Clair Obscur: an Illumination-Aware Method for Real-World Image Vectorization

Supplementary Material

7. Quantitative Results for Layer-Wise Representation

To assess semantic layer-wise representation, we provide additional comparisons using the Vector Compactness (VeC) [35] and CLIPScore metrics on the Face and Scene datasets, each containing 100 images.

VeC assesses how well the primitives are contained within the semantic object boundaries. VeC is computed by randomly sampling four semantic masks per image and measuring vectors with over 85% area overlap. Table 1 shows our method achieves higher VeC scores, indicating better containment of primitives within semantic object boundaries.

Table 1. VeC (%) comparison on Face and Scene datasets.

Dataset	VeC (%)	DiffVG	LIVE	O&R	LayerVec	Ours
Face	Avg.	48.3	51.9	49.3	<u>55.4</u>	**65.2**↑
	Std.	11.7	11.5	12.3	10.8	8.6
Scene	Avg.	53.4	55.6	52.8	<u>57.6</u>	**66.2**↑
	Std.	12.8	12.2	12.8	11.1	9.2

To quantify semantic alignment between the rendered SVG images and the original image content, we use CLIPScore as a metric of cross-modal semantic similarity. We first generate captions for each target image using BLIP, and then compute the CLIP similarity between the rendered SVG images and the corresponding target captions. In Figure 15, our method achieves competitive CLIP scores across path budgets, demonstrating stable layer-wise semantic alignment.

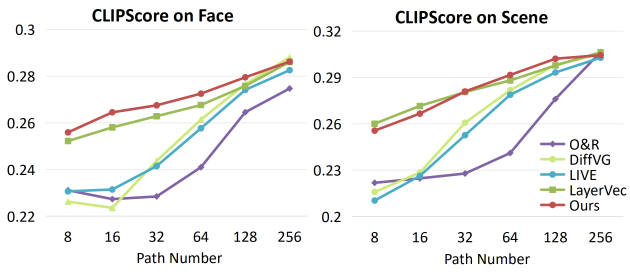


Figure 15. Comparison of CLIP similarity score.

8. Blending Mode of SVG

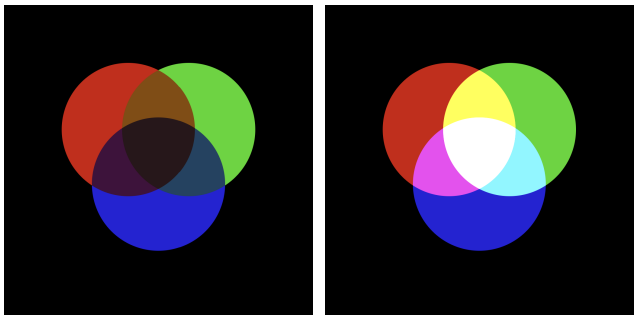
Given the decomposition in Eq. 2, the albedo, shade, and light layers must be composited into a unified SVG representation to produce the final output. We use the native blending modes defined in the SVG specification to implement the image formation model:

$$\mathcal{V}_{\text{final}} = \mathcal{V}_A * \mathcal{V}_S + \mathcal{V}_L.$$

Here, the element-wise multiplication between the albedo and shade layers is implemented with the `multiply` blending mode, while the additive light term is implemented using the `plus-lighter` blending mode.

The `multiply` mode performs channel-wise multiplication between the two overlapping layers, producing darker tones in intersecting regions. This behavior closely matches the attenuation effect of shading in intrinsic image decomposition, allowing the shade layer to modulate surface reflectance without altering the underlying albedo structure. In contrast, the `plus-lighter` mode performs linear additive blending and is therefore well aligned with modeling highlights or residual illumination. This operator preserves high-frequency lighting cues and naturally accumulates light intensities in overlapping areas.

Figure 16 provides a visual illustration of these two blending modes. By leveraging SVG's native compositing operators, our method achieves a fully vectorized representation whose rendering behavior directly corresponds to the intrinsic image formation model, while maintaining resolution-independence, editability, and semantic interpretability of individual layers.



(a) Multiply Blending. (b) Plus-lighter Blending.

Figure 16. Illustration of different SVG blending modes: multiply and plus-lighter.

9. Details of Layer Separation

After illumination refinement, the illumination layer $\mathcal{V}_I$ is separated into the shade layer $\mathcal{V}_S$ and the light layer

$\mathcal{V}_L$. The separation is performed by examining each vector path's fill color intensity. Let each path in $\mathcal{V}_I$ be represented as

$$\theta_n^I = \{P_n^I,\, C_n^I,\, \tau_n^I\},$$

where P_n^I denotes the geometric shape, $C_n^I \in \mathbb{R}^3$ is the RGB fill color, and τ_n^I is the opacity. A path is assigned to either the shade or light layer according to the normalized intensity condition:

$$\theta_n^S = \begin{cases} \theta_n^I, & \|C_n^I\|_\infty \le 1, \\ \varnothing, & \text{otherwise}, \end{cases}$$

$$\theta_n^L = \begin{cases} \varnothing, & \|C_n^I\|_\infty \le 1, \\ \{P_n^I, C_n^L, 1\}, & \text{otherwise}, \end{cases}$$

where $\|C_n^I\|_\infty = \max(C_n^I)$ measures per-path color intensity. Paths whose colors remain in the normalized range $[0, 1]$ are therefore grouped into $\mathcal{V}_S$, while those exceeding this range are treated as highlight-contributing and grouped into $\mathcal{V}_L$. During this process, all control points remain unchanged: shade-layer paths inherit both shapes and colors from $\mathcal{V}_I$, whereas light-layer paths inherit only the shapes.

Because the light layer uses additive blending, colors for paths in $\mathcal{V}_L$ cannot be taken directly from $\mathcal{V}_I$. Instead, they are recomputed from the residual illumination, defined as

$$\mathcal{R}_{\text{res}} = \mathcal{I} - \mathcal{R}(\mathcal{V}_A) * \mathcal{R}(\mathcal{V}_S),$$

where $\mathcal{R}(\cdot)$ denotes differentiable vector rendering. The final color for each light-layer path is obtained by averaging the residual illumination within its covered region:

$$C_n^L = \frac{1}{|P_n^I|} \sum_{x \in P_n^I} \mathcal{R}_{\text{res}}(x).$$

Finally, the shade and light layers are given by

$$\mathcal{V}_S = \sum_n \theta_n^S, \qquad \mathcal{V}_L = \sum_n \theta_n^L.$$

10. Experimental Setup

10.1. Implementation Details

During the structural optimization stage, we perform a total of 100 iterations: the first 50 iterations use only the structural loss to stabilize geometric alignment, while the remaining 50 iterations switch to the reconstruction loss to enhance overall visual fidelity. During the illumination refinement stage, each newly added batch of illumination paths is optimized for 100 iterations, including both parameter refinement and redundant path pruning. The number of path-addition rounds is capped at 5 to prevent unnecessary path growth and avoid overfitting.

10.2. Model Variant for simple images.

For stylized or emoji-like images with minimal illumination variation, we adopt a simplified configuration that omits the shade–light decomposition and optimizes only the albedo layer. At initialization, we perform semantic segmentation on the input image to obtain coarse region masks. These masks are then complemented by those produced through our proposed region-wise semantic binarization strategy. The two sets of masks are concatenated to guide the initialization of vector paths and the subsequent structural optimization. After the structural optimization stage, additional paths are iteratively added and pruned solely within the albedo layer, yielding the final compact SVG representation.

10.3. Controlled Color-Editing Experiment

Given an original image $I \in \mathbb{R}^{H\times W\times 3}$, a reference edited image $\hat{I} \in \mathbb{R}^{H\times W\times 3}$, and an SVG representation $\mathcal{V} = \{\,\theta_n\,\}_{n=1}^N$ consisting of N vector paths, our controlled color-editing procedure automatically determines a set of editable paths and assigns new colors consistent with the target modification.

Edit Region Detection. To localize regions that differ between I and $\hat{I}$, we compute a per-pixel difference map:

$$D(p) = \frac{1}{3}\sum_{c=1}^{3} \left| I_c(p) - \hat{I}_c(p) \right|, \qquad p \in \Omega,$$

where Ω denotes the image domain. A binary edit mask M_{edit} is then obtained via thresholding:

$$M_{\text{edit}}(p) = \begin{cases} 1, & D(p) > \tau, \\ 0, & \text{otherwise}, \end{cases}$$

with τ a fixed scalar threshold.

Per-Path Masking and Candidate Selection. Each vector path θ_n is rendered independently to obtain a binary mask $M_n \in \{0, 1\}^{H\times W}$. We measure how well a path spatially overlaps with the edit region using IoU:

$$\text{IoU}(n) = \frac{\sum_p M_n(p) M_{\text{edit}}(p)}{\sum_p \max(M_n(p),\, M_{\text{edit}}(p))}.$$

Paths with $\text{IoU}(n) > \gamma$ (a small constant) are retained as spatially relevant candidates.

We further enforce color compatibility. Let

$$\mu_n = \frac{1}{|S_n|}\sum_{p\in S_n} I(p), \qquad \hat{\mu}_n = \frac{1}{|S_n|}\sum_{p\in S_n} \hat{I}(p),$$

where $S_n = \{p : M_n(p) = 1\}$ is the support of path n. A path is kept only if

$$\|\mu_n - \hat{\mu}_n\|_2 \le \delta,$$

ensuring that the path corresponds to a region that actually changes color in the edited image.

Among the retained paths, we sort them by $|S_n|$ (descending) and choose the top $K \in \{1, 2, 4, 8, 16\}$ paths:

$$\mathcal{S}_K = \underset{n}{\mathrm{TopK}}\, |S_n|.$$

Automatic Color Assignment. For each selected path $n \in \mathcal{S}_K$, we compute the target color:

$$C_n^{\text{ref}} = \frac{1}{|S_n|} \sum_{p \in S_n} \hat{I}(p).$$

If the vectorization method includes an illumination model with a shade image $S(p)$ (e.g., $\mathcal{V}_S$), we compute the mean shade in the same region:

$$\bar{S}_n = \frac{1}{|S_n|} \sum_{p \in S_n} S(p),$$

and derive the updated albedo color via

$$C_n^A = \frac{C_n^{\text{ref}}}{\bar{S}_n + \varepsilon},$$

where ε avoids division by zero.

For methods without illumination decomposition, we directly set

$$C_n^A = C_n^{\text{ref}}.$$

The fill color of path n is then replaced by C_n^A, producing an updated SVG:

$$\mathcal{V}' = \{\, \theta_n' \,|\, n \in \mathcal{S}_K \text{ updated},\ n \notin \mathcal{S}_K \text{ unchanged}\}.$$

11. Additional Qualitative Results

We compare COVec with four representative vectorization methods. Figure 17 shows additional reconstruction results, where our method preserves both structure and tonal detail under the same path budget. Figure 18 illustrates the editability of the SVG produced by our method. Figures 19 and 20 further visualize our layer-wise representation.

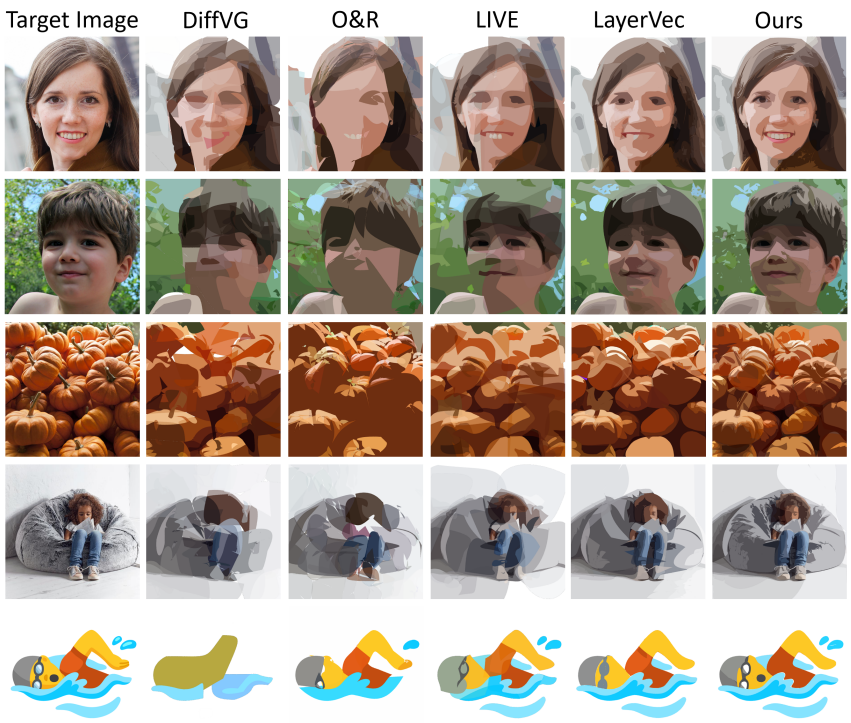


Figure 17. Qualitative reconstruction comparison across datasets. All examples in the Face and Scene datasets are vectorized using 64 paths, whereas Emoji examples are vectorized with 16 paths.

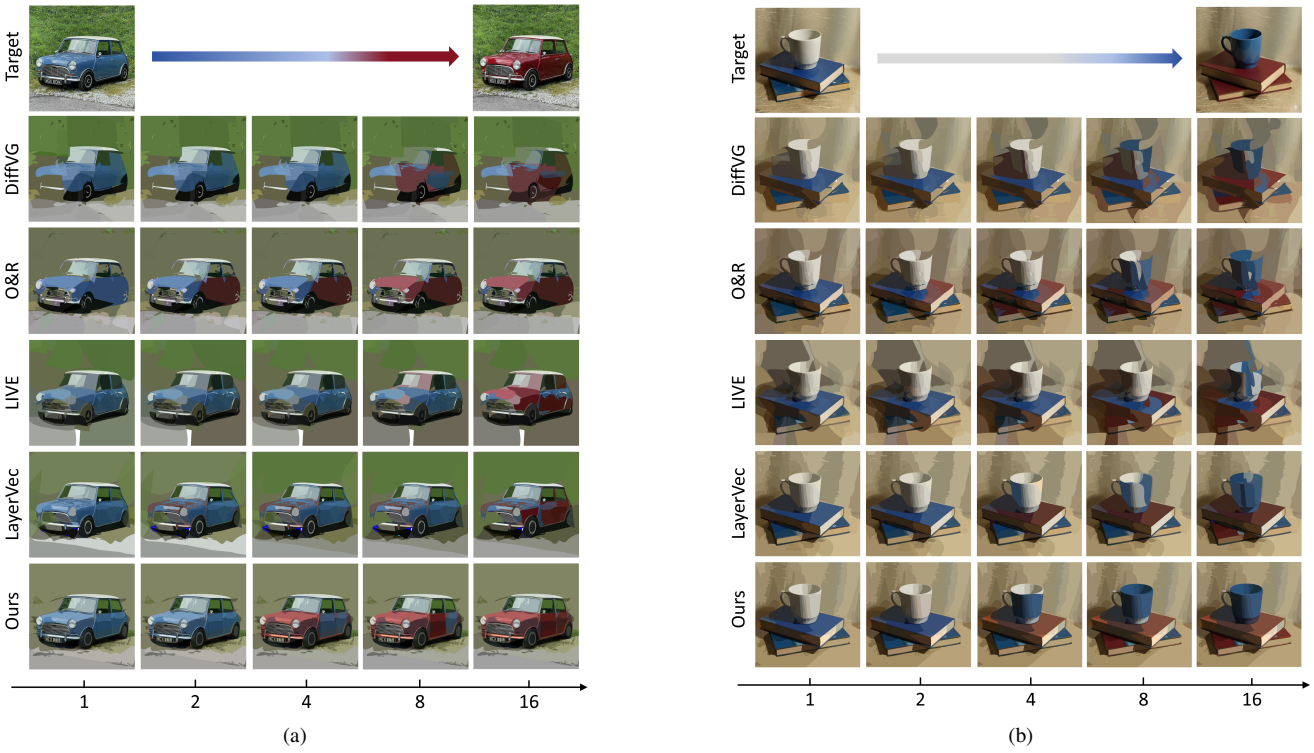


Figure 18. Color-editing comparison with controlled path modification. Each column corresponds to the number of edited paths.

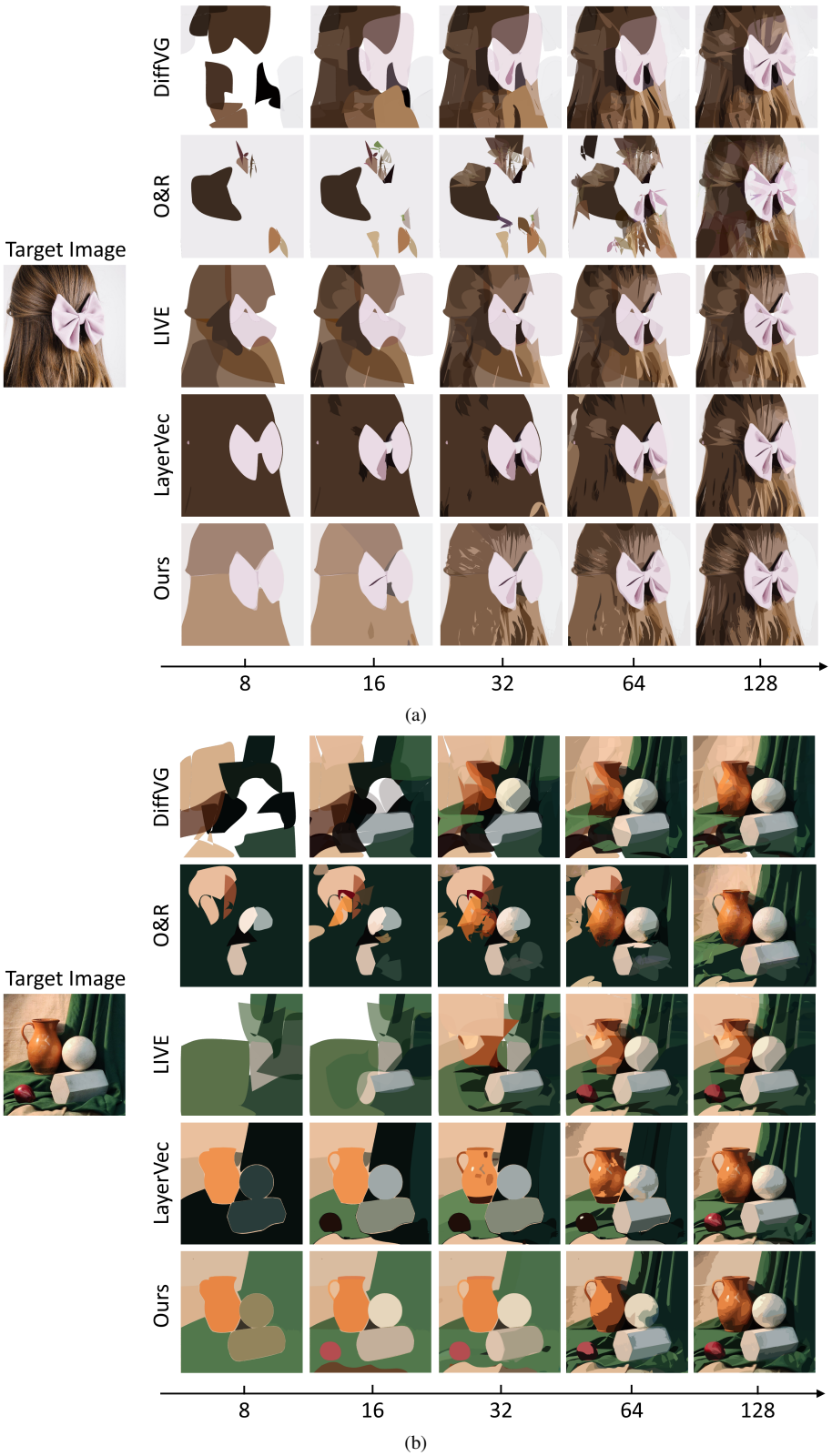


Figure 19. Comparison of layer-wise representation. Each column shows the result with an increasing number of paths, revealing how the layered representation is progressively organized.

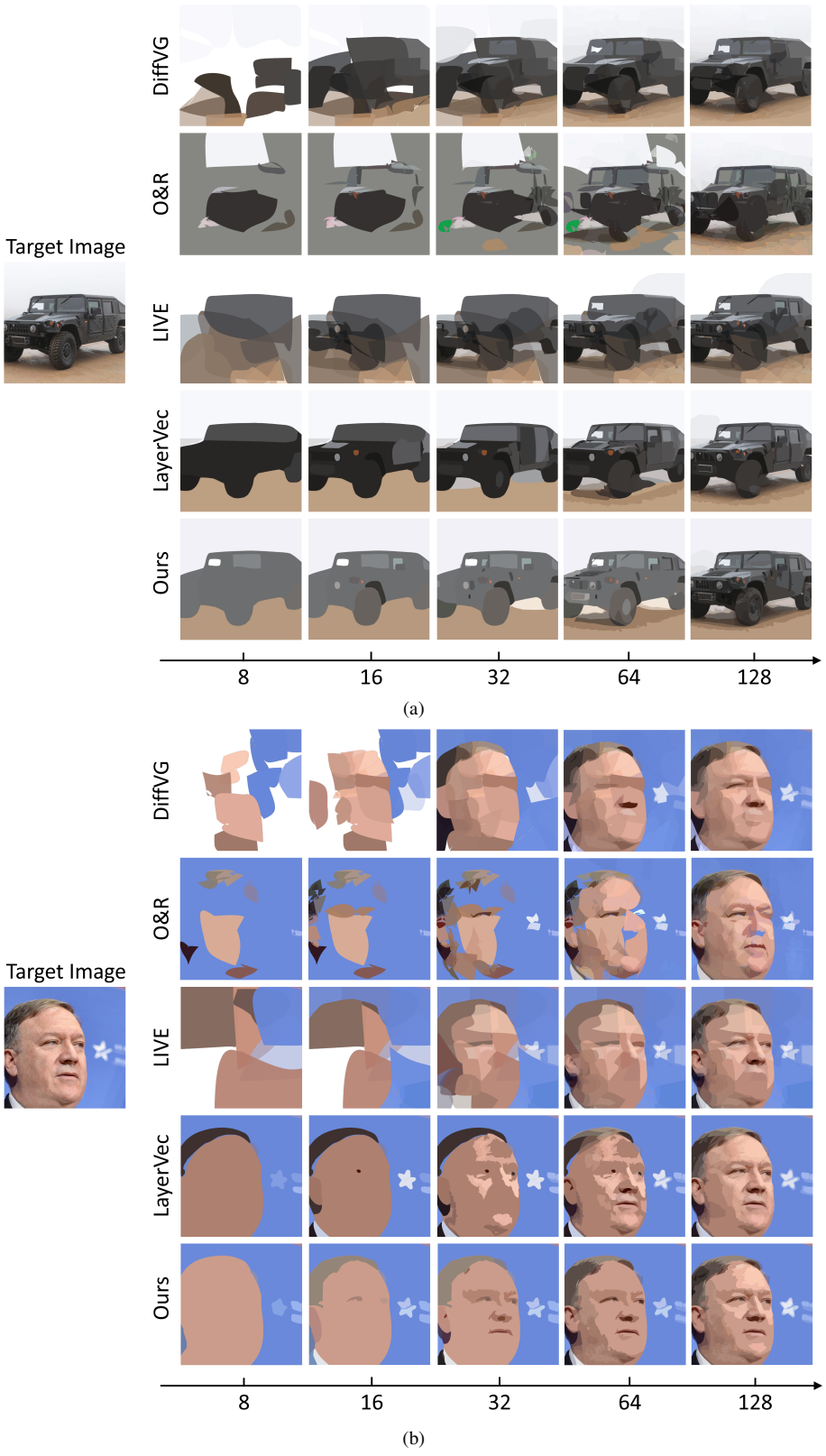


Figure 20. Comparison of layer-wise representation. Each column shows the result with an increasing number of paths, revealing how the layered representation is progressively organized.