SVGs of exceptional quality at a significantly enhanced rate. Through abundant experiments, we demonstrate that the vectorization quality of SAMVG is superior to the previous state-of-the-art methods while concurrently exhibiting enhanced operational efficiency.

5. ACKNOWLEDGEMENT

This work was supported by National Natural Science Foundation of China (62302297, 72192821, 62272447), Young Elite Scientists Sponsorship Program by CAST (2022QNRC 001), Shanghai Sailing Program (22YF1420300), Beijing Natural Science Foundation (L222117), the Fundamental Research Funds for the Central Universities (YG2023QNB17), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), Shanghai Science and Technology Commission (21511101200).

6. REFERENCES

[1] Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum, "Image vectorization using optimized gradient meshes," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, pp. 11–es, 2007.

[2] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin, "Diffusion curves: a vector representation for smooth-shaded images," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, pp. 1–8, 2008.

[3] Guofu Xie, Xin Sun, Xin Tong, and Derek Nowrouzezahrai, "Hierarchical diffusion curves for accurate automatic image vectorization," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 6, pp. 1–11, 2014.

[4] Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi, "Towards layer-wise image vectorization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16314–16323.

[5] Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley, "Differentiable vector graphics rasterization for editing and learning," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.

[6] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte, "DeepSVG: A hierarchical generative network for vector graphics animation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16351–16361, 2020.

[7] Vage Egiazarian, Oleg Voynov, Alexey Artemov, Denis Volkhonskiy, Aleksandr Safin, Maria Taktasheva, Denis Zorin, and Evgeny Burnaev, "Deep vectorization of technical drawings," in *European Conference on Computer Vision*. Springer, 2020, pp. 582–598.

[8] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens, "A learned representation for scalable vector graphics," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7930–7939.

[9] I-Chao Shen and Bing-Yu Chen, "ClipGen: A deep generative model for clipart vectorization and synthesis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 12, pp. 4211–4224, 2021.

[10] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J Mitra, "Im2Vec: Synthesizing vector graphics without vector supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7342–7351.

[11] Teng Hu, Ran Yi, Haokun Zhu, Liang Liu, Jinlong Peng, Yabiao Wang, Chengjie Wang, and Lizhuang Ma, "Stroke-based neural painting and stylization with dynamically predicted painting region," *arXiv preprint arXiv:2309.03504*, 2023.

[12] Hao Su, Jianwei Niu, Xuefeng Liu, Jiahe Cui, and Ji Wan, "Vectorization of raster manga by deep reinforcement learning," *arXiv preprint arXiv:2110.04830*, 2021.

[13] Ming Yang, Hongyang Chao, Chi Zhang, Jun Guo, Lu Yuan, and Jian Sun, "Effective clipart image vectorization through direct optimization of bezigons," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 2, pp. 1063–1075, 2015.

[14] James Richard Diebel, *Bayesian Image Vectorization: the probabilistic inversion of vector image rasterization*, Ph.D. thesis, Stanford University, 2008.

[15] Peter Selinger, "Potrace: a polygon-based tracing algorithm," 2003.

[16] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al., "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[17] Yizong Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995.

[18] Muhammad Sarfraz and Murtaza Khan, "An automatic algorithm for approximating boundary of bitmap characters," *Future Generation Computer Systems*, vol. 20, no. 8, pp. 1327–1336, 2004.

[19] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 586–595.

[20] Penousal Machado and Amílcar Cardoso, "Computing aesthetics," in *Brazilian Symposium on Artificial Intelligence*. Springer, 1998, pp. 219–228.

[21] Gregory K Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, 1991.