Paper ID: 4677

Erasing Appearance Preservation in Image Smoothing

Anonymous Author(s)

Supplementary Material

In this document, we provide additional details of the model verification, additional technical backgrounds, additional implementation details, more qualitative results, more comparisons, raw user study data, and raw experimental data.

All involved tests, datasets, methods, results, and statistics are already presented in the main article, and this document only contains detailed extensions or implementations.

All EAP results in this document are achieved using our knapsack solver in absence of user interaction.

Contents

Figure 1: Texture dataset with human annotated ground truth pixel-wise texturalness measurement.

1 Additional Details for Verification

In the main article, we have modeled the EAP (Erasing Appearance Preservation) pixel-wise supporting position selection task as a knapsack problem. Here we detail our verification on Xu et al. [2012]'s dataset as presented in the main paper *(Section 2.1, line 351-354)*. Our models in the main article are:

Knapsack item value. For any pixel at position p, we have modeled item value $\boldsymbol{v}_p$ as

$$\boldsymbol{v}_p = \sum_{i \in l_p} \sum_{j \in l_p} w_{ij} ||\tau(\boldsymbol{X}_i) - \tau(\boldsymbol{Y}_j)||_2^2 \tag{1}$$

where l_p is a local window surrounding p, $\tau(\cdot)$ is CIE RGB-to-Lab transform, and $\boldsymbol{X}$ and $\boldsymbol{Y}$ are source and smoothed images respectively. Gaussian term $w_{ij} = \exp(||\boldsymbol{X}_i - \boldsymbol{Y}_j||_2^2 / 2\sigma^2)$ makes this value faithful to human perception.

Knapsack item weight. The modeling of item weight $\boldsymbol{w}_p$ is

$$\boldsymbol{w}_p = \epsilon + \sum_{i \in l_p} \sum_{j \in l_p} ||\boldsymbol{Y}_i - \boldsymbol{Y}_j||_2^2 \tag{2}$$

where ϵ is a scaler to prevent zero weights. This weight modeling is aimed at protecting desired details in images, as mentioned in the main article.

1.1 Objective

The fundamental goal of EAP knapsack is to identify pixel positions with task-specific patterns, *i.e*, discovering textural pixels for texture removal, or identifying reflectance color pixels for intrinsic decomposition, *etc*. Because knapsack algorithms are aimed at finding items with as large as possible total values and limited total weights, our verifications are presented as:

1. to verify that item value $\boldsymbol{v}_p$ is positively correlated to task-specific desired patterns.
2. to verify that item weight $\boldsymbol{w}_p$ is negatively correlated to task-specific desired patterns.

1.2 Setup

We use texture removal task to verify our modeling. In particular, we use RTV Texture Boundary (RTVDB) dataset in Xu et al. [2012] to obtain in-the-wild image instances and the paired texture annotations. Xu et al. [2012] provides 200 textured images and 200 paired human annotated structure

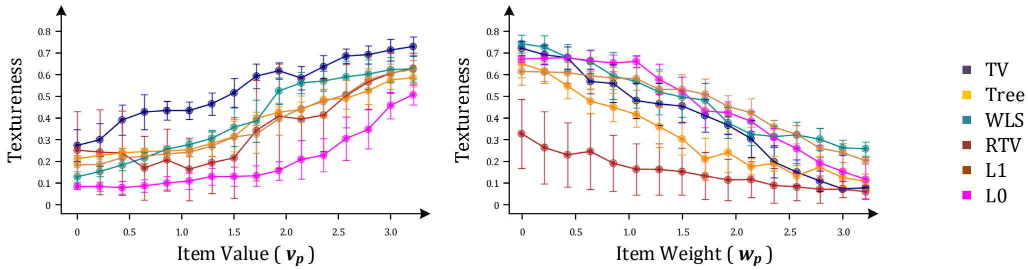

Figure 2: Correlation between our item value/weight modeling and ground truth texturalness.

boundary maps. We measure pixel-wise distance to nearest structure boundary as structuralness (and the inverted value as texturalness). As shown in Fig.1, pixels with dense and continuous texture are annotated with high texturalness (low structuralness) whereas pixels over structural edges or object boundaries are marked with low texturalness (high structuralness).

1.3 Involved image smoothing energy

We exploit various image smoothing energies (**WLS**, Welsch [1977]; **TV**, RUDIN et al. [1992]; **L0**, Xu et al. [2011]; **RTV**, Xu et al. [2012]; **TREE**, Bao et al. [2014]; **L1**, Bi et al. [2015]) in this verification.

1.4 Implementation details

Using all involved methods, we perform smoothing on all image instances in the RTVDB dataset and randomly sample one million pixel position points in all smoothed images. For each point, we obtain the measured item value, item weight, and the ground truth texturalness.

1.5 Verification

As in Fig. 2, we report the obtained correlation between item weight, item value, and texturalness. We divide all values/weights into 16 bins and then report the mean value and standard deviation in each bin for each candidate. Regardless of various frontend smoothing energies, $\boldsymbol{v}_p$ always shows positive correlation with ground truth texturalness. In the meanwhile, $\boldsymbol{w}_p$ always shows negative correlation with the texturalness. These evidences verify that our modeling has solid foundations in the texture removal tasks. Given the typicality of texture removal in image smoothing, our approach in other related tasks is likely to have similar verifications. These evidences also shows that our knapsack modeling is statistically well-motivated and technically solid to tackle specific problems.

2 Algorithmic Backgrounds: 0-1 Knapsack

The 0-1 knapsack problem is one of the most typical combinatorial problem in mathematics, statistics, and economics. Given a set of items, each with a weight and a value, the objective is to determine whether to include each item in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

Dynamic Programming (DP) 0-1 knapsack solver is one of the most typical solutions for this problem. A python version of this standard solver can be sketched as:

```python
# Python code of standard 01 knapsack dynamic programming (DP) solver.

def knapsack_01(n, c, w, v):
    value = [[0 for j in range(int(c) + 1)] for i in range(n + 1)]
    for i in range(1, n + 1):
        for j in range(1, int(c) + 1):
            value[i][j] = value[i - 1][j]
            if j >= w[i - 1] and value[i][j] < value[i - 1][int(j - w[i - 1])] + v[i - 1]:
                value[i][j] = value[i - 1][int(j - w[i - 1])] + v[i - 1]
    x = [0 for i in range(n)]
    j = c
    for i in range(n, 0, -1):
        if value[i][int(j)] > value[i - 1][int(j)]:
            x[i - 1] = 1
            j -= w[i - 1]
    return x

# Below is a unit test for this function.

knapsack_capability = 5
item_quantity = 6
test_weights = [2.4, 2.6, 3.2, 1.2, 5.7, 2.9]
test_values = [2.7, 3.2, 1.6, 5.3, 4.3, 3.1]
test_result = knapsack_01(item_quantity, knapsack_capability, test_weights, test_values)
print(test_result)
```

It is worth noticing that the time complexity of this standard solver is $O(NW)$ with N being the item quantity and W being the (integer number of) the knapsack capability. This solver may yield optimal solutions, nevertheless it may also consume hours to compute one large image with pixel quantity at about $1e8$. Therefore, it is important to apply some basic accelerations for practical usages.

2.1 Scalable 0-1 knapsack solver

The problem of scaling 0-1 Knapsack algorithm is extensively studied in the field of mathematics and statistics. We adopt a simple yet very effective solver using Stanford Greedy Knapsack Heuristic (GKH, Stanford [2001]).

The basic idea is that, instead of testing all items in or out of the knapsack, we may only compute a relatively small subset of all items so that a vast majority of the time consuming can be saved. When the knapsack capability is extremely large, the target items can be viewed as particles with the value-weight density $\boldsymbol{d}_p$

$$\boldsymbol{d}_p = \frac{\boldsymbol{v}_p}{\boldsymbol{w}_p} \tag{3}$$

For each item p, if the density $\boldsymbol{d}_p$ is very large, we can assume that this item is of great value and then directly put it into the knapsack without extra consideration. On the contrary, if the density $\boldsymbol{d}_p$ is minimal, we may directly ignore it because it may not make much contribution to our value-weight trade-offs. Finally, when the density $\boldsymbol{d}_p$ is neither too large nor too small, we apply standard knapsack algorithm to solve these mid-range items.

In particular, given the knapsack capability U, we first roughly estimate item quantity $\bar{Q}$ in the knapsack

$$\bar{Q} = \frac{U}{\bar{d}} \tag{4}$$

where $\bar{d}$ is the average density of all items. After that, we define a parameter called *interested range* denoted by $N_{\text{intersted}}$. Given a set of item densities $\boldsymbol{d}_{1...N}$ sorted from small to large, we divide them into three groups

$$\begin{aligned}&\{\boldsymbol{d}_1, ..., \boldsymbol{d}_N\} \rightarrow \\ &\{\boldsymbol{d}_1, ..., \boldsymbol{d}_{\bar{Q}-\frac{1}{2}N_{\text{intersted}}}\} + \{\boldsymbol{d}_{\bar{Q}-\frac{1}{2}N_{\text{intersted}}}, ..., \boldsymbol{d}_{\bar{Q}+\frac{1}{2}N_{\text{intersted}}}\} + \{\boldsymbol{d}_{\bar{Q}+\frac{1}{2}N_{\text{intersted}}}, ..., \boldsymbol{d}_N\}\end{aligned} \tag{5}$$

where $N_{\text{intersted}} = 200$ is a good choice for large images. All items in $\{\boldsymbol{d}_1, ..., \boldsymbol{d}_{\bar{Q}-\frac{1}{2}N_{\text{intersted}}}\}$ are directly excluded from the knapsack, and all items in $\{\boldsymbol{d}_{\bar{Q}+\frac{1}{2}N_{\text{intersted}}}, ..., \boldsymbol{d}_N\}$ are directly included in the knapsack. After that, we solve the standard 0-1 knapsack within the subset $\{\boldsymbol{d}_{\bar{Q}-\frac{1}{2}N_{\text{intersted}}}, ..., \boldsymbol{d}_{\bar{Q}+\frac{1}{2}N_{\text{intersted}}}\}$. A python version of this accelerated solver can be sketched as:

```python
# python code of Greedy Knapsack Heuristic (GKH) 01 knapsack solver.

def knapsack_01_GKH(sorted_item_indices, Q_bar, N_interested, n, c, w, v):
    x = [0 for i in range(n)]
    # exclude items with low density
    x[0: Q_bar - N_interested // 2] = 0
    # include items with high density
    x[Q_bar + N_interested // 2: n] = 1
    # analyse interested items with mid-range density
    n -= N_interested
    c = N_interested
    v = [v[sorted_item_indices[i]] for i in range(Q_bar - N_interested // 2, Q_bar + N_interested // 2)]
    w = [w[sorted_item_indices[i]] for i in range(Q_bar - N_interested // 2, Q_bar + N_interested // 2)]
    x[Q_bar - N_interested // 2: Q_bar + N_interested // 2] = knapsack_01(n, c, w, v)
    return x
```

With this improved solver, 512×512 pixels can be solved in roughly 175 ms as reported in the main article. This implementation can significantly speed up the solving, and almost no visual quality sacrifice is caused in human perception.

2.2 Stabilized solver and full implementation codes

As in the main paper, the actual knapsack problem is solved in multiple iterations. This solver can be further stabilized by introducing randomness. For example, we can add some random noise in some starting iterations and then progressively remove these noise when nearing convergence. This *simulated annealing* trick can contribute to the robustness of the solver.

Furthermore, we provide full codes of our knapsack solver implementation in the *code-and-data* files to aid in reproducibility.

3 Image Smoothing Implementation Details

We clarify detailed implementation applying EAP to different image smoothing energies.

3.1 EAP + Total Variation

The total variation (RUDIN et al. [1992]) smoothing energy can be written in form of the formulation in the main article as

$$\rho(\boldsymbol{Y}) = \lambda(|\partial_x \boldsymbol{Y}| + |\partial_y \boldsymbol{Y}|) \tag{6}$$

where $\boldsymbol{Y}$ is the image for optimization. $\partial_x \boldsymbol{Y}$ or $\partial_y \boldsymbol{Y}$ refers to image gradient in x-axis or y-axis. Total variation is relatively easy to implement and we use gradient descent method to solve the smoothing problem. Note that we have two options to apply EAP: (a) solve knapsack each time after total variation is converged, or (b) solve knapsack for each total variation minimization iteration. Nevertheless, we find these two methods yields nearly same visual results. We recommend to use (a) for implementation flexibility. We fix $\lambda = 0.1$ in experiments.

3.2 EAP + Weighted Least Square

Weighted least square (Welsch [1977]) energy can be formulated for each pixel position p as

$$\rho(\boldsymbol{Y})_p = \lambda \sum_{i \in l_p} \exp(-||\boldsymbol{Y}_p - \boldsymbol{Y}_i||/\sigma_s)||\boldsymbol{Y}_p - \boldsymbol{Y}_i||_2^2 \tag{7}$$

which is fully differentiable and can be implemented in the same way as total variation. We fix l_p as a 3×3 window, $\lambda = 0.1$, and $\sigma_s = 0.25$ in experiments.

3.3 EAP + L0 Smoothing

L0 smoothing (Xu et al. [2011]) energy can be formulated as

$$\rho(\boldsymbol{Y}) = \lambda(\#\partial_x \boldsymbol{Y} + \#\partial_y \boldsymbol{Y}) \tag{8}$$

where $\#$ is the counting metric outputting the quantity of non-zero elements in a matrix. L0 smoothing has closed-form solvers. Official L0 smoothing implementation recommends to use fast Fourier transform to preserve source image appearance, causing a bit troublesome to apply EAP to get rid of appearance preservation. Our solution is to use Xu et al. [2011]'s secondary gradient descent solver to replace Fourier transform so that EAP can be easily implemented. Note that this modification does not numerically change the results because the Fourier transform solver and the gradient descent solver yield mathematically same results as mentioned in Xu et al. [2011]. We fix $\lambda = 0.01$ in experiments.

3.4 EAP + Relative Total Variation

Relative total variation (Xu et al. [2012]) energy can be formulated as:

$$\rho(\boldsymbol{Y})_p = \lambda \left(\frac{\mathfrak{D}_x(p)}{\mathfrak{L}_x(p) + \epsilon} + \frac{\mathfrak{D}y(p)}{\mathfrak{L}_y(p) + \epsilon} \right) \tag{9}$$

where the term of $\mathfrak{D}(\cdot)$ is defined as

$$\mathfrak{D}_x(p) = \sum_{q \in l_p} g_{pq} |\partial_x \boldsymbol{Y}_q| \qquad \text{and} \qquad \mathfrak{D}_y(p) = \sum_{q \in l_p} g_{pq} |\partial_y \boldsymbol{Y}_q| \tag{10}$$

and $\mathfrak{L}(\cdot)$ is defined as

$$\mathfrak{L}_x(p) = |\sum_{q \in l_p} g_{pq} \partial_x \boldsymbol{Y}_q| \qquad \text{and} \qquad \mathfrak{L}_y(p) = |\sum_{q \in l_p} g_{pq} \partial_y \boldsymbol{Y}_q| \tag{11}$$

and g_{pq} is defined as

$$g_{pq} = \exp\left(-\frac{(p_x - q_x)^2 + (p_y - q_y)^2}{2\sigma^2} \right) \tag{12}$$

and the default parameters are 3×3 window l_p, $\sigma = 0.2$, and $\lambda = 0.015$. Official relative total variation solver is numerical, and the objective is solved iteratively in a two-step manner. We replace its original appearance approximation with EAP in the *structure extraction* step and the other step remains same.

3.5 EAP + Spanning Tree

It is worth noticing that Spanning Tree (Bao et al. [2014]) is particularly a filtering method rather than an optimization-based method. We include this method because it is typical and common-used, and it also has optimization-based versions. The formulation of optimization-based spanning tree energy is

$$\rho(\boldsymbol{Y})_p = \lambda ||\boldsymbol{Y}_p - \sum_{j \in \hbar} w_p(j) \boldsymbol{Y}_j||_2^2 \tag{13}$$

where $\hbar$ is the set of all pixel positions and $w_p(\cdot)$ is the *Spanning Tree Collaborative Weight*. The term $w_p(\cdot)$ inherently obeys the global addend constraints

$$\left\{ \sum_{j \in \hbar} w_p(j) = 1 \;\; | \;\; \forall p \in \hbar \right\} \tag{14}$$

and for more details please refer to Bao et al. [2014]. EAP can also be implemented using this smoothing energy with default parameter $\lambda = 0.1$.

3.6 EAP + L1 smoothing

L1 smoothing (Bi et al. [2015]) energy can be formulated as:

$$\rho(\boldsymbol{Y})_p = \lambda_1 (\sum_{i \in l_p} \sum_{j \in l_p} w_{ij} ||\boldsymbol{Y}_i - \boldsymbol{Y}_j||_2) + \lambda_2 (\sum_{i \in g_p} \sum_{j \in g_p} w_{ij} ||\boldsymbol{Y}_i - \boldsymbol{Y}_j||_2) \tag{15}$$

where w_{ij} is the weight for color distance in weighted CIE-Lab space. l_p is a local window at p, and g_p is a super-pixel region that contains p. We use the recommended official configurations $\lambda_1 = 20.00$, $\lambda_2 = 0.01$, and all parameters are same as official implementations. So EAP can be directly embedded to the official Split-Bregman solver.

4 Ablative Study Implementation Details

We detail the ablative study in the main paper.

4.1 Official Implementations

We present the smoothed results using official implementations of different image smoothing algorithms.

4.2 Extreme Parameter

We present the smoothed results using extreme lambda in different image smoothing algorithms, but without using EAP. In the main paper we have presented results using L1 smoothing ($\lambda = 10.0$). In the supplementary material we present results with some other configurations: (1) L0 smoothing and $\lambda = 0.1$. (2) RTV and $\lambda = 0.2$. (3) L1 smoothing and $\lambda = 5$. Please see also Fig. 6-10 for details.

4.3 Iterative Smoothing

We present the smoothed results by repeating original image smoothing algorithms multiple times (10 times, same as the our default EAP configuration), but without using EAP. In the main paper we have presented results using L1 smoothing (repeating 10 times). In the supplementary material we present results with some other configurations: (1) L0 smoothing (repeating 10 times). (2) RTV (repeating 10 times). (3) L1 smoothing (repeating 10 times). Please see also Fig. 6-10 for details.

4.4 Without Weight

We present the smoothed results without using knapsack weights $\boldsymbol{w}_p$. We set a fixed threshold (0.1) to the knapsack values $\boldsymbol{v}_p$. All pixels above this threshold are viewed as erasing positions. The results are presented in the main paper.

4.5 Meaningless Weight

We present the smoothed results by replacing all knapsack weights $\boldsymbol{w}_p$ with a constant (1.0). The results are presented in the main paper.

4.6 Without Value

We present the smoothed results without using knapsack values $\boldsymbol{v}_p$. We set a fixed threshold (0.1) to the knapsack weights $\boldsymbol{w}_p$. All pixels above this threshold are viewed as erasing positions. The results are presented in the main paper.

4.7 Meaningless Value

We present the smoothed results by replacing all knapsack value $\boldsymbol{v}_p$ with a constant (1.0). The results are presented in the main paper.

4.8 Full Method

Our proposed solution with 0-1 knapsack solving. The results are presented in the main paper.

5 Image Decomposition Implementation Details

5.1 EAP + DPGMM + L1 smoothing

Bi et al. [2015] proposes to smooth images in multiple stages. After a standard L1 smoothing, this technique applies Dirichlet Process Gaussian Mixture Model (DPGMM) to cluster all pixels into an adaptive quantity of clusters. After that, this technique smooths images again. A rough formulation could be written as

$$\rho(\boldsymbol{Y})_p = \lambda_1(\sum_{i\in l_p}\sum_{j\in l_p} w_{ij}||\boldsymbol{Y}_i-\boldsymbol{Y}_j||_2)+\lambda_2(\sum_{i\in g_p}\sum_{j\in g_p} w_{ij}||\boldsymbol{Y}_i-\boldsymbol{Y}_j||_2)+\lambda_3(\sum_{i\in c_p}\sum_{j\in c_p} w_{ij}||\boldsymbol{Y}_i-\boldsymbol{Y}_j||_2) \tag{16}$$

where c_p is a DPGMM cluster that contains pixel position p, and all other terms remain same with L1 smoothing. For more implementation details please refer to Bi et al. [2015]. This method is aimed at building up object-wise color consistency for intrinsic decomposition. This smoothing energy can be directly implemented in the same way with L1 smoothing.

5.2 EAP + Bell

Bell (Bell et al. [2014]) method is a very typical intrinsic decomposition method. Its optimization is discrete and it does not explicitly require appearance preserving. However, the appearance preservation is in particular hidden in its formulations. Many other intrinsic decomposition methods can be formulated in similar ways as Bell method. The embedding of EAP into Bell can be applied to many other intrinsic methods.

Given the image $\boldsymbol{X}$, Bell method solves a reflectance map $\boldsymbol{R}$ and a shading map $\boldsymbol{S}$ so that

$$\boldsymbol{X} = \boldsymbol{R} \odot \boldsymbol{S} \tag{17}$$

where $\odot$ is the Hadamard product. Bell's overall optimization can be written as

$$E_{\text{intrinsic}} = E_{\text{reflectance}}(\boldsymbol{R}) + E_{\text{shading}}(\boldsymbol{S}) \quad \text{s.t.} \quad \boldsymbol{X} = \boldsymbol{R} \odot \boldsymbol{S} \tag{18}$$

In Bell method, the term $E_{\text{reflectance}}$ includes *pairwise reflectance prior*, *shading smoothness prior*, *absolute shading intensity prior*, and E_{shading} includes *shading discontinuity prior*. For detailed formulations, please refer to Bell et al. [2014]. Among all involved prior constraints, no explicit appearance preservation can be found. However, Bell's appearance preservation is in particular hidden in their priors.

Taking their *shading discontinuity prior* as an example, which can be written as

$$e_{\text{discontinuity}} = \sum_{(i,j)\in B} |\log \boldsymbol{S}_i - \log \boldsymbol{S}_j| \tag{19}$$

where B is a set of many pixel position pairs (i, j), and in each pair, i and j are two adjacent pixel positions. We can transform this constraint into

$$\begin{aligned} e_{\text{discontinuity}} &= \sum_{(i,j)\in B} |\log \frac{\boldsymbol{X}_i}{\boldsymbol{R}_i} - \log \frac{\boldsymbol{X}_j}{\boldsymbol{R}_j}| \\ &= \sum_{(i,j)\in B} |\log \boldsymbol{X}_i - \log \boldsymbol{R}_i - \log \boldsymbol{X}_j + \log \boldsymbol{R}_j| \\ &= \sum_{(i,j)\in B} |(\log \boldsymbol{X}_i - \log \boldsymbol{X}_j) - (\log \boldsymbol{R}_i - \log \boldsymbol{R}_j)| \\ &= \sum_{(i,j)\in B} |\partial_i \log \boldsymbol{X}_i - \partial_i \log \boldsymbol{R}_i| \end{aligned} \tag{20}$$

According to Poisson Perez et al. [2003] opinion, when the sampled points in B are dense enough, the approximation of image gradients can be viewed as that for the definite integral of image intensity

$$\begin{aligned} \int_0^i e_{\text{discontinuity}} &= \int_0^i \sum_{(i,j)\in B} |\partial_i \log \boldsymbol{X}_i - \partial_i \log \boldsymbol{R}_i| \\ &= \sum_{(i,j)\in B} \int_0^i |\partial_i \log \boldsymbol{X}_i - \partial_i \log \boldsymbol{R}_i| \\ &\approx \sum_{(i,j)\in B} |\log \boldsymbol{X}_i - \log \boldsymbol{R}_i| \end{aligned} \tag{21}$$

That is to say, Bell's *shading discontinuity prior* is in particular mathematically consistent to the log-space appearance preservation between source image $\boldsymbol{X}$ and smoothed reflectance $\boldsymbol{R}$.

Similarly, Bell's *shading smoothness prior* can be written as

$$\begin{aligned} e_{\text{shading\_smoothness}} &= \sum_i (\log \boldsymbol{S}_i - \log \boldsymbol{S}_{i+1})^2 \\ &= \sum_i (\log \frac{\boldsymbol{X}_i}{\boldsymbol{R}_i} - \log \frac{\boldsymbol{X}_{i+1}}{\boldsymbol{R}_{i+1}})^2 \\ &= \sum_i ((\log \boldsymbol{X}_i - \log \boldsymbol{X}_{i+1}) - (\log \boldsymbol{R}_i - \log \boldsymbol{R}_{i+1}))^2 \\ &= \sum_i (\partial_i \log \boldsymbol{X}_i - \partial_i \log \boldsymbol{R}_i)^2 \\ \int_0^i e_{\text{shading\_smoothness}} &\approx \sum_i (\log \boldsymbol{X}_i - \log \boldsymbol{R}_i)^2 \end{aligned} \tag{22}$$

where the *shading discontinuity prior* can also be written as log-space appearance preservation.

Finally, Bell's *absolute shading intensity prior* can also be transformed

$$\begin{aligned} e_{\text{absolute\_shading\_intensity}} &= \sum_i |\log \boldsymbol{S}_i - \log \bar{\boldsymbol{S}}| \\ &= \sum_i |\log \boldsymbol{X}_i - \log \boldsymbol{R}_i - \log \bar{\boldsymbol{S}}| \\ &= \sum_i |\log \boldsymbol{X}_i - \log \boldsymbol{R}_i \cdot \bar{\boldsymbol{S}}| \end{aligned} \tag{23}$$

where $\bar{\boldsymbol{S}}$ is the average value of $\boldsymbol{S}$. That is to say, the *absolute shading intensity prior* is transformed into log-space appearance preservation between the scaled source $\boldsymbol{X}$ and smoothed reflectance $\boldsymbol{R}$.

In particular, we apply EAP to all these appearance preservation terms. We compute the 0-1 knapsack using source image $\boldsymbol{X}$ and smoothed reflectance image $\boldsymbol{R}$, and then replace the three mentioned priors with their EAP versions.

Many other intrinsic methods can be implemented in the same way by finding the hidden appearance preservation in their priors and then apply EAP to getting rid of their appearance preservation terms.

Besides, the original Bell method recommends to only compute reflectance intensity whereas ignore the reflectance hue/saturation (or called chroma). We obey this principle in all quantitative comparisons to ensure the fairness and that its performance is consistent to previous literature. However, we also find that its image editing usability and capability can be improved if it computes all reflectance channels. Therefore, in our human-involved interactive user study, we use the Bell to compute all reflectance channels to make the evaluation more perception-based.

5.3 EAP + Advanced image decomposition

Carroll et al. [2011] proposes to decompose illumination maps given reflectance maps and appearance maps. In main paper we directly view this method as a black-box and it is not necessary to make more modifications.

One important notice is that this technique is in particular an interactive application. Users can use scribbles to control the decomposed layers. However, all results in the our work are obtained automatically without any extra interaction.

6 Image Manipulation Implementation Details

Given a set of image editing layers, *i.e*, Adobe PhotoShop Layers, we here clarify some terminologies mentioned in the main article. We illustrate some examples in Fig. 3 and Fig. 4.

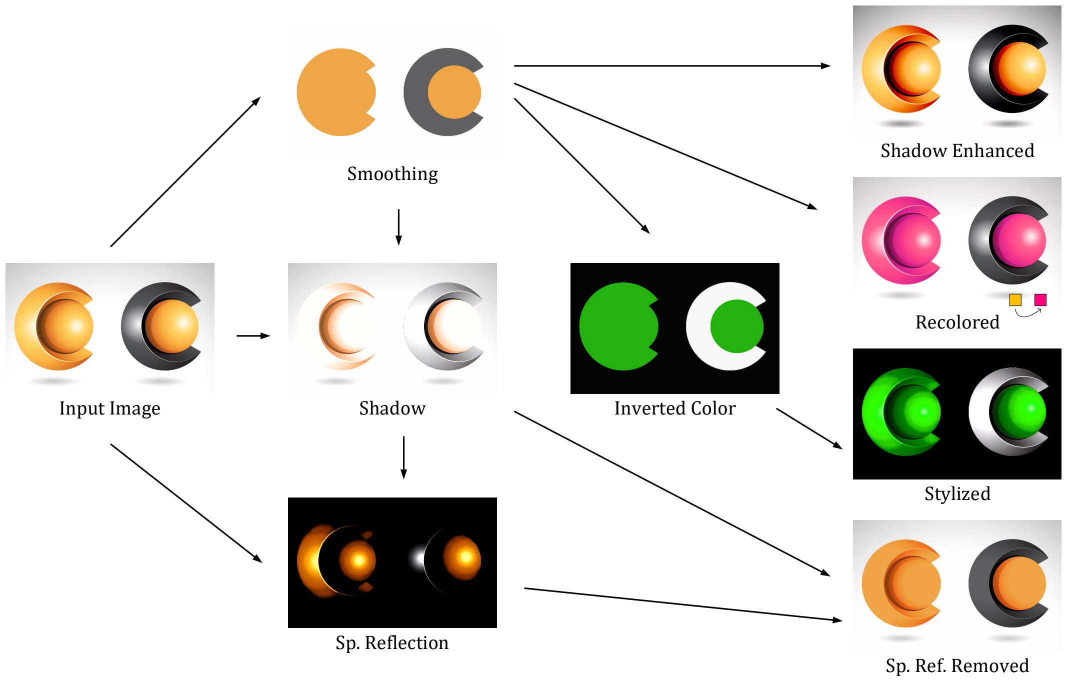

Figure 3: Toy examples of involved image editing workflows in our experiments.

6.1 Layer removal or reordering

Users may directly remove any layer or change the order of some layers. These are fundamental use cases of layer-based image editing.

6.2 Color inverting

Users may also invert the layer colors. This is useful when the objective is to recoloring reflectance layers or invert illumination conditions, *i.e*, black-to-white object recoloring, day-to-night relighting, *etc*.

6.3 Curves tuning and Exposure/Gamma correction

Users may tune the intensity curves of any specific layers. In particular, exposure tuning and gamma correction are frequently used cases. Given user-specified exposure scaler k and gamma scaler g, the mapping can be sketched as

$$y = (kx)^g \tag{24}$$

where y is output intensity and x is input intensity. Note that this transform can be applied to any single layer, multiple selected layers, or all layers.

6.4 Masking

Because EAP smoothing method yields highly structured color maps, we can use very simple transforms to obtain high-quality masks. In particular, given a smoothed structure $\boldsymbol{Y}$ and a user-given source color $\boldsymbol{c}_s \in \mathbb{R}^3$, a mask $\boldsymbol{M}$ can be measured as

$$\boldsymbol{M}_p = \lambda ||\boldsymbol{Y}_p - \boldsymbol{c}_s||_2^2 \tag{25}$$

where λ is a scaler to normalize this mask. Note that this mask is not binarized, and it is a soft segmentation of the source image. This mask can be applied to many use cases to separately editing image constitutes. Because $\boldsymbol{c}_s$ can be sampled with only one click in images, it is very flexible to obtain the mask $\boldsymbol{M}$.

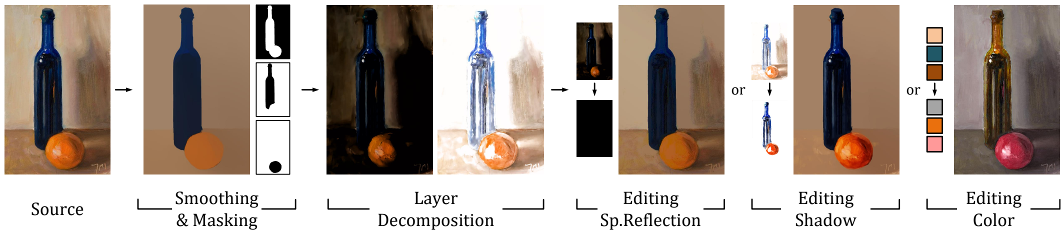

Figure 4: Typical image manipulation using strategies involved in our image editing experiments.

6.5 Recoloring

Given source color $\boldsymbol{c}_s \in \mathbb{R}^3$ and target color $\boldsymbol{c}_t \in \mathbb{R}^3$, a directional color vector $\boldsymbol{c}_v$ can be written as

$$\boldsymbol{c}_v = \boldsymbol{c}_t - \boldsymbol{c}_s \tag{26}$$

Then, using the mask $\boldsymbol{M}$, users may directly manipulate the appearance colors in image $\boldsymbol{X}$

$$\boldsymbol{X}_p^* = \boldsymbol{M}_p \odot \boldsymbol{c}_v + \boldsymbol{X}_p \tag{27}$$

where $\odot$ is the Hadamard product, and $\boldsymbol{X}^*$ is the edited image. Note that this formulation is only a basic and simple recoloring maneuver and advanced image recoloring is extensive studied in methods like Carroll et al. [2011]; Tan et al. [2016, 2018]. Finally, this recoloring can be applied to any single or multiple decomposed layers.

7 IIW/SAW Test Implementation Details

One notice is that some intrinsic methods output single-channel gray-scale illumination maps whereas some others yield colorful illumination maps. Because IIW/SAW is focused on single-channel illumination, we convert all available illumination maps into gray-scale in IIW/SAW tests.

7.1 Intrinsic Image in the Wild

As to IIW Bell et al. [2014] test, we directly use the official implementation to evaluate the candidates. One notice is that L1 smoothing is very time consuming and we use the evaluation method mentioned in Bi et al. [2015] to speed up the experiments.

7.2 Shadow Annotation in the Wild

As to SAW Kovacs et al. [2017] test, instead of the original metric, we use the challenged metric in Li and Snavely [2018] as our measurements. The enhanced metric is considered to be able to highlight performance differences. Similarly, for L1 smoothing we use method in Bi et al. [2015] to obtain the test data.

8 Deep Learning Method Implementation Details

8.1 CGIntrinsics

Li and Snavely [2018]'s method has official open sourced implementations. We directly use their codes and their default configurations in our experiments.

8.2 GloSH

Li and Snavely [2018] reported their method as the current quantitative state of the art in IIW/SAW benchmarks. This method does not have open sourced implementations. The official paper Li and Snavely [2018] includes all necessary details and we reproduce their results using their recommended configurations. Our presented results on their method should only be considered as third-party reproductions. Nevertheless, all quantitative statistics of this method are directly transferred from their official paper.

9 User Study Implementation Details

Given the reflectance map $\boldsymbol{Y}$, the illumination map $\boldsymbol{S}$, and the source image $\boldsymbol{X}$, we formulate several simple and fundamental illumination manipulation use cases for the user study.

First of all, because some intrinsic decomposition methods do not require the perfect reconstruction of the original images, which is not acceptable in image editing, instead of directly using their illumination maps $\boldsymbol{S}$, we compute another reconstruction-guaranteed illumination map $\boldsymbol{S}^*$ as

$$\boldsymbol{S}^*_p = \frac{\boldsymbol{X}_p}{\boldsymbol{Y}_p + \epsilon} \tag{28}$$

where $\epsilon = 1e - 10$. In this way, the decomposition is enforced to perfectly reconstruct the source image.

9.1 Shadow enhancement

Given a simple fixed shadow threshold t_s, we view under-threshold pixels in illumination map $\boldsymbol{S}$ as shadow, and the shadow enhancement can be formulated as

$$\boldsymbol{X}^*_p = \begin{cases} \boldsymbol{Y}_p(\boldsymbol{S}^*_p)^2 & \boldsymbol{S}^*_p < t_s \\ \boldsymbol{Y}_p\boldsymbol{S}^*_p & \text{Others} \end{cases} \tag{29}$$

where $\boldsymbol{X}^*_p$ is the manipulated image. This transform only includes a very simple gamma correction, and the results can faithfully reflect the visual quality of the decomposed layers.

9.2 Specular reflection removal

Given a simple fixed specular reflection threshold t_r, we view over-threshold pixels in illumination map $\boldsymbol{S}$ as specular reflections, and the specular reflection removal can be formulated as

$$\boldsymbol{X}^*_p = \begin{cases} \boldsymbol{Y}_p & \boldsymbol{S}^*_p > t_s \\ \boldsymbol{Y}_p\boldsymbol{S}^*_p & \text{Others} \end{cases} \tag{30}$$

This transform is achieved via removing the specular reflection parts in the illumination map. It is very simple, and thus, faithful to the quality of the original decomposed layers.

9.3 Experiment setup and results

As mentioned in the main paper, we apply learning-based methods Zhou et al. [2019]; Li and Snavely [2018], optimization methods Bell et al. [2014]; Bi et al. [2015], and their EAP versions to decompose 100 scenes into layers. For fairness, we do not provide independent thresholds t_s and t_r for different methods or images. We use consistent parameter $t_s = 0.50$ and $t_r = 0.75$ for all involved methods and images for the sake of fairness. We obtain 600 results with enhanced shadows and 600 results with specular reflections eliminated. We employ Amazon Mechanical Turk (AMT) to rank the visual quality of these results and report the obtained ranking in Table. 1 and Table. 2.

9.4 Raw user study data

The 1200 raw user study results are listed in the *code-and-data* files. We also present some high-resolution ones from Fig. 42 to Fig. 55.

10 Additional Results

10.1 Image smoothing results

We provide additional results on image smoothing from Fig. 5 to Fig. 29.

10.2 Image decomposition results

We provide additional results on image decomposition from Fig. 32 to Fig. 36.

10.3 Intrinsic decomposition results

We provide additional results on intrinsic decomposition from Fig. 38 to Fig. 41.

10.4 Image manipulation results

We provide additional results on image manipulation from Fig. 42 to Fig. 55.

10.5 Supporting position visualizations

We extend the supporting position visualization in the main article from Fig. 57 to Fig. 62.

10.6 Limitation

We present limitation in Fig. 63.

References

Linchao Bao, Yibing Song, Qingxiong Yang, Hao Yuan, and Gang Wang. 2014. Tree filtering Efficient structurepreserving smoothing with a minimum spanning tree. *IEEE Transactions on Image Processing* (2014). 5, 8

Sean Bell, Kavita Bala, and Noah Snavely. 2014. Intrinsic Images in the Wild. *ACM Transactions on Graphics* 33, 4 (2014). 10, 13, 14

Sai Bi, Xiaoguang Han, and Yizhou Yu. 2015. An L1 image transform for edgepreserving smoothing and scenelevel intrinsic decomposition. *ACM Transactions on Graphics* (2015). 5, 8, 9, 10, 13, 14

Robert Carroll, Ravi Ramamoorthi, and Maneesh Agrawala. 2011. Illumination decomposition for material recoloring with consistent interreflections. *ACM Transactions on Graphics* (2011). 11, 13

Balazs Kovacs, Sean Bell, Noah Snavely, and Kavita Bala. 2017. Shading Annotations in the Wild. *CVPR* (2017). 13

Zhengqi Li and Noah Snavely. 2018. CGIntrinsics: Better Intrinsic Image Decomposition through Physically-Based Rendering. *ECCV* (2018). 13, 14

Patrick Perez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. *ACM Transactions on Graphics* (2003). 10

RUDIN, L, OSHER, S, , FATEMI, and E. 1992. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena* (1992). 5, 7

Stanford. 2001. A Greedy Knapsack Heuristic. *Shortest Paths Revisited, NP-Complete Problems and What To Do About Them* (2001). 6

Jianchao Tan, Jose Echevarria, and Yotam Ginglod. 2018. Efficient palette-based decomposition and recoloring of images via RGBXY-space geometry. *ACM Transactions on Graphics* (2018). 13

Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. 2016. Decomposing Images into Layers via RGB-space Geometry. *ACM Transactions on Graphics* (2016). 13

Paul W Holland Roy E Welsch. 1977. Robust regression using iteratively reweighted leastsquares. *Communications in Statisticstheory and Methods* (1977). 5, 7

Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. 2011. Image smoothing via L0 gradient minimization. *ACM Transactions on Graphics* (2011). 5, 7, 8

Li Xu, Qiong Yan, Yang Xia, and Jiaya Jia. 2012. Structure extraction from texture via relative total variation. *ACM Transactions on Graphics* (2012). 4, 5, 8

Hao Zhou, Xiang Yu, and David W Jacobs. 2019. GLoSH: Global-Local Spherical Harmonics for Intrinsic Image Decomposition. *ICCV* (2019). 14

Table 1: Amazon Mechanical Turk (AMT) human ranking on Shadow Enhancement (SE).

Instance	CGIntrinsics	GLoSH	Bell	EAP + Bell	DL1	EAP + DL1
./sample/0.webp	5	6	3	2	4	1
./sample/1.webp	6	5	3	2	4	1
./sample/2.webp	5	6	4	2	3	1
./sample/3.webp	5	6	4	1	3	2
./sample/4.webp	6	5	4	2	3	1
./sample/5.webp	5	6	3	1	4	2
./sample/6.webp	6	5	3	2	4	1
./sample/7.webp	6	5	4	2	3	1
./sample/8.webp	5	6	4	2	3	1
./sample/9.webp	6	5	4	2	3	1
./sample/10.webp	5	6	4	1	3	2
./sample/11.webp	6	5	4	2	3	1
./sample/12.webp	6	5	4	2	3	1
./sample/13.webp	5	6	3	2	4	1
./sample/14.webp	6	5	3	2	4	1
./sample/15.webp	6	5	3	2	4	1
./sample/16.webp	6	5	4	2	3	1
./sample/17.webp	6	5	4	2	3	1
./sample/18.webp	6	5	4	2	3	1
./sample/19.webp	6	5	3	2	4	1
./sample/20.webp	6	5	3	1	4	2
./sample/21.webp	3	5	6	2	4	1
./sample/22.webp	6	5	4	1	3	2
./sample/23.webp	6	5	4	2	3	1
./sample/24.webp	5	6	3	2	4	1
./sample/25.webp	6	5	4	2	3	1
./sample/26.webp	5	6	4	2	3	1
./sample/27.webp	6	5	3	2	4	1
./sample/28.webp	5	6	4	1	3	2
./sample/29.webp	6	5	3	2	4	1
./sample/30.webp	6	5	4	2	3	1
./sample/31.webp	3	6	5	4	1	2
./sample/32.webp	6	5	4	1	3	2
./sample/33.webp	6	5	4	1	3	2
./sample/34.webp	6	5	4	1	3	2
./sample/35.webp	5	6	3	1	4	2
./sample/36.webp	6	5	4	2	3	1
./sample/37.webp	3	5	2	6	4	1
./sample/38.webp	6	5	4	2	3	1
./sample/39.webp	3	6	1	5	4	2
./sample/40.webp	6	5	3	2	4	1
./sample/41.webp	6	5	3	2	4	1
./sample/42.webp	6	5	3	1	4	2
./sample/43.webp	5	6	3	2	4	1
./sample/44.webp	5	6	4	2	3	1
./sample/45.webp	6	5	3	2	4	1
./sample/46.webp	5	6	4	2	3	1
./sample/47.webp	5	6	4	2	3	1
./sample/48.webp	6	5	4	2	3	1
./sample/49.webp	6	5	3	2	4	1
./sample/50.webp	6	5	3	1	4	2
./sample/51.webp	6	5	3	1	4	2
./sample/52.webp	6	5	4	1	3	2
./sample/53.webp	6	5	3	1	4	2
./sample/54.webp	5	6	4	2	3	1
./sample/55.webp	5	6	4	2	3	1
./sample/56.webp	6	5	3	2	4	1
./sample/57.webp	6	5	3	1	4	2
./sample/58.webp	6	5	3	2	4	1
./sample/59.webp	6	5	3	2	4	1
./sample/60.webp	6	5	4	2	3	1
./sample/61.webp	6	5	4	2	3	1
./sample/62.webp	6	5	3	2	4	1
./sample/63.webp	5	6	4	2	3	1
./sample/64.webp	6	5	4	2	3	1
./sample/65.webp	5	6	4	2	3	1
./sample/66.webp	6	2	5	3	4	1
./sample/67.webp	6	5	3	1	4	2
./sample/68.webp	5	6	3	2	4	1
./sample/69.webp	5	6	4	2	3	1
./sample/70.webp	6	5	3	1	4	2
./sample/71.webp	6	5	3	2	4	1
./sample/72.webp	6	2	5	3	4	1
./sample/73.webp	5	6	3	2	4	1
./sample/74.webp	6	1	3	5	4	2
./sample/75.webp	6	5	3	2	4	1
./sample/76.webp	5	6	4	2	3	1
./sample/77.webp	6	5	3	1	4	2
./sample/78.webp	5	6	4	2	3	1
./sample/79.webp	6	5	3	2	4	1
./sample/80.webp	6	5	3	1	4	2
./sample/81.webp	6	5	4	1	3	2
./sample/82.webp	5	6	4	2	3	1
./sample/83.webp	5	6	4	2	3	1
./sample/84.webp	6	5	4	2	3	1
./sample/85.webp	6	5	4	2	3	1
./sample/86.webp	6	5	4	2	3	1
./sample/87.webp	6	5	4	1	3	2
./sample/88.webp	6	5	4	2	3	1
./sample/89.webp	6	5	4	2	3	1
./sample/90.webp	6	5	4	2	3	1
./sample/91.webp	3	6	2	5	4	1
./sample/92.webp	6	5	4	2	3	1
./sample/93.webp	6	5	3	1	4	2
./sample/94.webp	6	5	4	2	3	1
./sample/95.webp	5	6	4	1	3	2
./sample/96.webp	6	5	4	2	3	1
./sample/97.webp	6	5	4	1	3	2
./sample/98.webp	6	5	3	1	4	2
./sample/99.webp	6	2	5	3	4	1
Mean	5.70	5.30	3.55	1.71	3.45	1.29
Std	0.46	0.46	0.50	0.45	0.50	0.45

Table 2: Amazon Mechanical Turk (AMT) human ranking on Specular Reflection Removal (SRR).

Instance	CGIntrinsics	GLoSH	Bell	EAP + Bell	DL1	EAP + DL1
./sample/0.webp	5	6	3	2	4	1
./sample/1.webp	5	3	2	6	4	1
./sample/2.webp	5	6	4	2	3	1
./sample/3.webp	6	5	3	2	4	1
./sample/4.webp	6	5	4	2	3	1
./sample/5.webp	5	6	3	2	4	1
./sample/6.webp	6	5	4	3	2	1
./sample/7.webp	6	5	4	1	3	2
./sample/8.webp	5	6	4	2	3	1
./sample/9.webp	5	6	3	2	4	1
./sample/10.webp	5	6	4	1	3	2
./sample/11.webp	5	6	4	2	3	1
./sample/12.webp	6	5	3	2	4	1
./sample/13.webp	6	5	4	1	3	2
./sample/14.webp	5	6	3	2	4	1
./sample/15.webp	5	6	4	2	3	1
./sample/16.webp	6	5	3	1	4	2
./sample/17.webp	5	6	4	2	3	1
./sample/18.webp	5	6	4	2	3	1
./sample/19.webp	6	2	4	5	3	1
./sample/20.webp	1	6	5	4	3	2
./sample/21.webp	6	5	4	1	3	2
./sample/22.webp	6	5	3	2	4	1
./sample/23.webp	5	6	4	1	3	2
./sample/24.webp	2	4	6	5	3	1
./sample/25.webp	5	6	3	2	4	1
./sample/26.webp	2	5	4	3	6	1
./sample/27.webp	5	6	3	1	4	2
./sample/28.webp	6	4	1	5	3	2
./sample/29.webp	5	6	4	2	3	1
./sample/30.webp	6	5	3	1	4	2
./sample/31.webp	6	5	3	2	4	1
./sample/32.webp	5	6	3	2	4	1
./sample/33.webp	5	6	4	2	3	1
./sample/34.webp	5	6	3	2	4	1
./sample/35.webp	5	6	3	1	4	2
./sample/36.webp	5	6	3	2	4	1
./sample/37.webp	5	6	3	2	4	1
./sample/38.webp	1	4	6	5	3	2
./sample/39.webp	5	6	4	2	3	1
./sample/40.webp	6	5	4	2	3	1
./sample/41.webp	5	6	3	2	4	1
./sample/42.webp	5	6	4	1	3	2
./sample/43.webp	5	6	3	2	4	1
./sample/44.webp	5	6	4	2	3	1
./sample/45.webp	5	6	3	2	4	1
./sample/46.webp	6	4	5	2	3	1
./sample/47.webp	5	6	3	1	4	2
./sample/48.webp	5	6	4	2	3	1
./sample/49.webp	6	5	4	2	3	1
./sample/50.webp	5	6	3	1	4	2
./sample/51.webp	5	6	3	1	4	2
./sample/52.webp	5	6	4	2	3	1
./sample/53.webp	5	6	3	1	4	2
./sample/54.webp	5	6	4	2	3	1
./sample/55.webp	5	6	3	2	4	1
./sample/56.webp	5	6	3	1	4	2
./sample/57.webp	5	6	3	2	4	1
./sample/58.webp	5	3	6	2	4	1
./sample/59.webp	5	6	3	2	4	1
./sample/60.webp	3	6	2	5	4	1
./sample/61.webp	5	6	4	1	3	2
./sample/62.webp	5	6	4	2	3	1
./sample/63.webp	5	6	4	2	3	1
./sample/64.webp	5	6	3	1	4	2
./sample/65.webp	5	6	3	2	4	1
./sample/66.webp	5	6	4	2	3	1
./sample/67.webp	5	6	4	2	3	1
./sample/68.webp	5	6	3	2	4	1
./sample/69.webp	5	6	3	2	4	1
./sample/70.webp	5	6	3	1	4	2
./sample/71.webp	4	5	6	2	3	1
./sample/72.webp	5	6	4	2	3	1
./sample/73.webp	6	5	3	2	4	1
./sample/74.webp	6	5	4	1	3	2
./sample/75.webp	5	6	4	1	3	2
./sample/76.webp	5	6	3	2	4	1
./sample/77.webp	5	4	3	6	1	2
./sample/78.webp	2	6	5	4	3	1
./sample/79.webp	6	5	3	2	4	1
./sample/80.webp	6	5	3	2	4	1
./sample/81.webp	5	6	3	2	4	1
./sample/82.webp	5	6	4	2	3	1
./sample/83.webp	6	5	3	2	4	1
./sample/84.webp	5	6	4	1	3	2
./sample/85.webp	5	6	4	2	3	1
./sample/86.webp	5	6	4	1	3	2
./sample/87.webp	1	5	3	6	4	2
./sample/88.webp	5	6	4	1	3	2
./sample/89.webp	6	5	3	1	4	2
./sample/90.webp	5	6	3	1	4	2
./sample/91.webp	6	5	4	1	3	2
./sample/92.webp	5	6	4	2	3	1
./sample/93.webp	5	6	4	2	3	1
./sample/94.webp	5	6	3	2	4	1
./sample/95.webp	5	6	4	1	3	2
./sample/96.webp	5	6	4	1	3	2
./sample/97.webp	5	6	4	1	3	2
./sample/98.webp	6	5	4	2	3	1
./sample/99.webp	5	6	4	2	3	1
Mean	5.22	5.77	3.54	1.66	3.47	1.34
Std	0.41	0.44	0.52	0.47	0.50	0.47

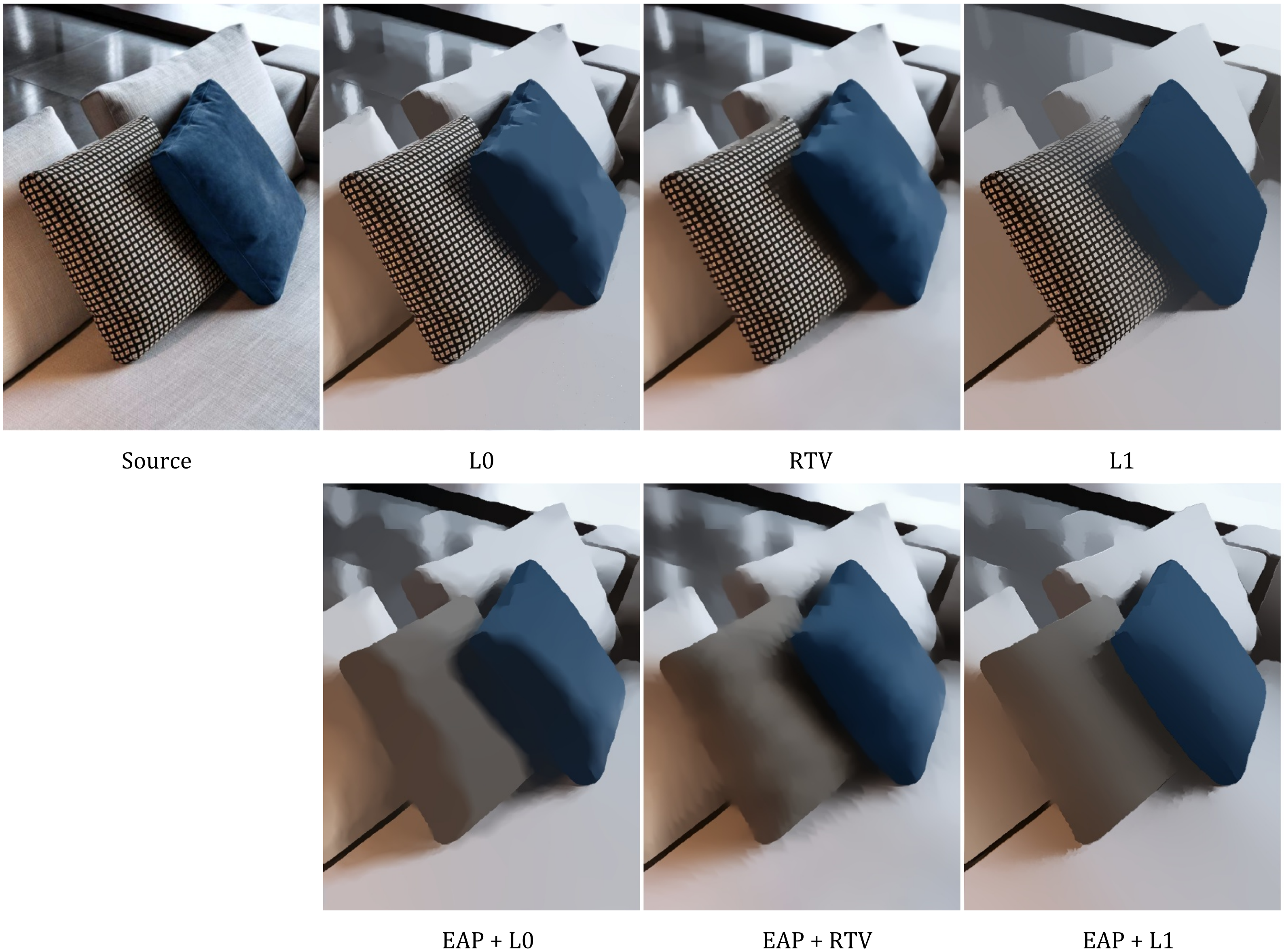

Figure 5: **Texture removal.** Visual comparison of texture removal with or without the EAP framework.

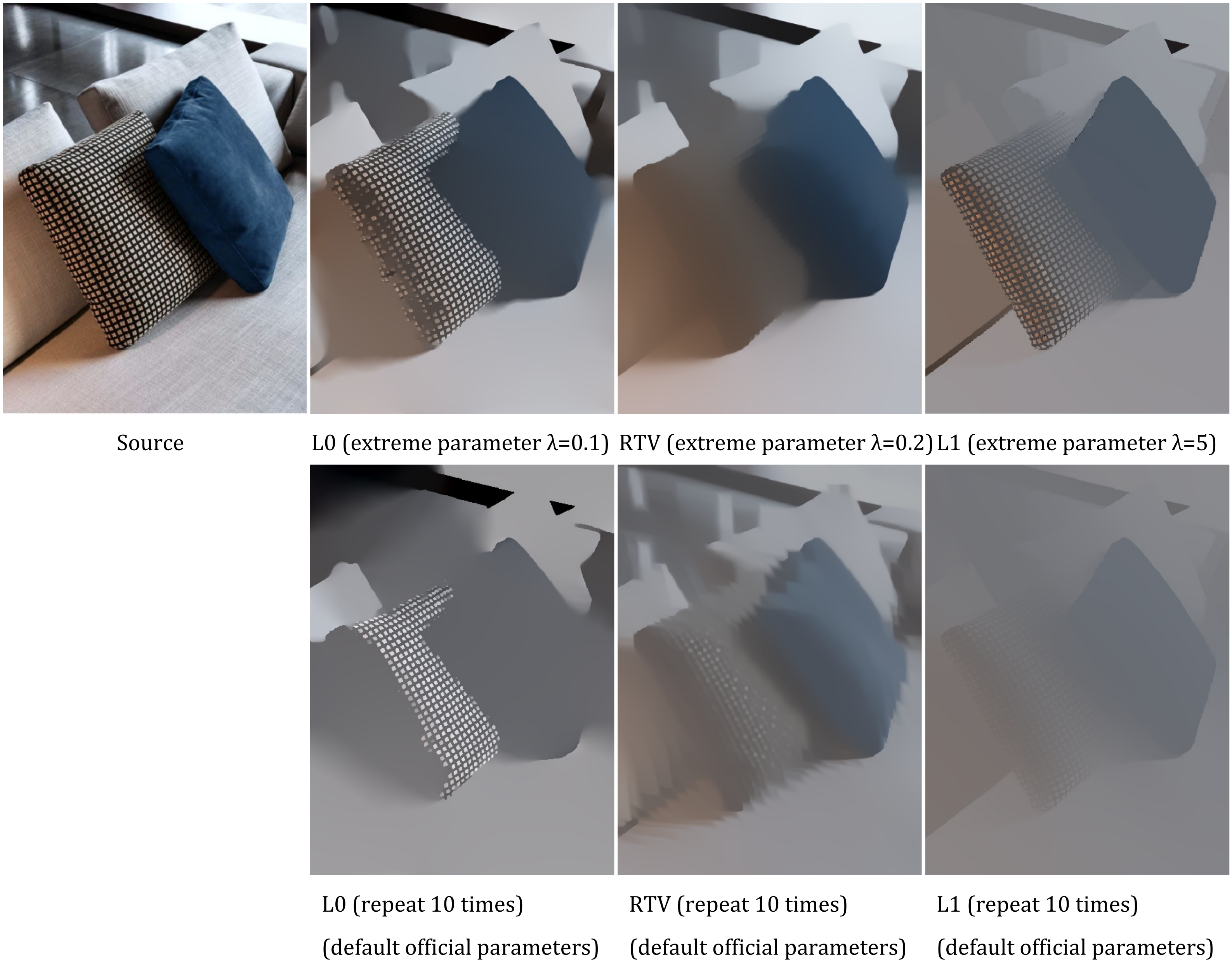

Figure 6: **Texture removal.** Trying to fine tune existing methods to facilitate adequate smoothing. The EAP results cannot be achieved by tuning parameters of existing methods or repeating existing methods for multiple times.

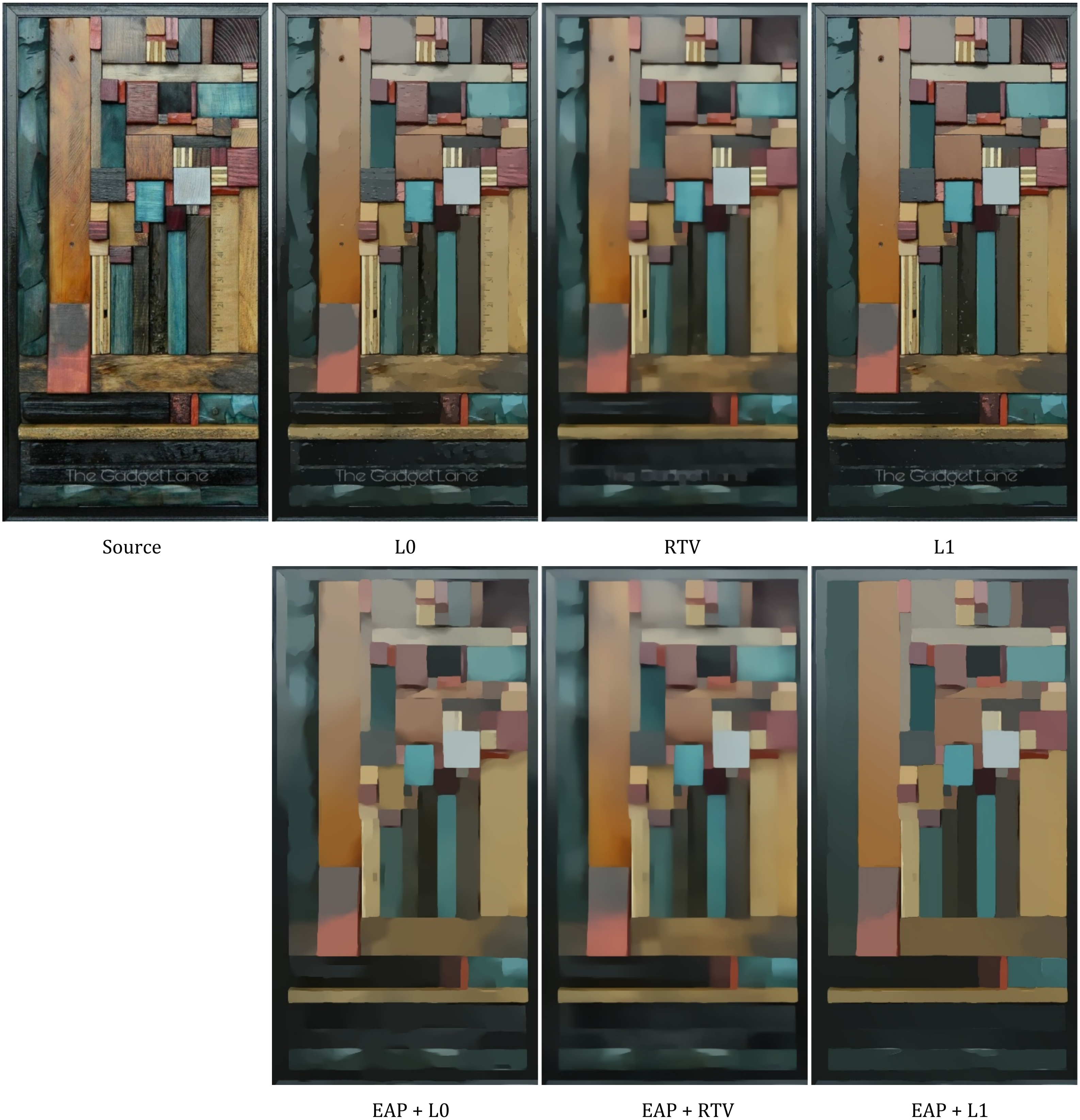

Figure 7: **Texture removal.** Visual comparison of texture removal with or without the EAP framework.

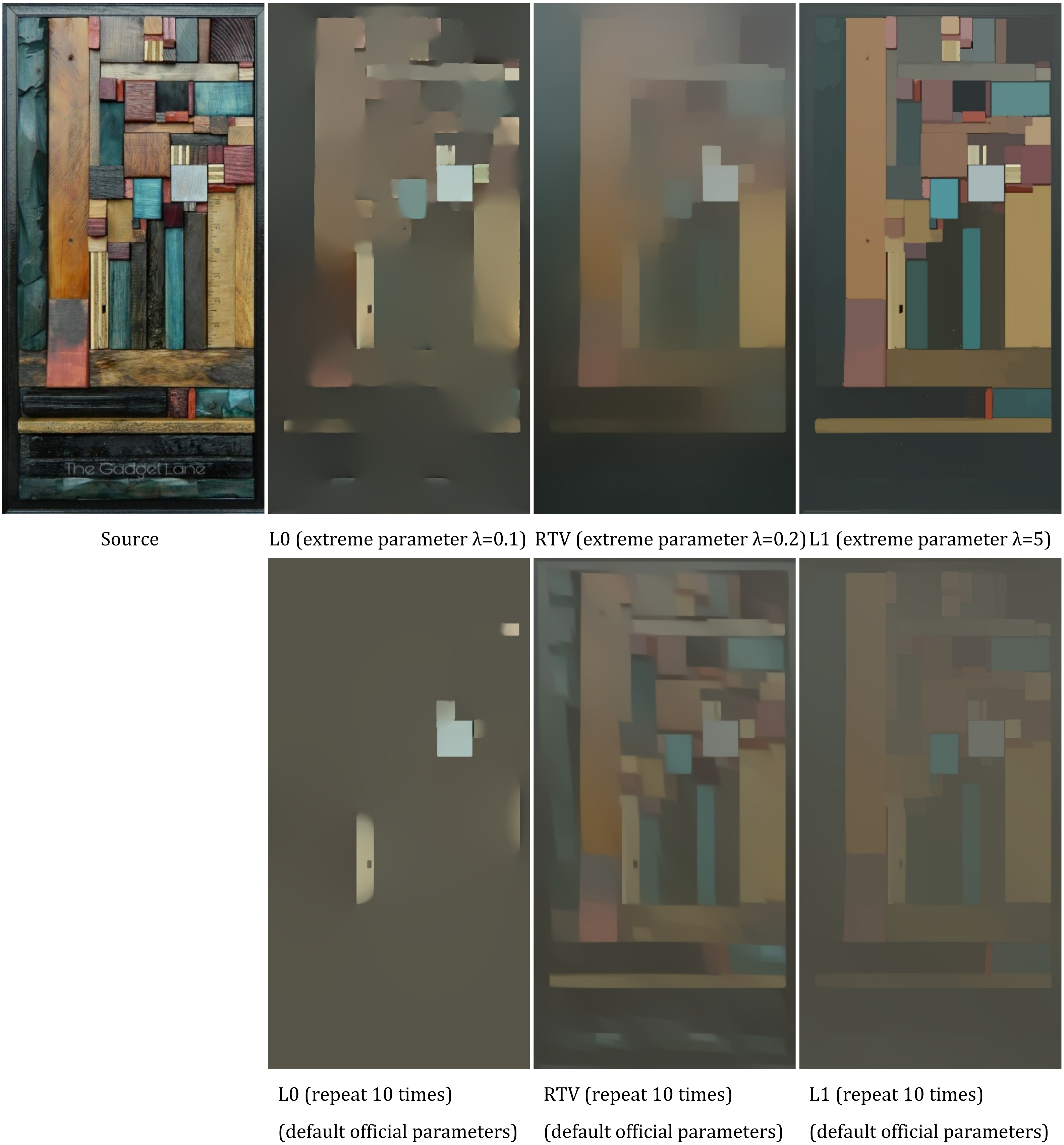

Figure 8: **Texture removal.** Trying to fine tune existing methods to facilitate adequate smoothing. The EAP results cannot be achieved by tuning parameters of existing methods or repeating existing methods for multiple times.

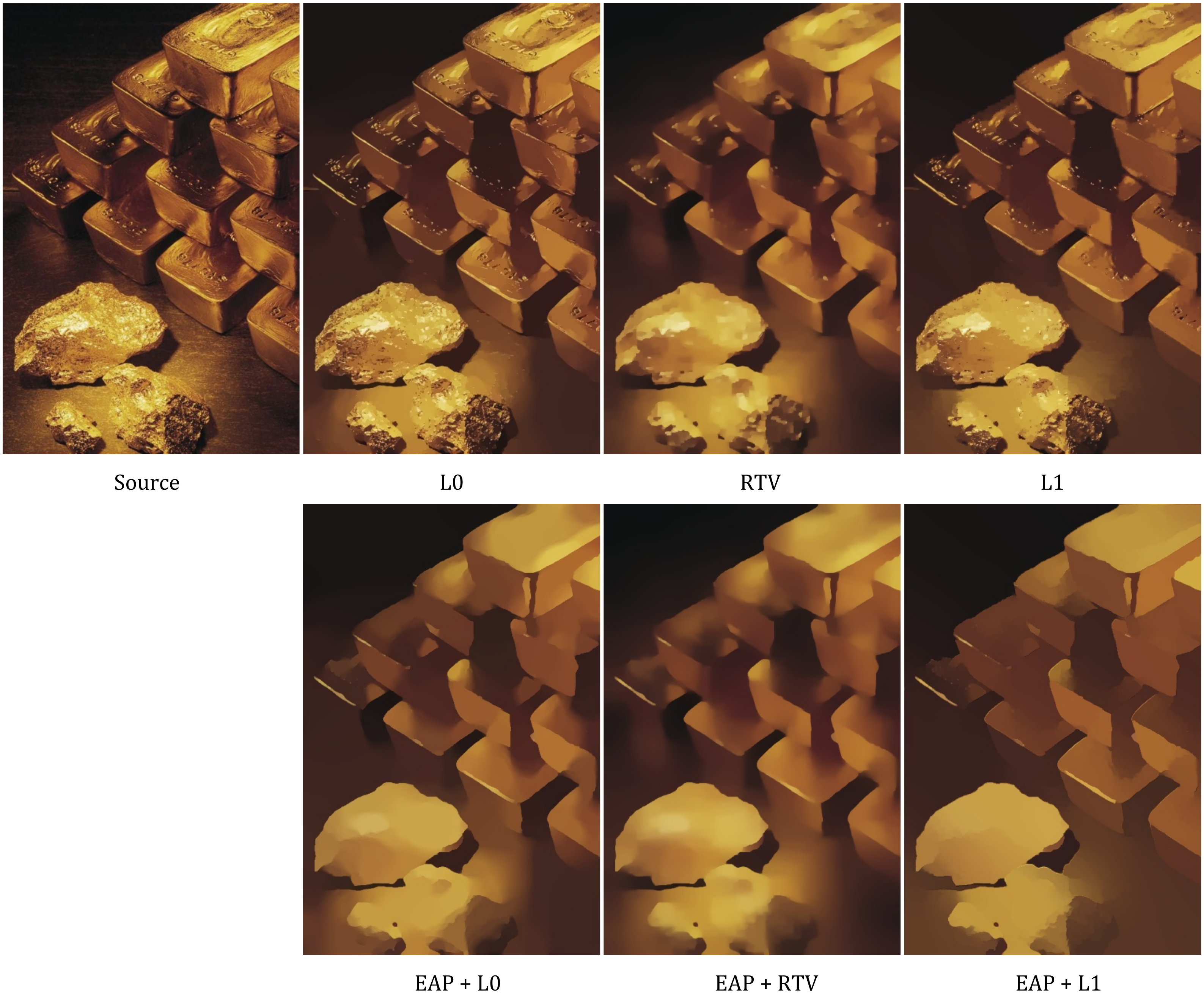

Figure 9: **Texture removal.** Visual comparison of texture removal with or without the EAP framework.

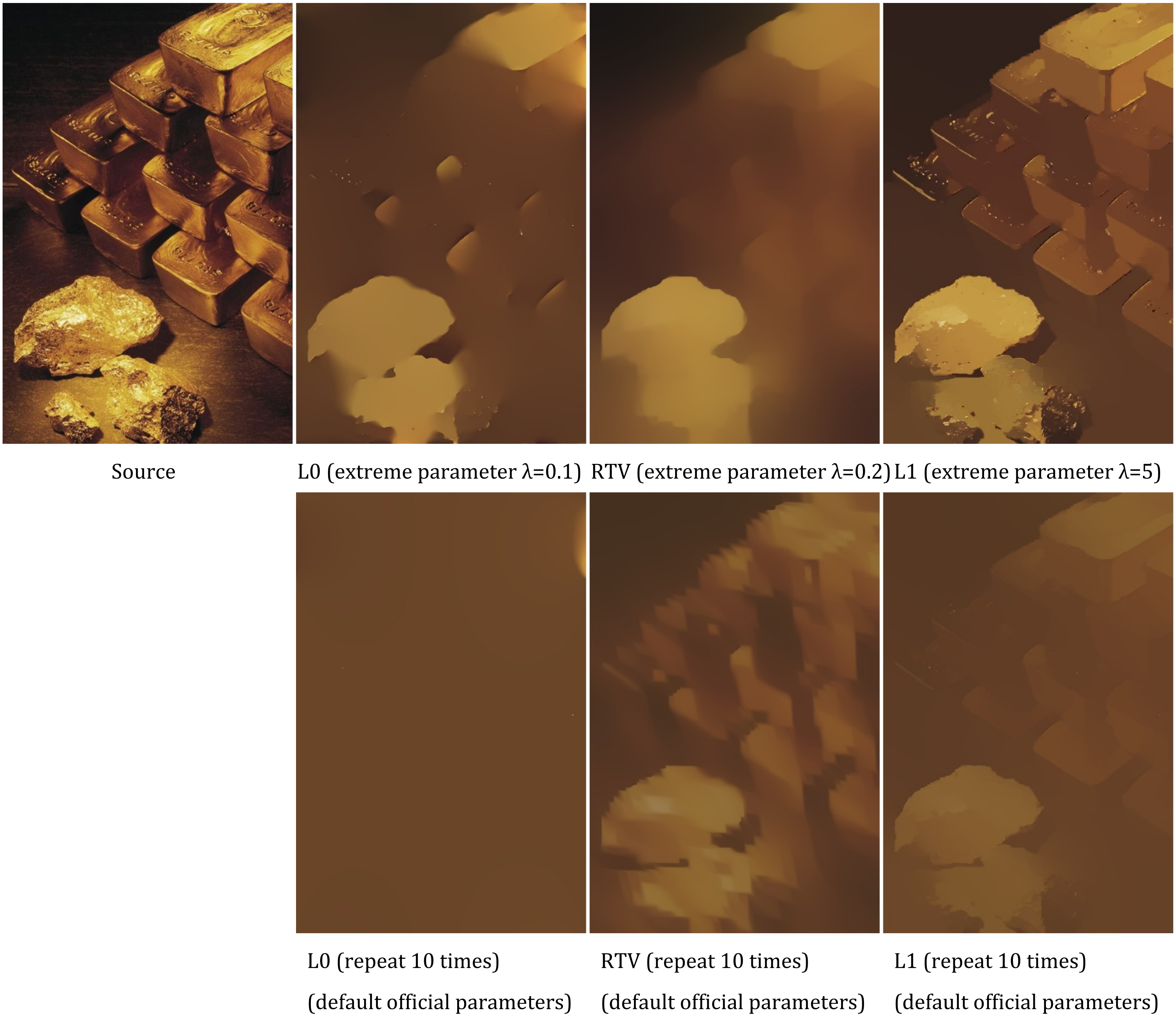

Figure 10: **Texture removal.** Trying to fine tune existing methods to facilitate adequate smoothing. The EAP results cannot be achieved by tuning parameters of existing methods or repeating existing methods for multiple times.

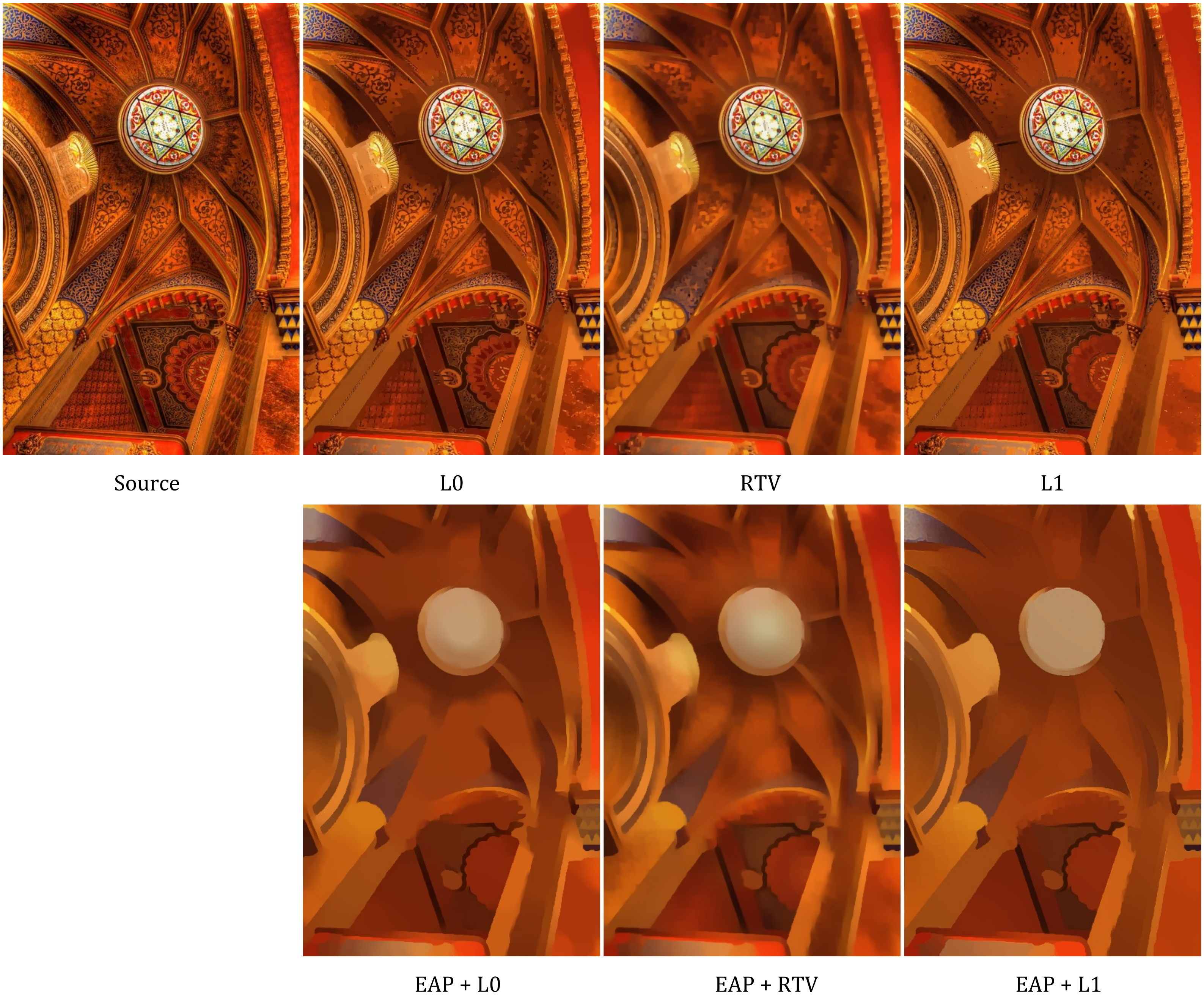

Figure 11: **Texture removal.** Visual comparison of texture removal with or without the EAP framework.

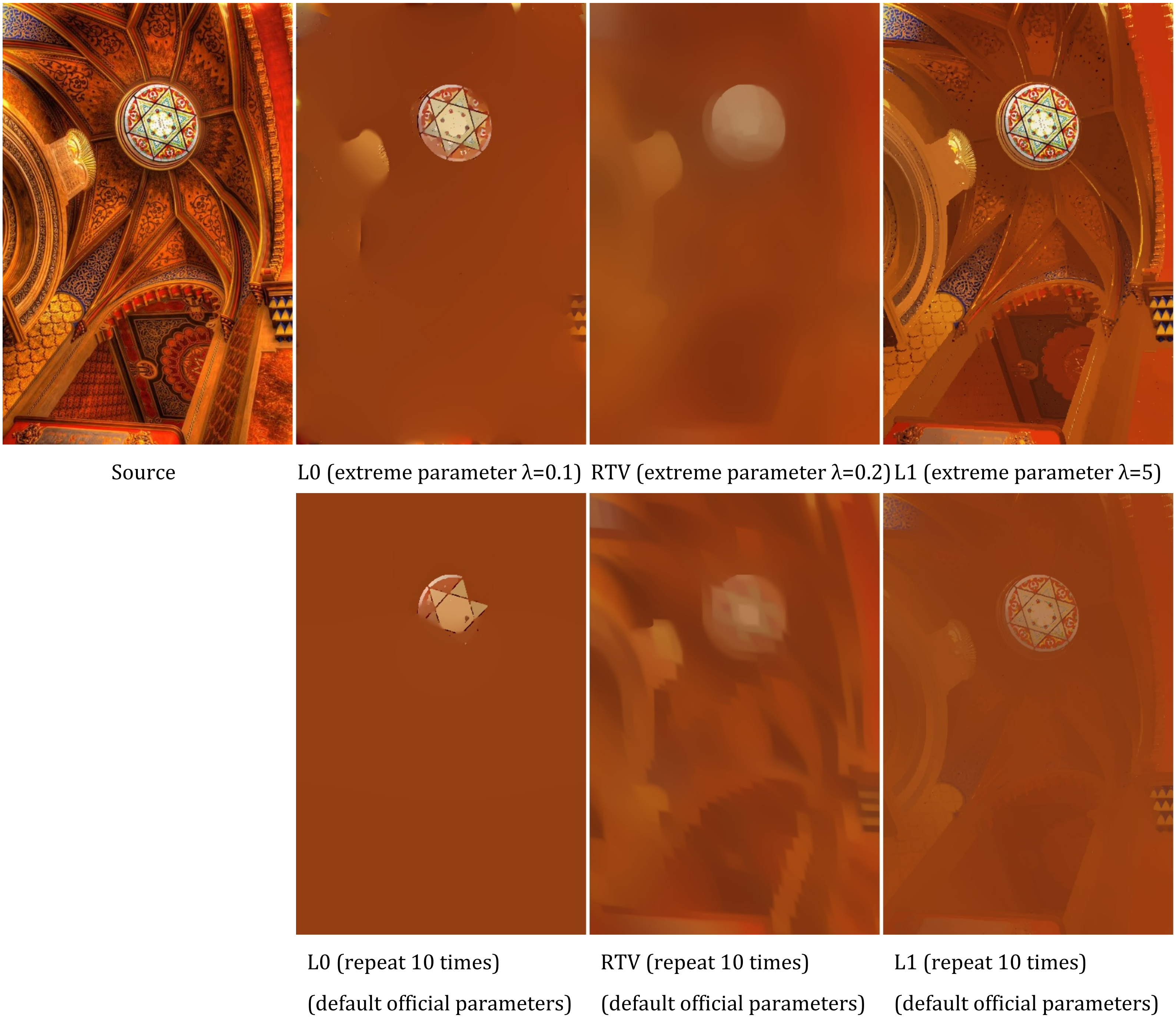

Figure 12: **Texture removal.** Trying to fine tune existing methods to facilitate adequate smoothing. The EAP results cannot be achieved by tuning parameters of existing methods or repeating existing methods for multiple times.

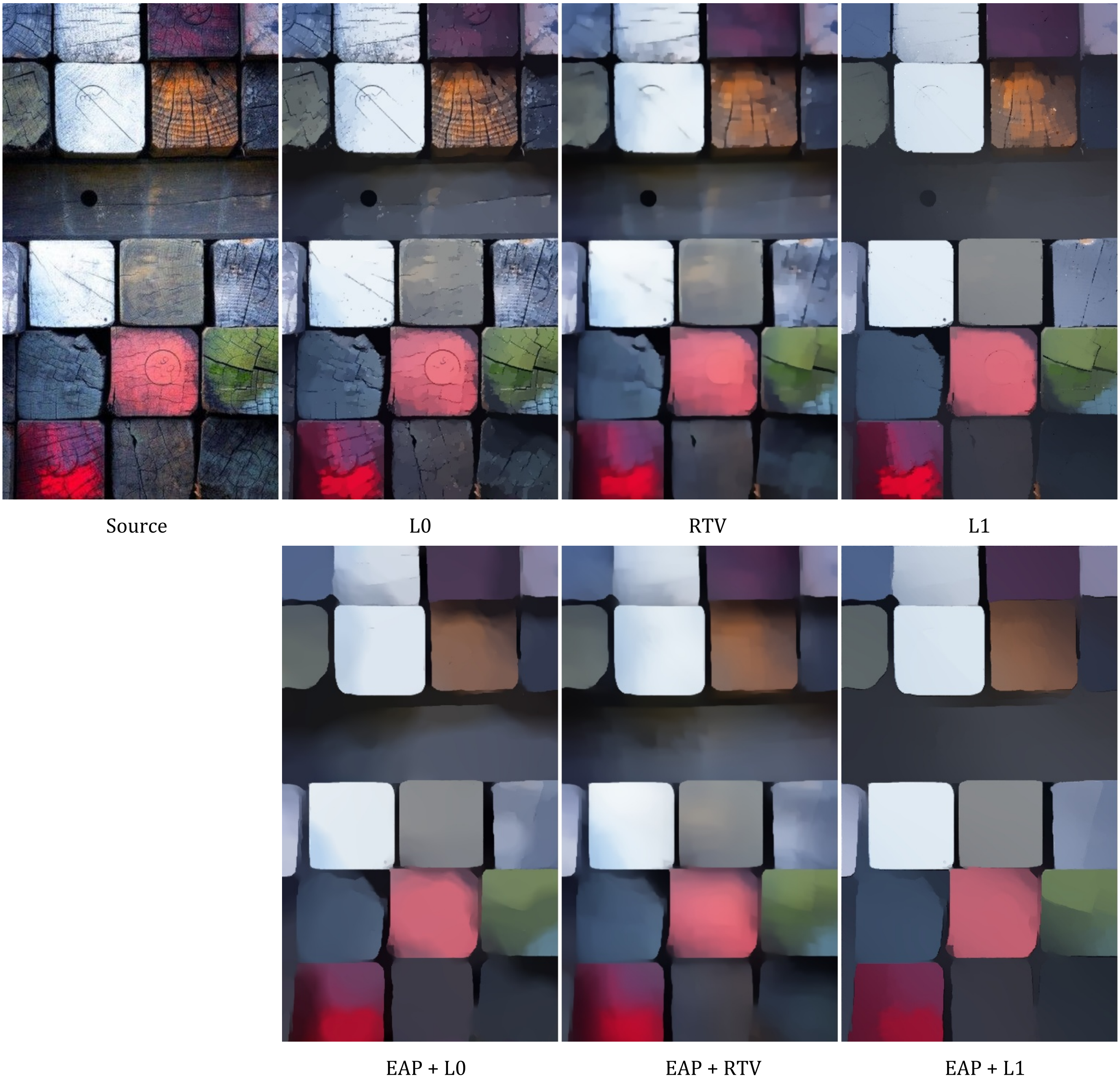

Figure 13: **Texture removal.** Visual comparison of texture removal with or without the EAP framework.

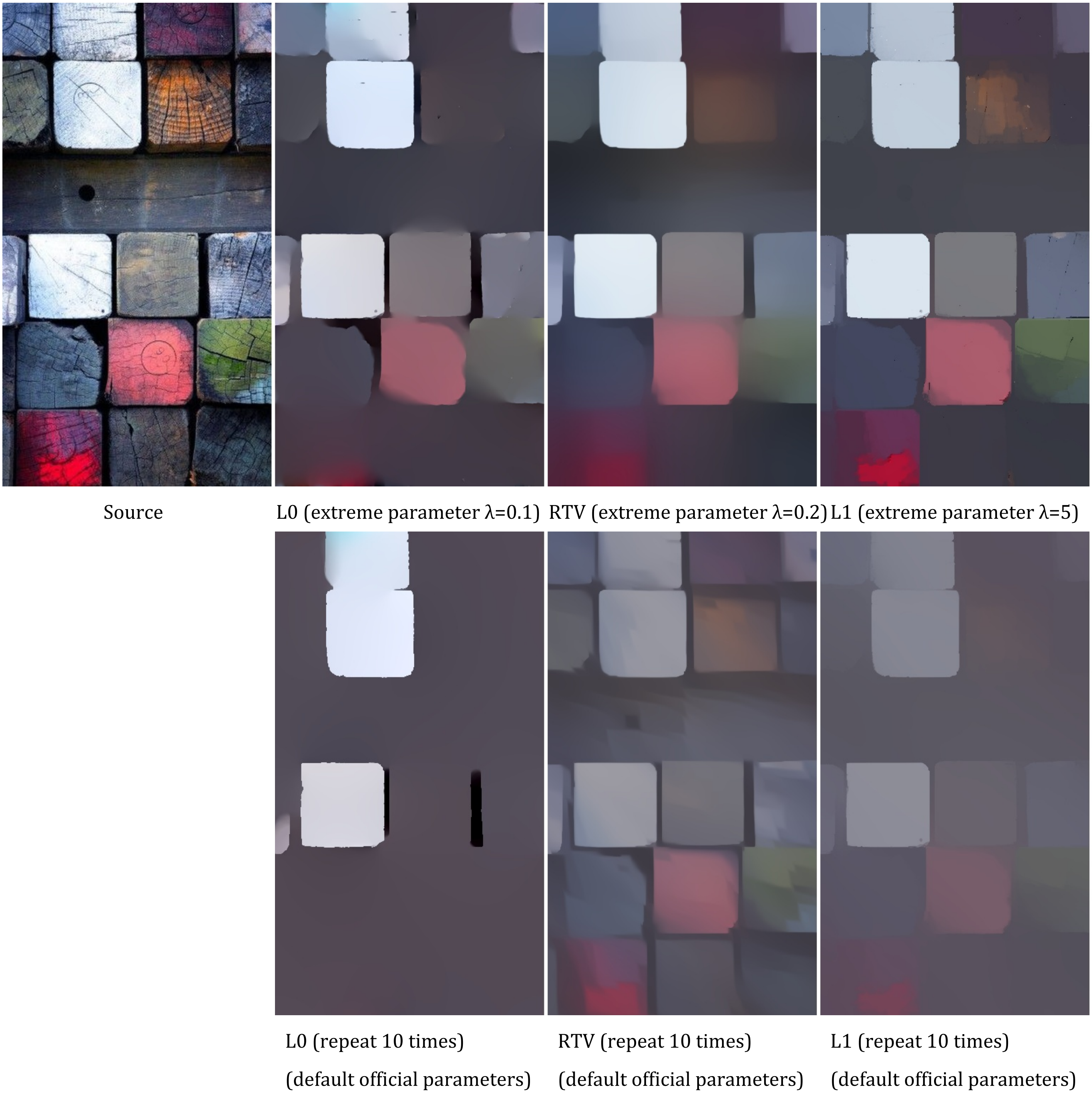

Figure 14: **Texture removal.** Trying to fine tune existing methods to facilitate adequate smoothing. The EAP results cannot be achieved by tuning parameters of existing methods or repeating existing methods for multiple times.

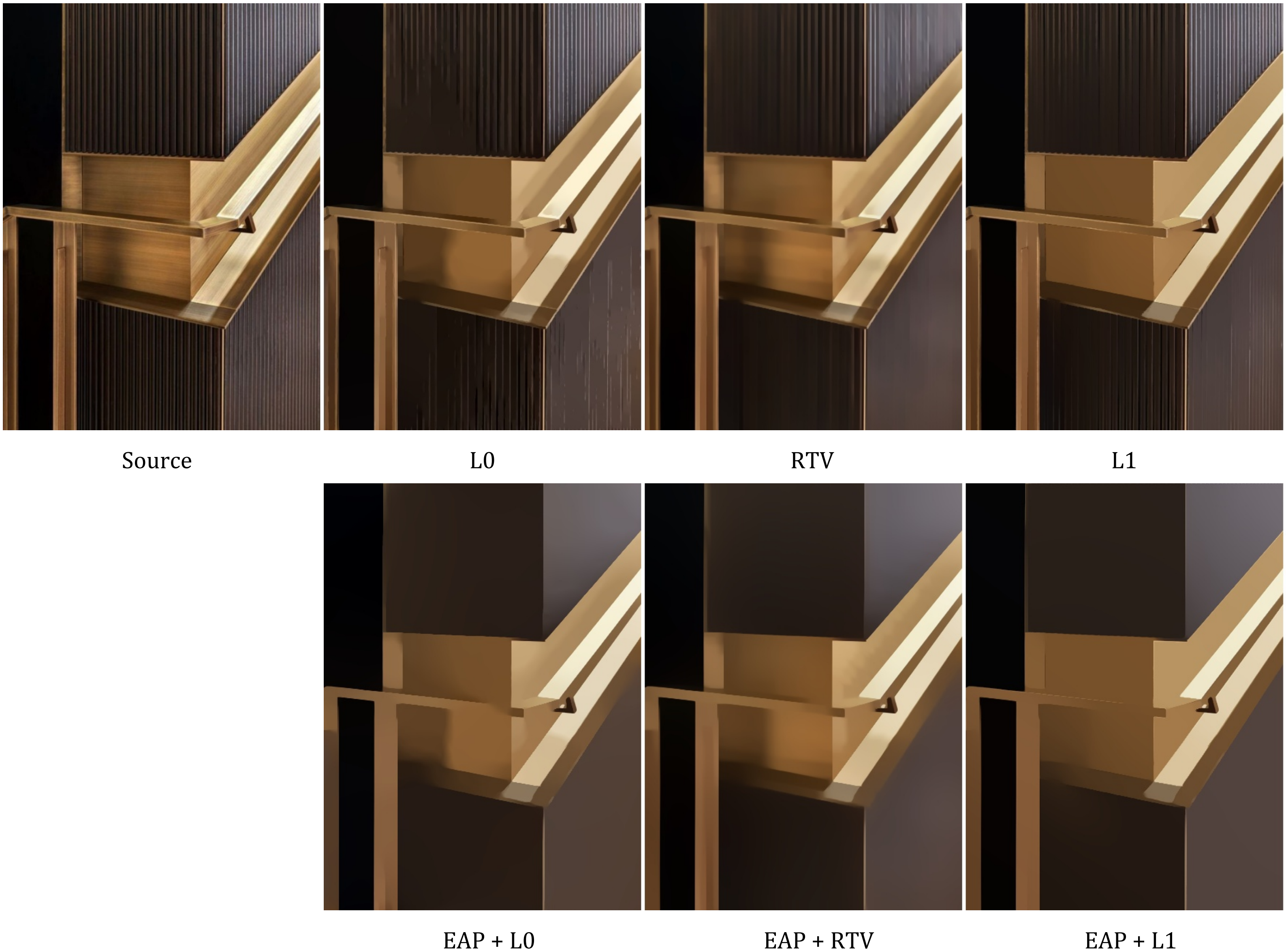

Figure 15: **Texture removal.** Visual comparison of texture removal with or without the EAP framework.

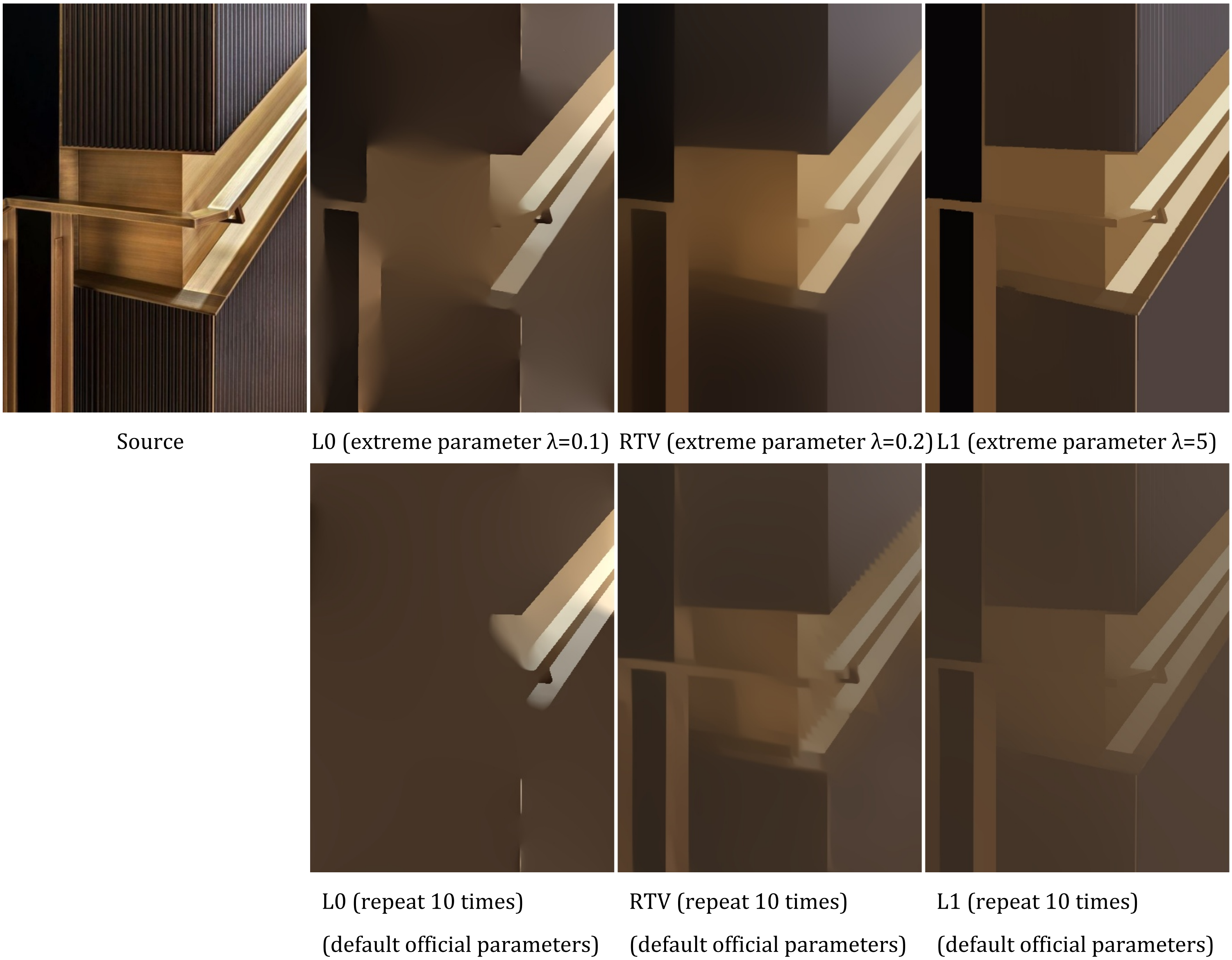

Figure 16: **Texture removal.** Trying to fine tune existing methods to facilitate adequate smoothing. The EAP results cannot be achieved by tuning parameters of existing methods or repeating existing methods for multiple times.

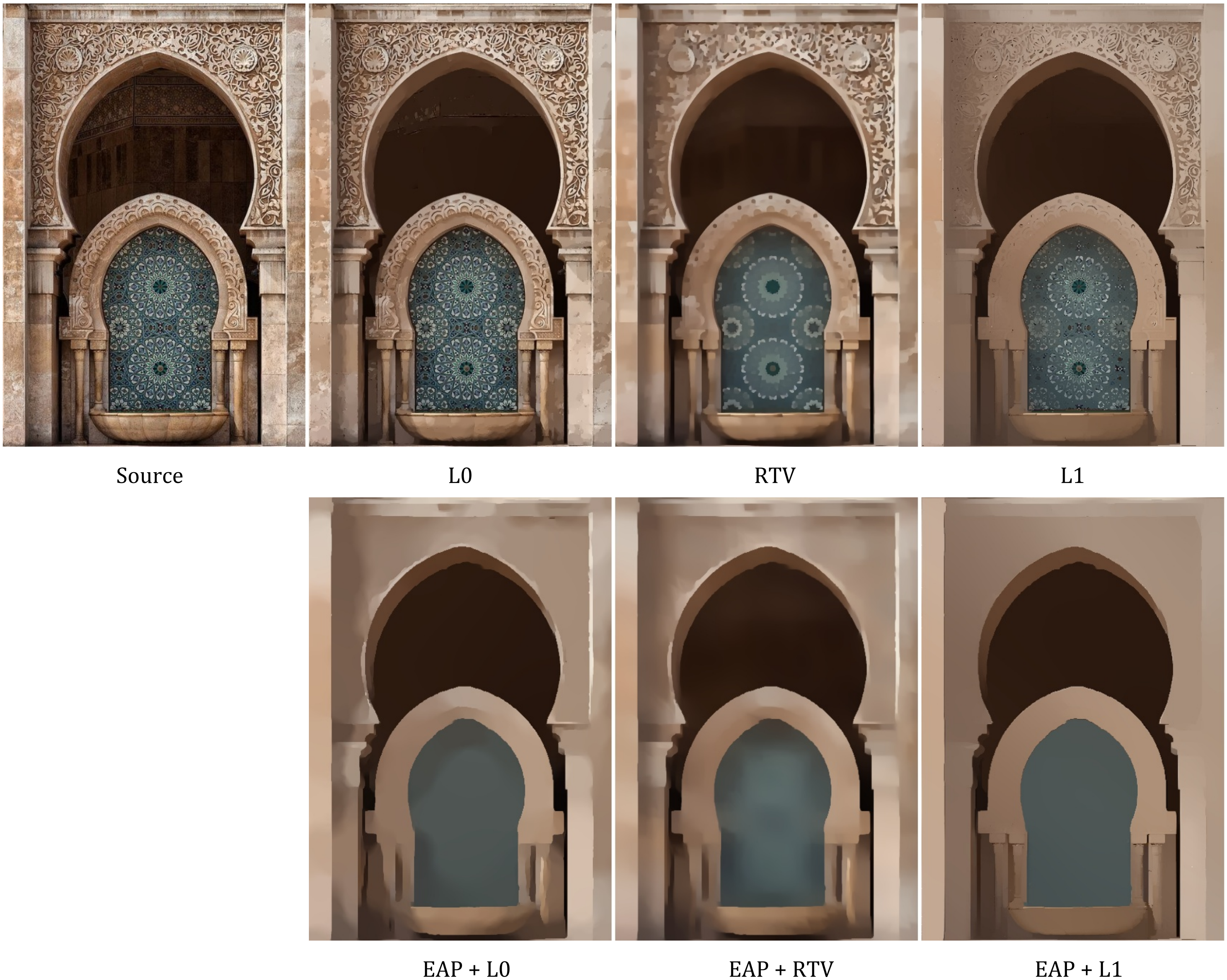

Figure 17: **Texture removal.** Visual comparison of texture removal with or without the EAP framework.

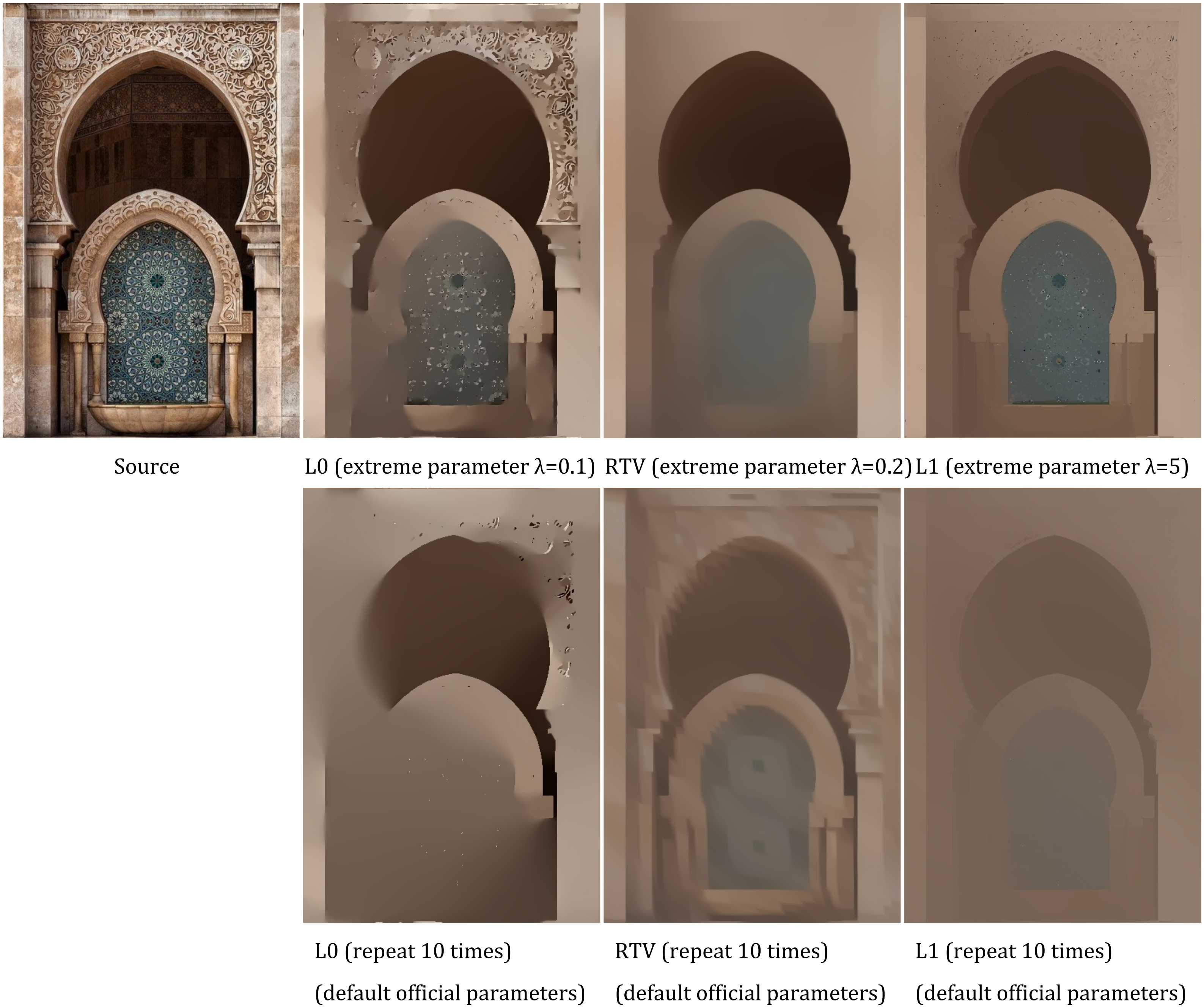

Figure 18: **Texture removal.** Trying to fine tune existing methods to facilitate adequate smoothing. The EAP results cannot be achieved by tuning parameters of existing methods or repeating existing methods for multiple times.

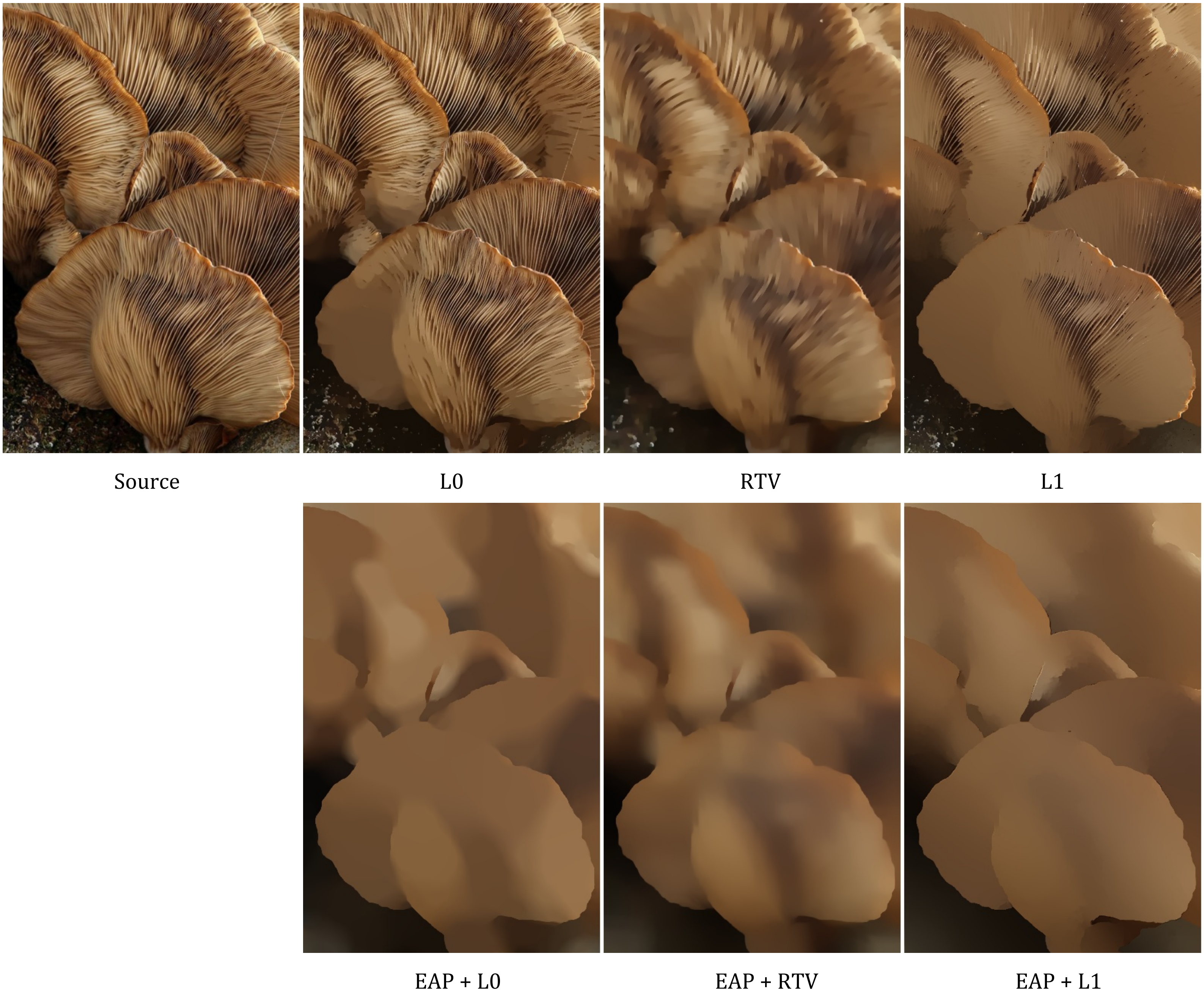

Figure 19: **Texture removal.** Visual comparison of texture removal with or without the EAP framework.

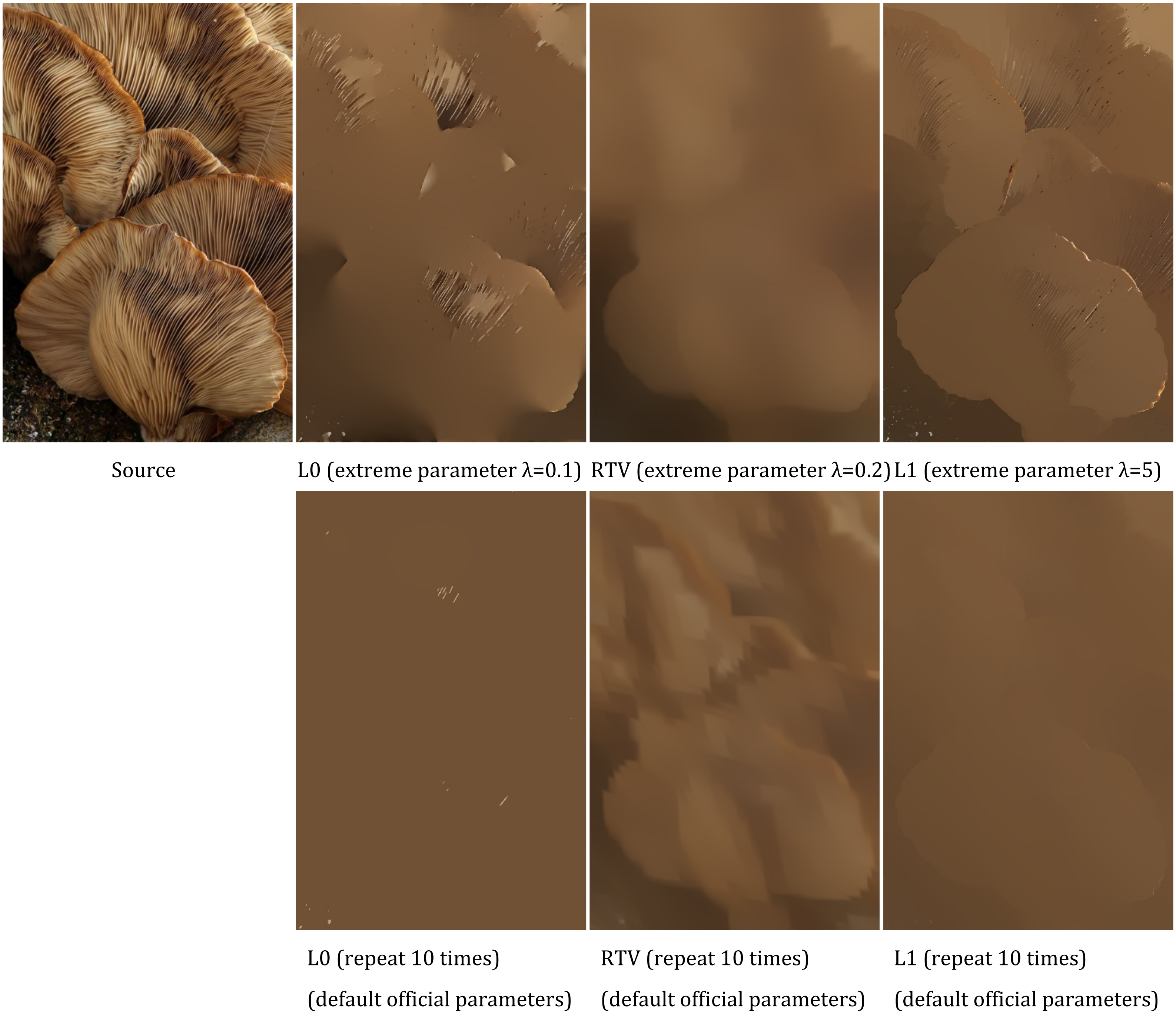

Figure 20: **Texture removal.** Trying to fine tune existing methods to facilitate adequate smoothing. The EAP results cannot be achieved by tuning parameters of existing methods or repeating existing methods for multiple times.

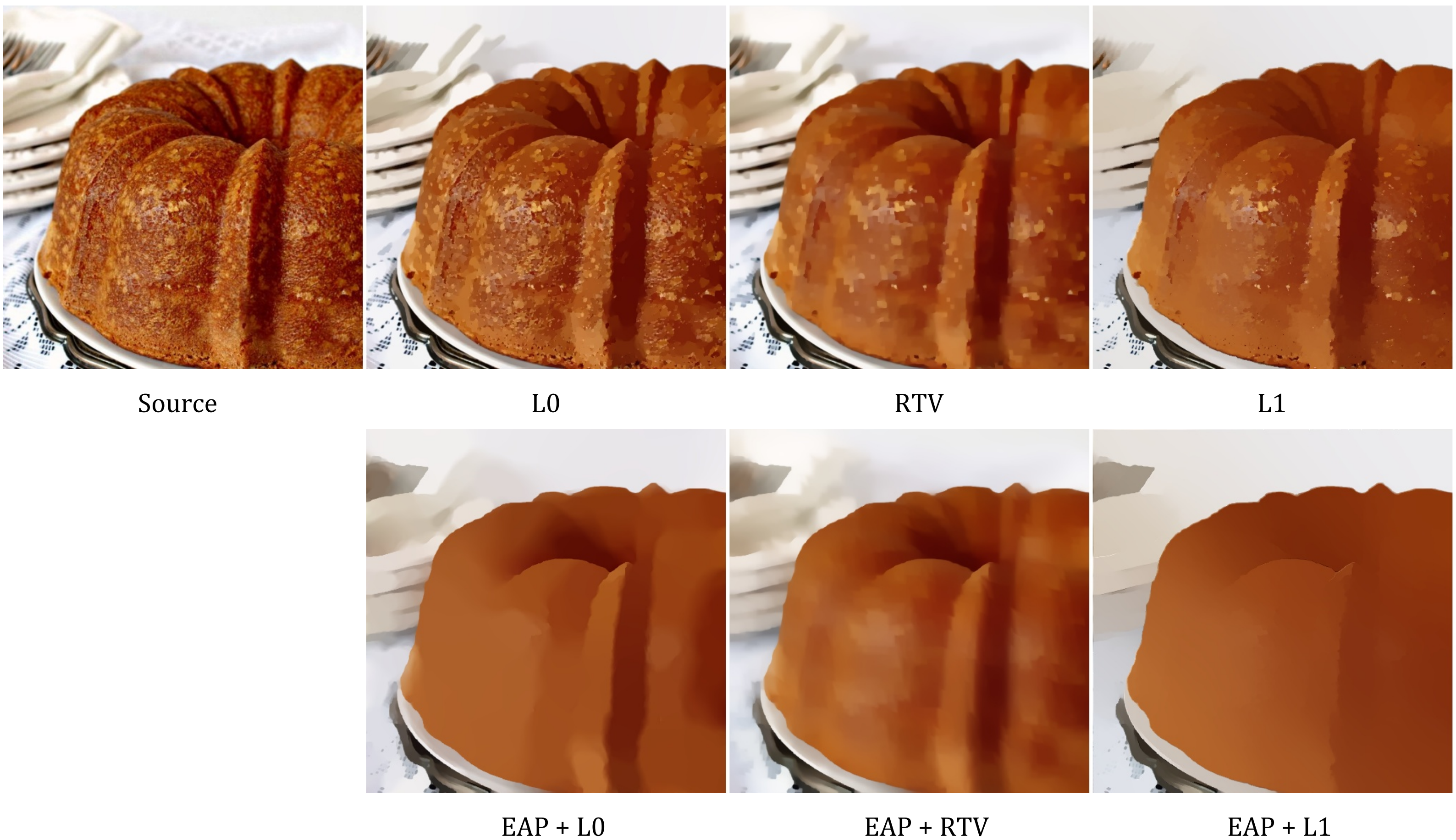

Figure 21: **Texture removal.** Visual comparison of texture removal with or without the EAP framework.

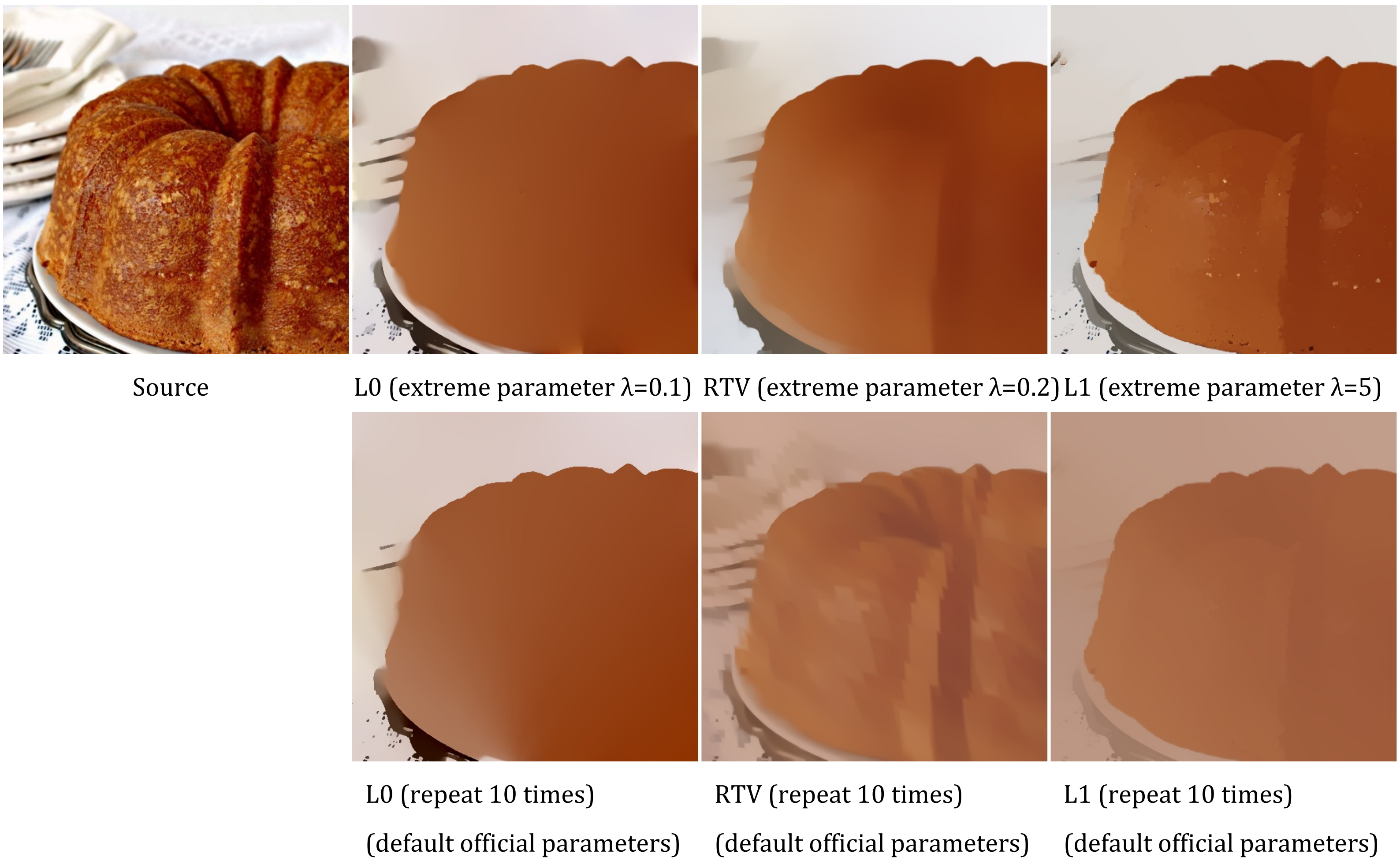

Figure 22: **Texture removal.** Trying to fine tune existing methods to facilitate adequate smoothing. The EAP results cannot be achieved by tuning parameters of existing methods or repeating existing methods for multiple times.

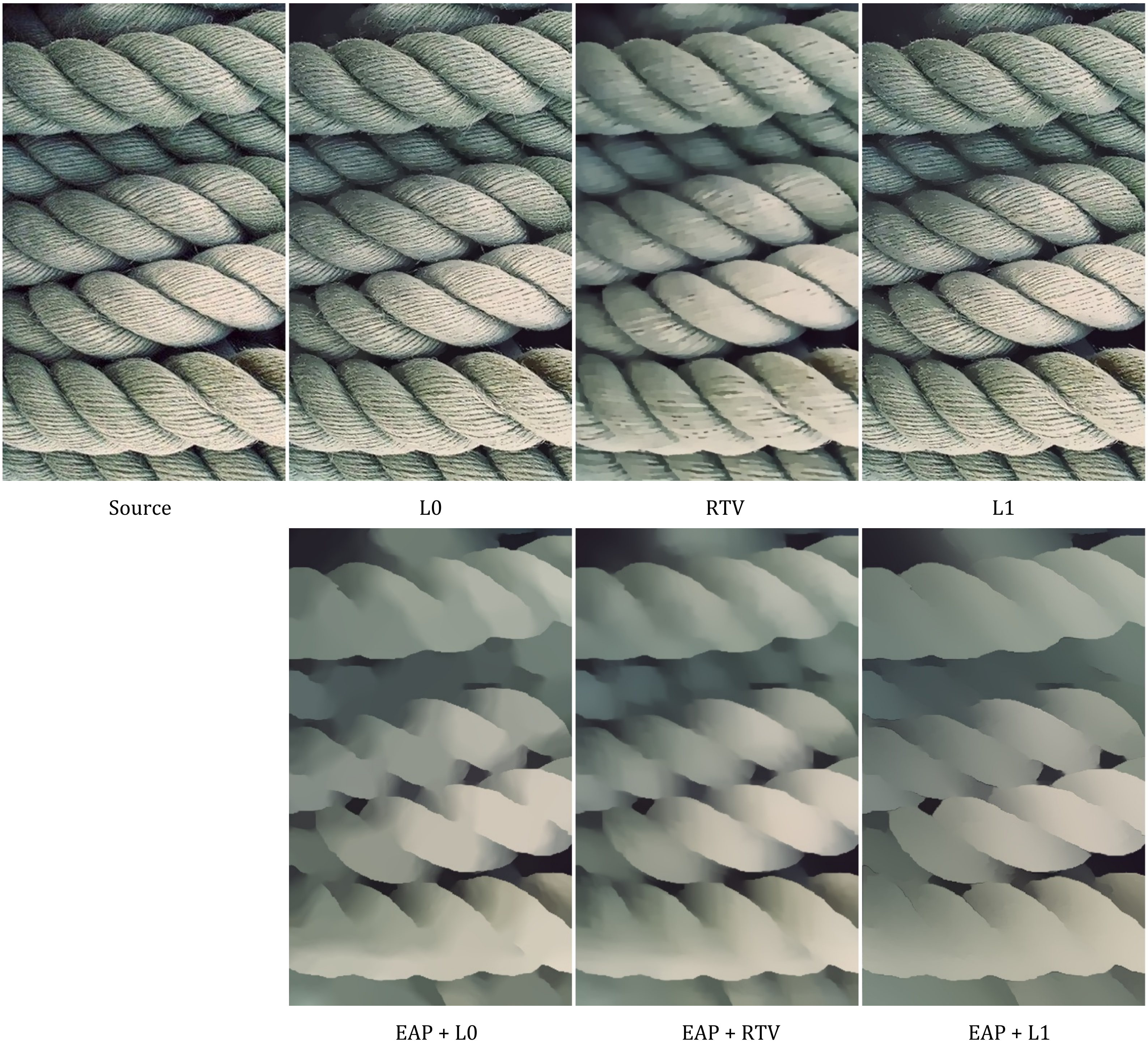

Figure 23: **Texture removal.** Visual comparison of texture removal with or without the EAP framework.

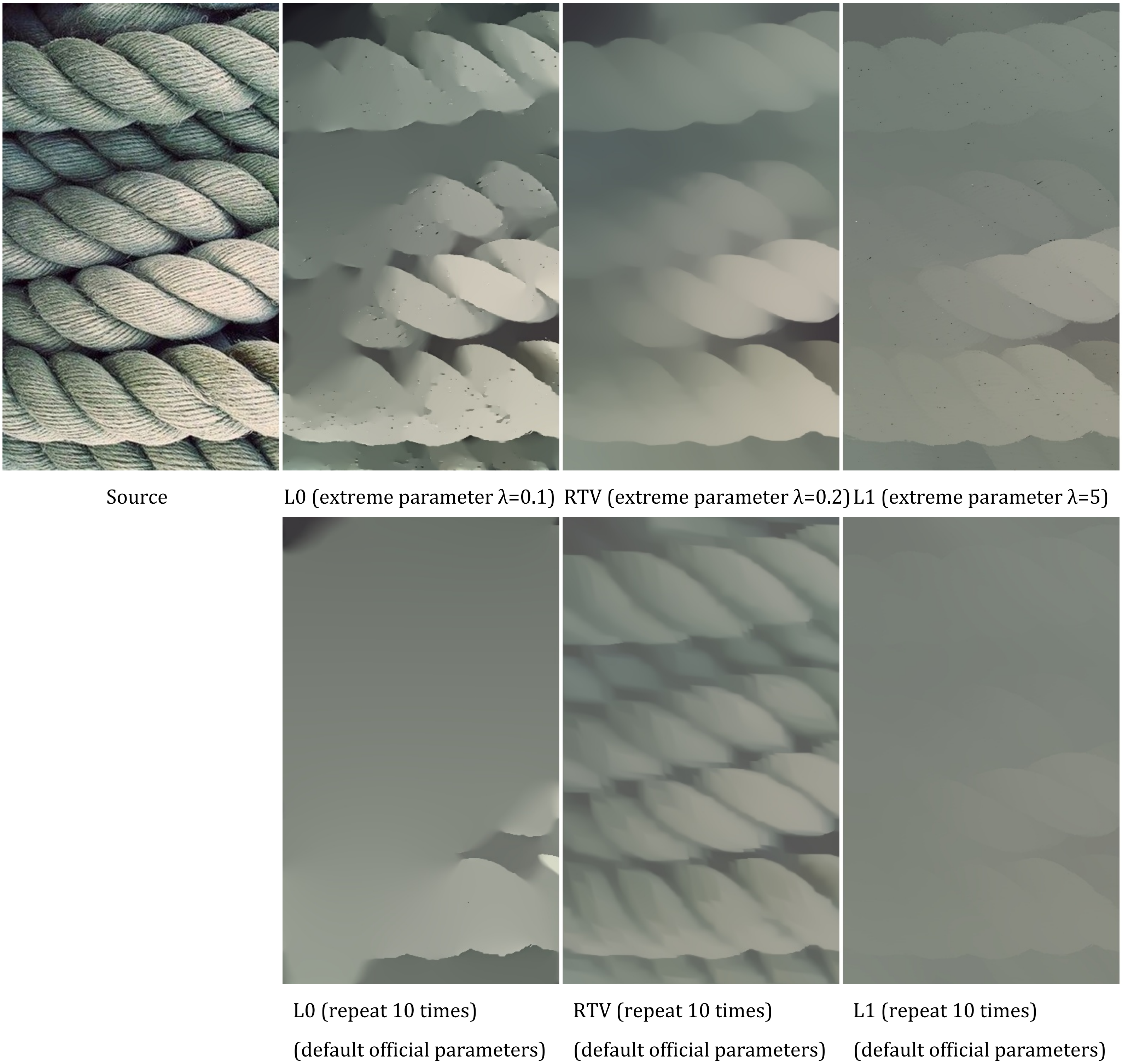

Figure 24: **Texture removal.** Trying to fine tune existing methods to facilitate adequate smoothing. The EAP results cannot be achieved by tuning parameters of existing methods or repeating existing methods for multiple times.

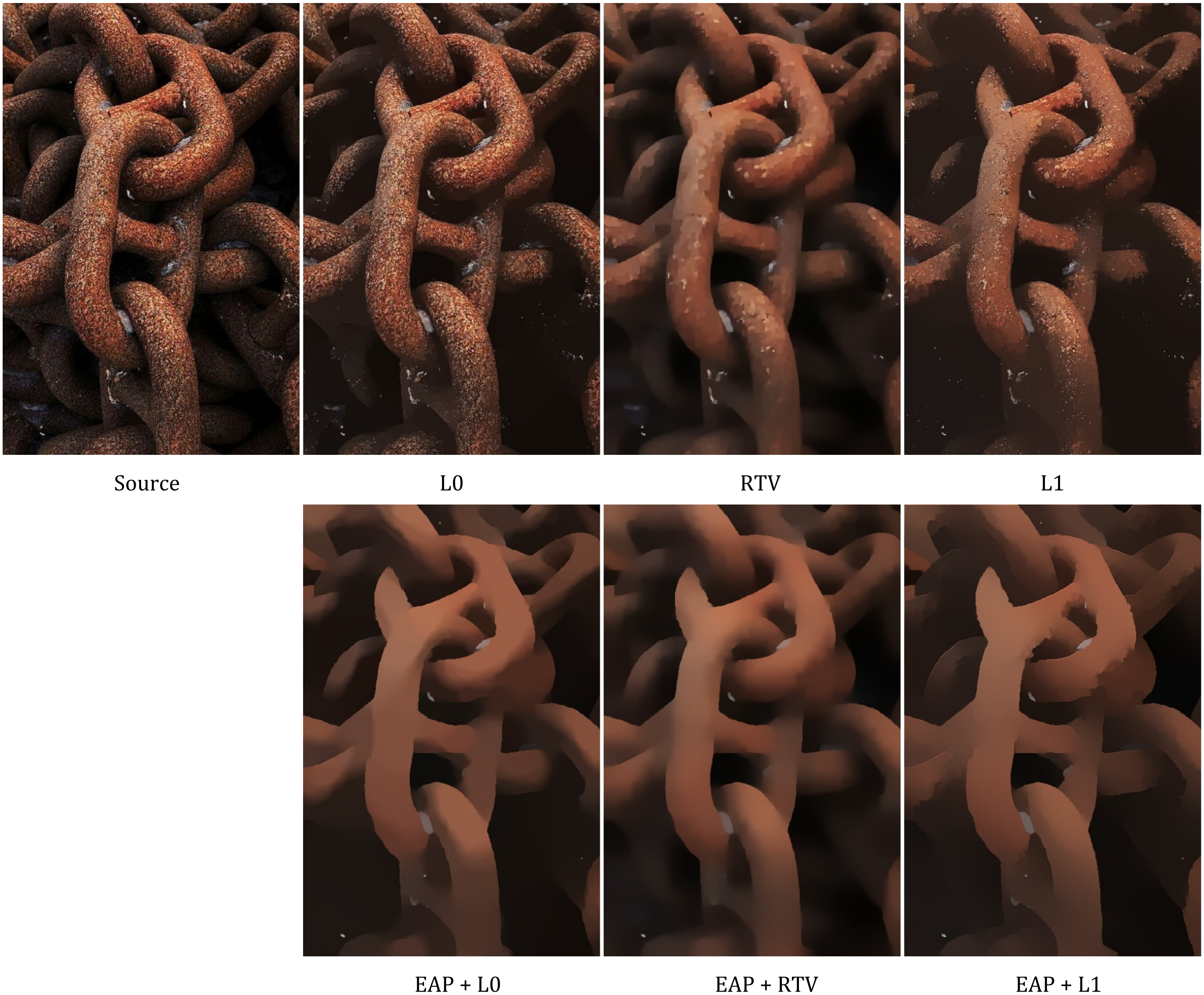

Figure 25: **Texture removal.** Visual comparison of texture removal with or without the EAP framework.

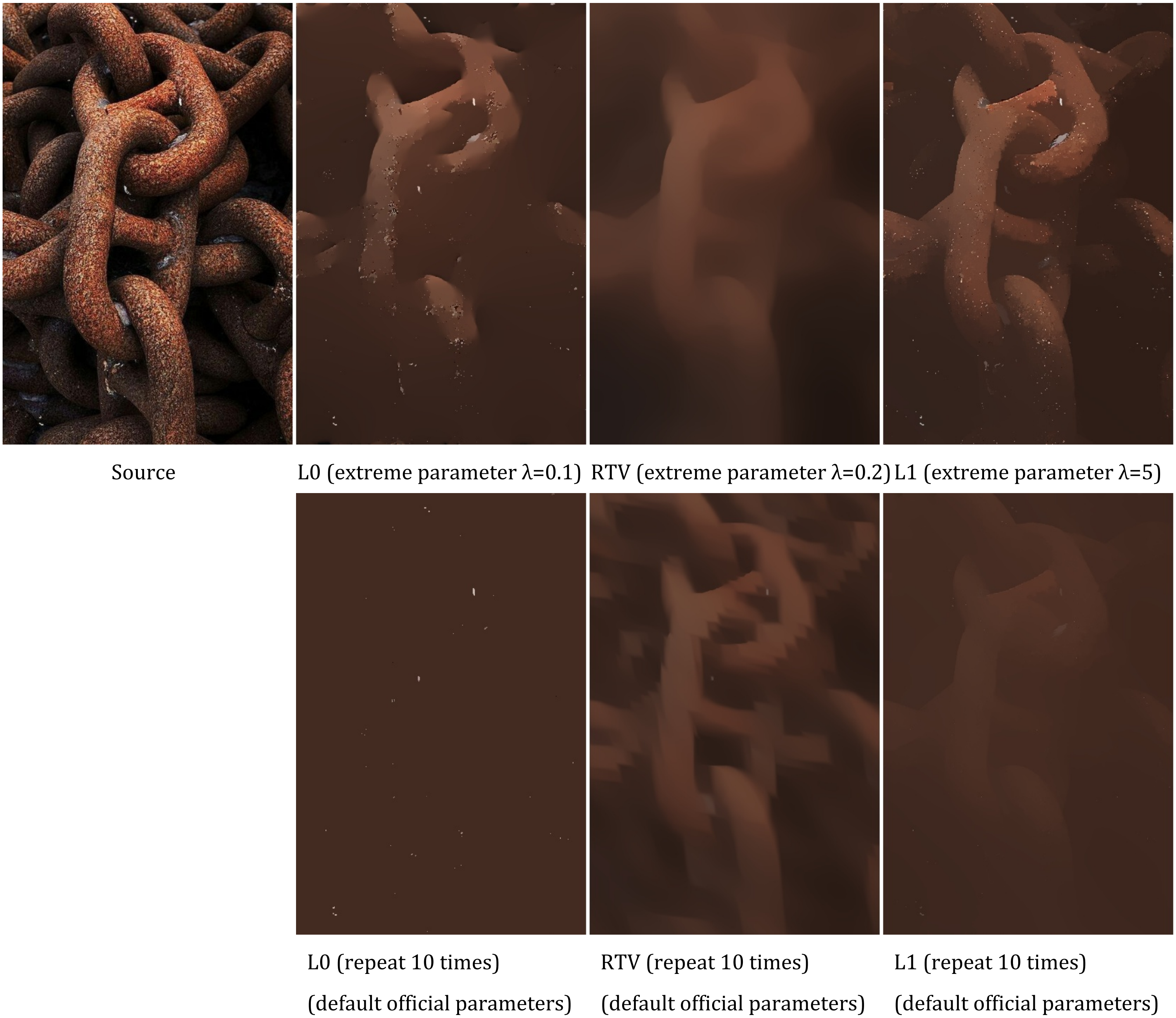

Figure 26: **Texture removal.** Trying to fine tune existing methods to facilitate adequate smoothing. The EAP results cannot be achieved by tuning parameters of existing methods or repeating existing methods for multiple times.

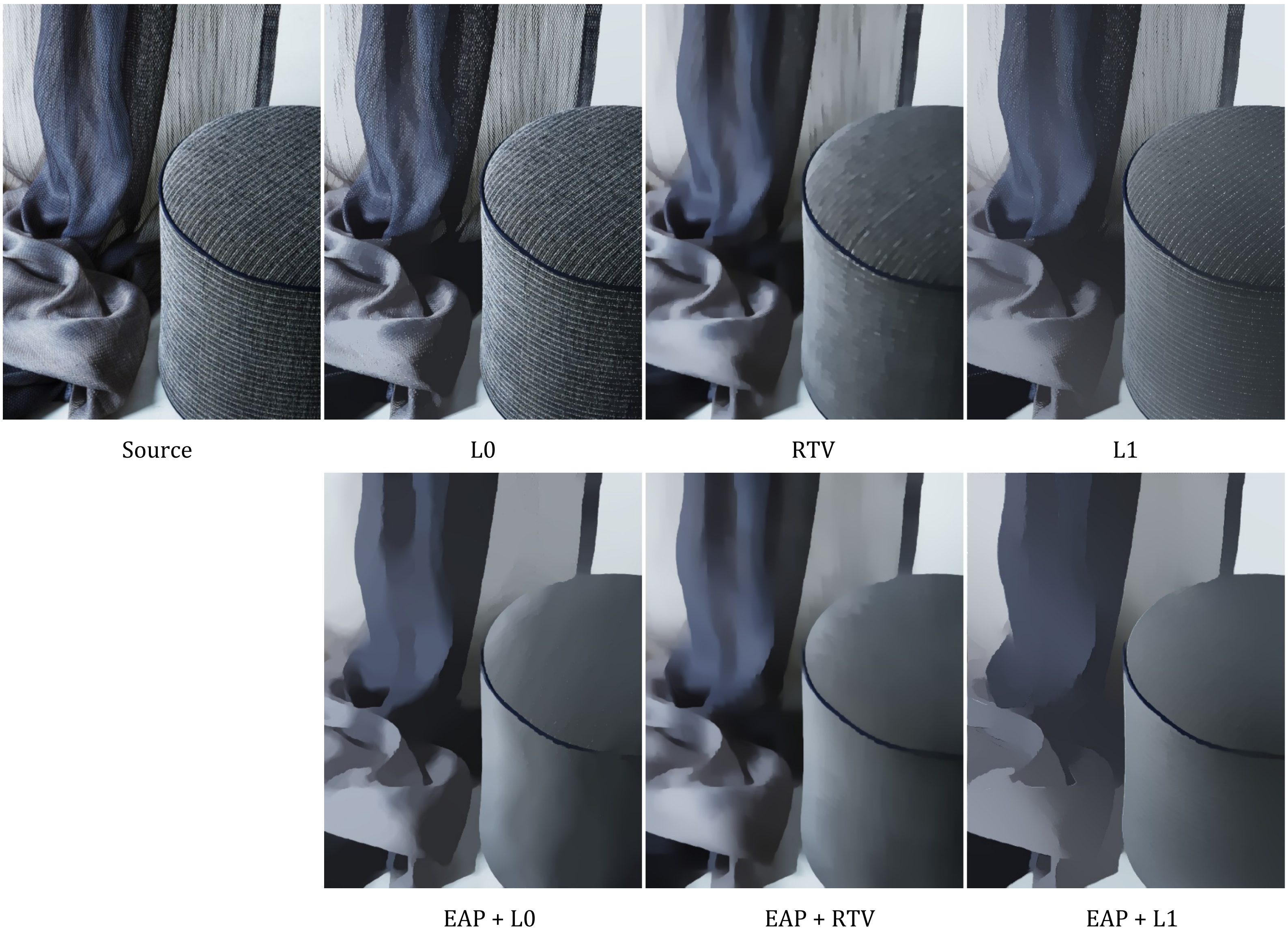

Figure 27: **Texture removal.** Visual comparison of texture removal with or without the EAP framework.

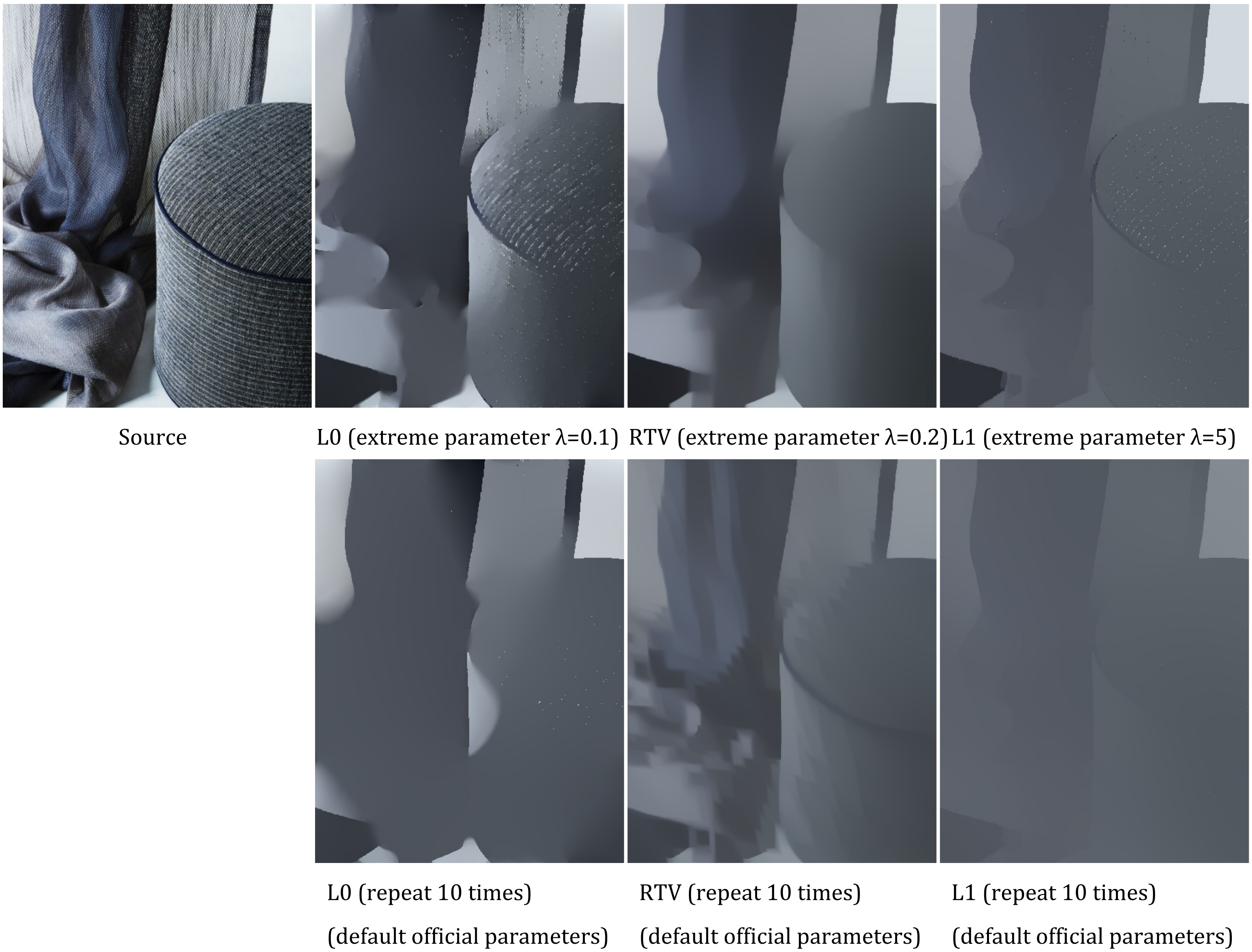

Figure 28: **Texture removal.** Trying to fine tune existing methods to facilitate adequate smoothing. The EAP results cannot be achieved by tuning parameters of existing methods or repeating existing methods for multiple times.

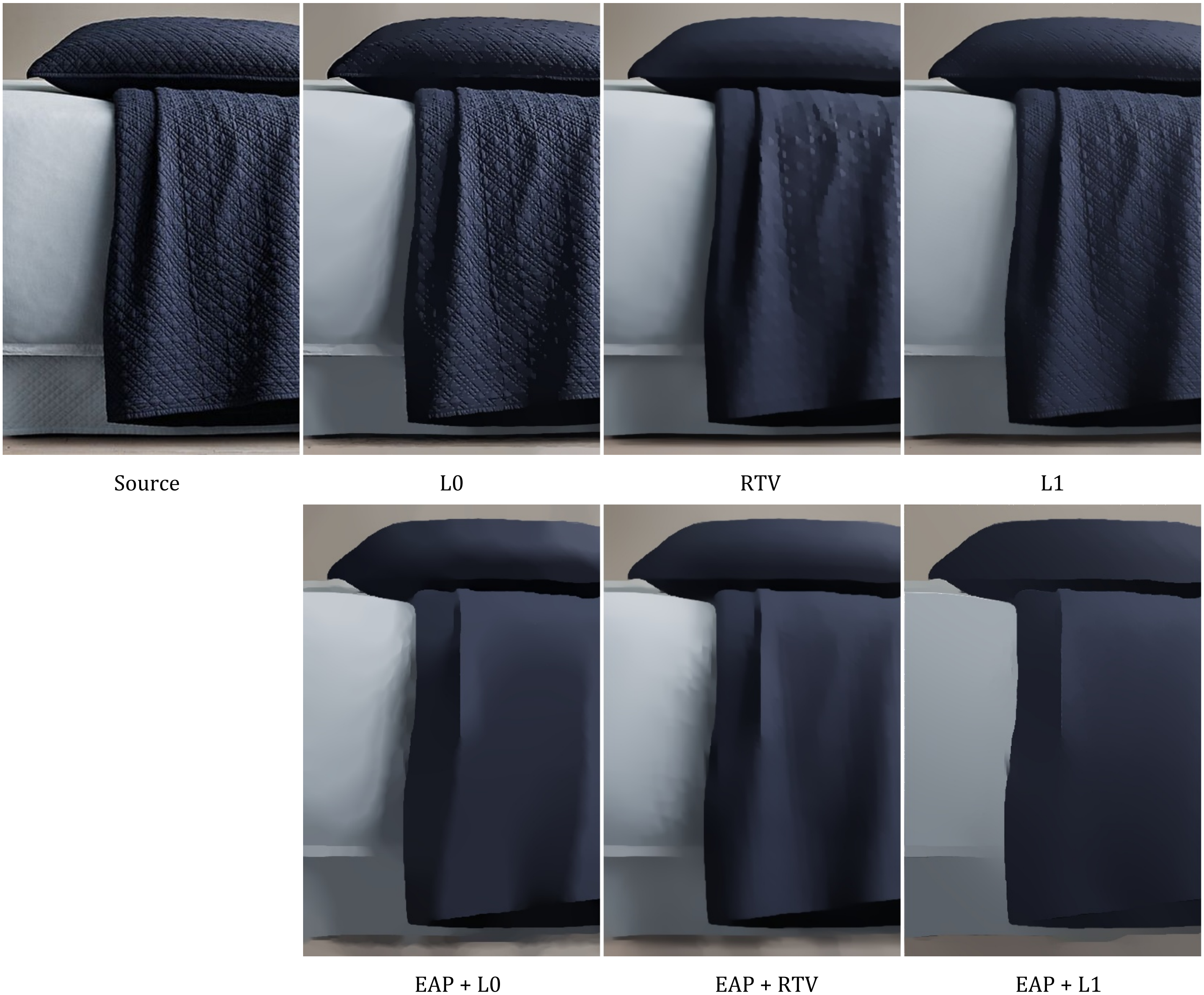

Figure 29: **Texture removal.** Visual comparison of texture removal with or without the EAP framework.

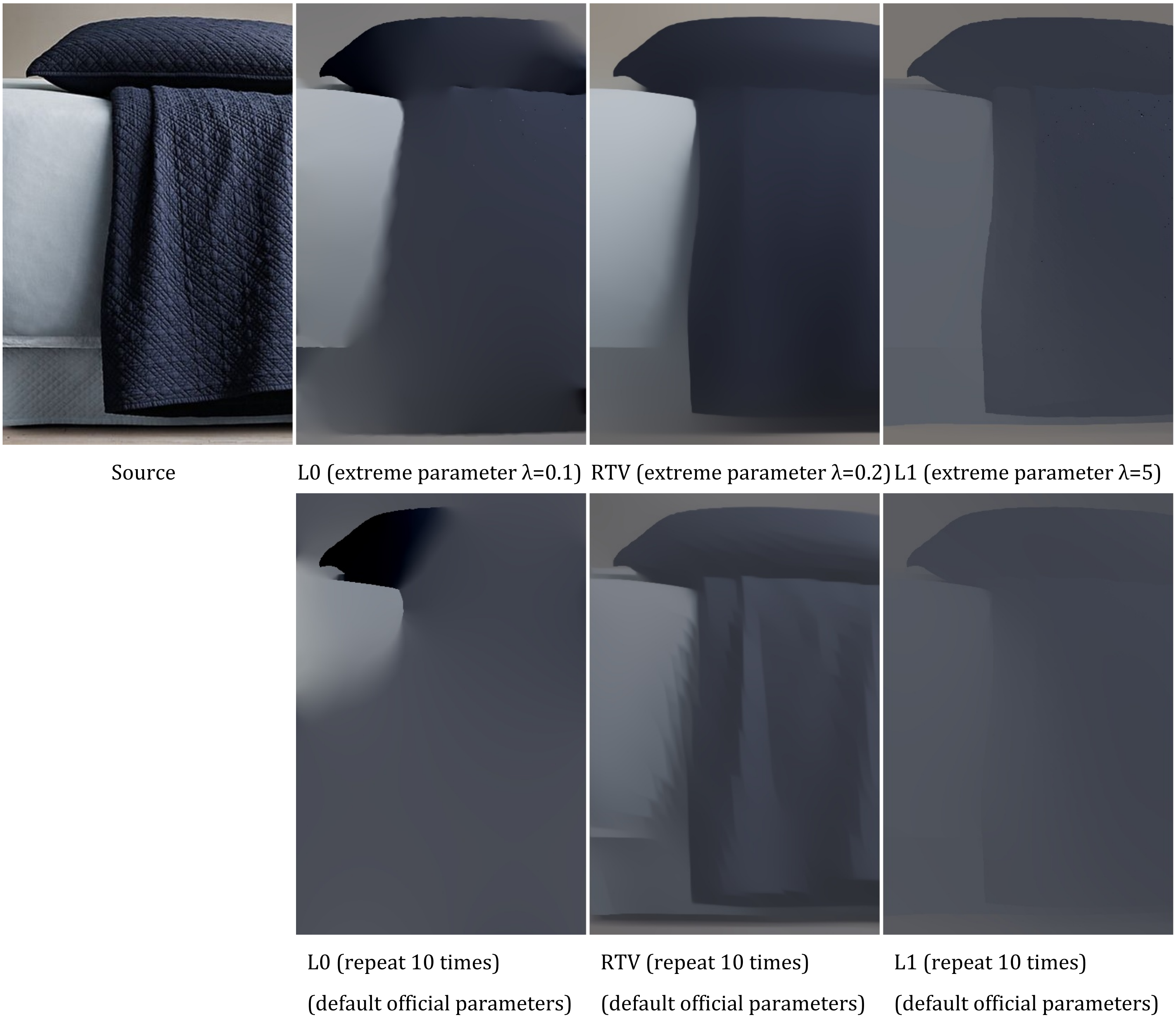

Figure 30: **Texture removal.** Trying to fine tune existing methods to facilitate adequate smoothing. The EAP results cannot be achieved by tuning parameters of existing methods or repeating existing methods for multiple times.

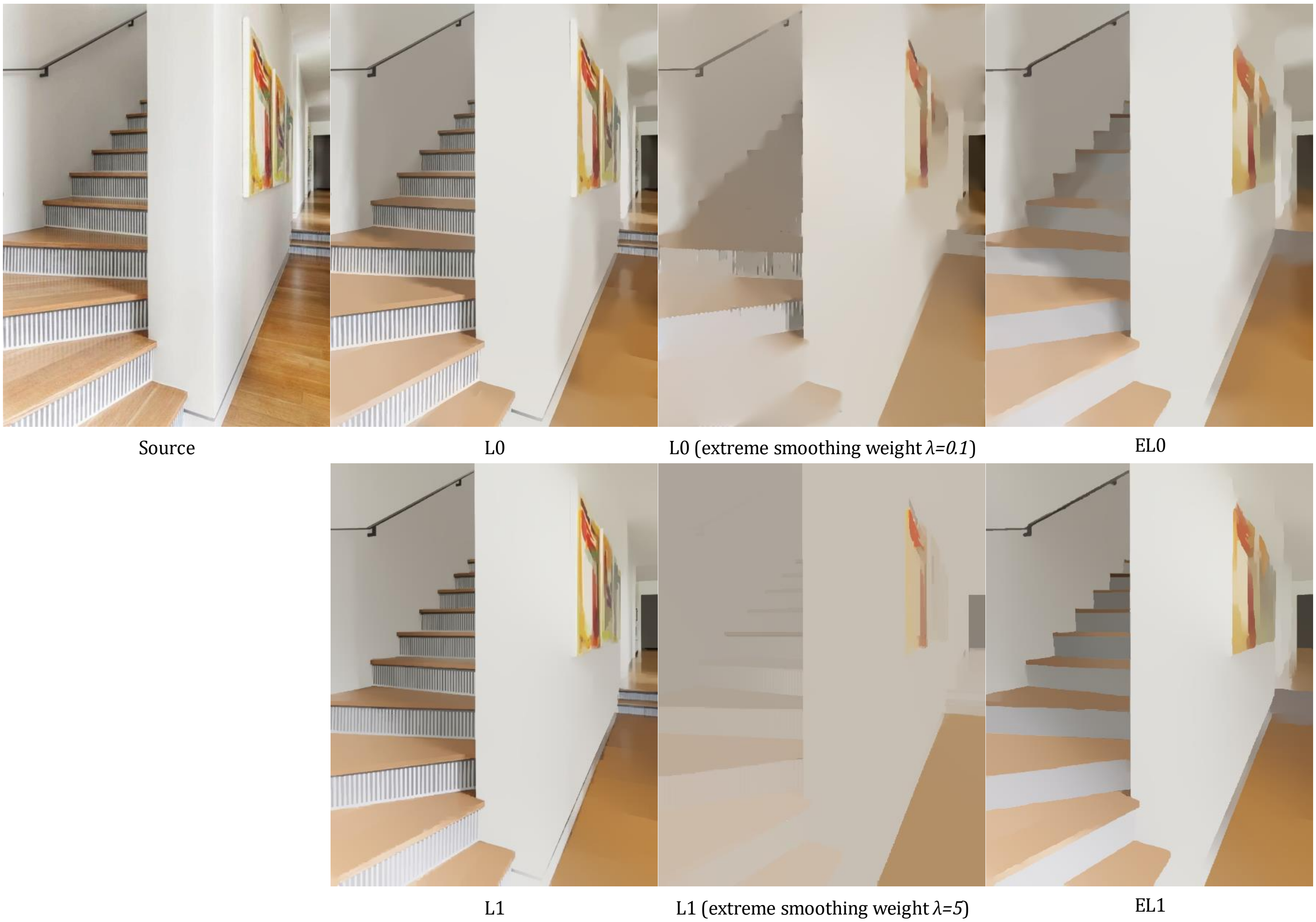

Figure 31: **Texture removal.** The visual effect of EAP cannot be achieved by tuning smoothing parameters of previous methods. We provide several smoothed results using previous methods and extreme parameters. The EAP achieves results significantly better than that from extremely tuned previous methods.

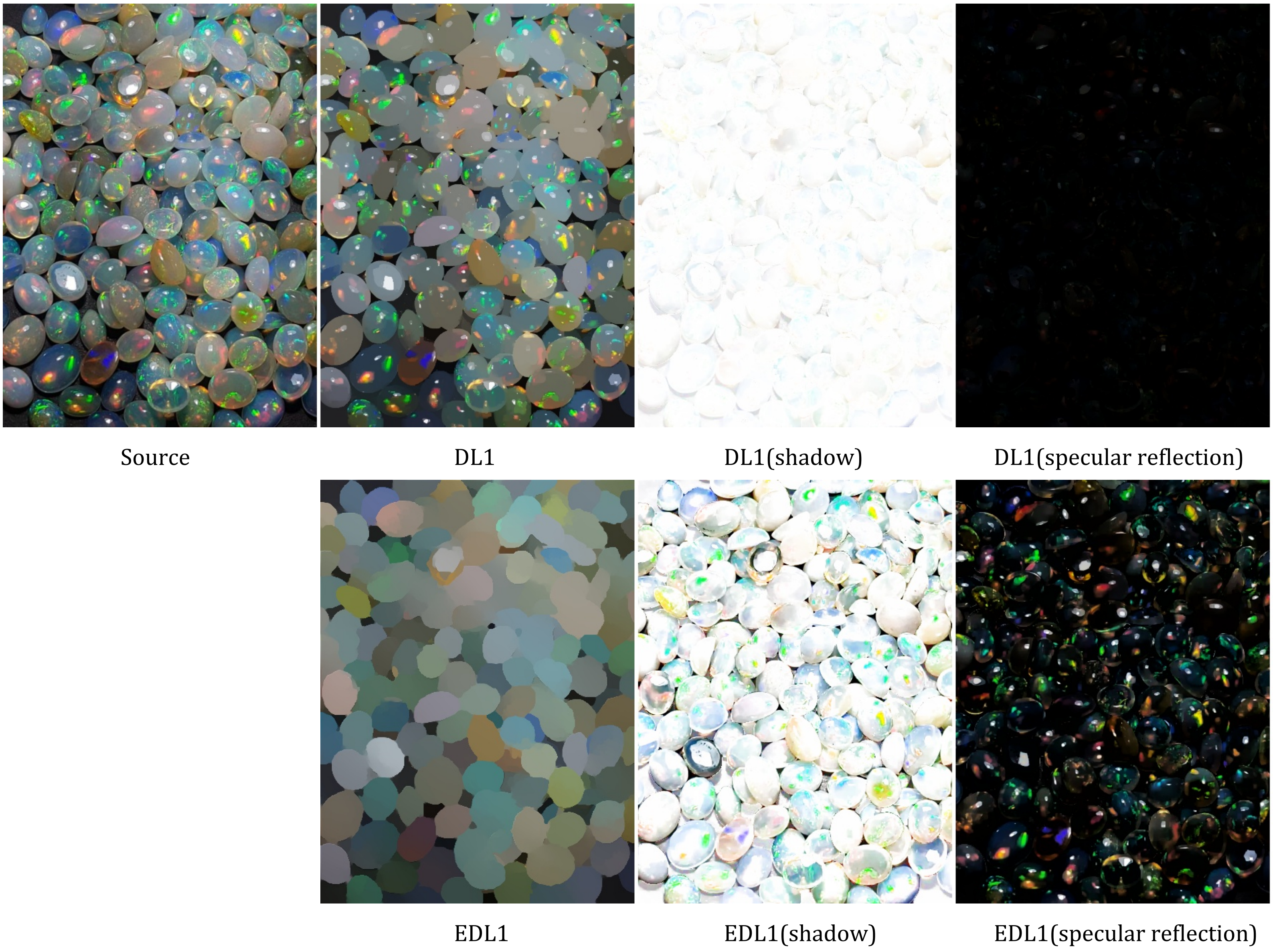

Figure 32: **Layer decomposition.** Visual comparison of layer decomposition with or without the EAP framework.

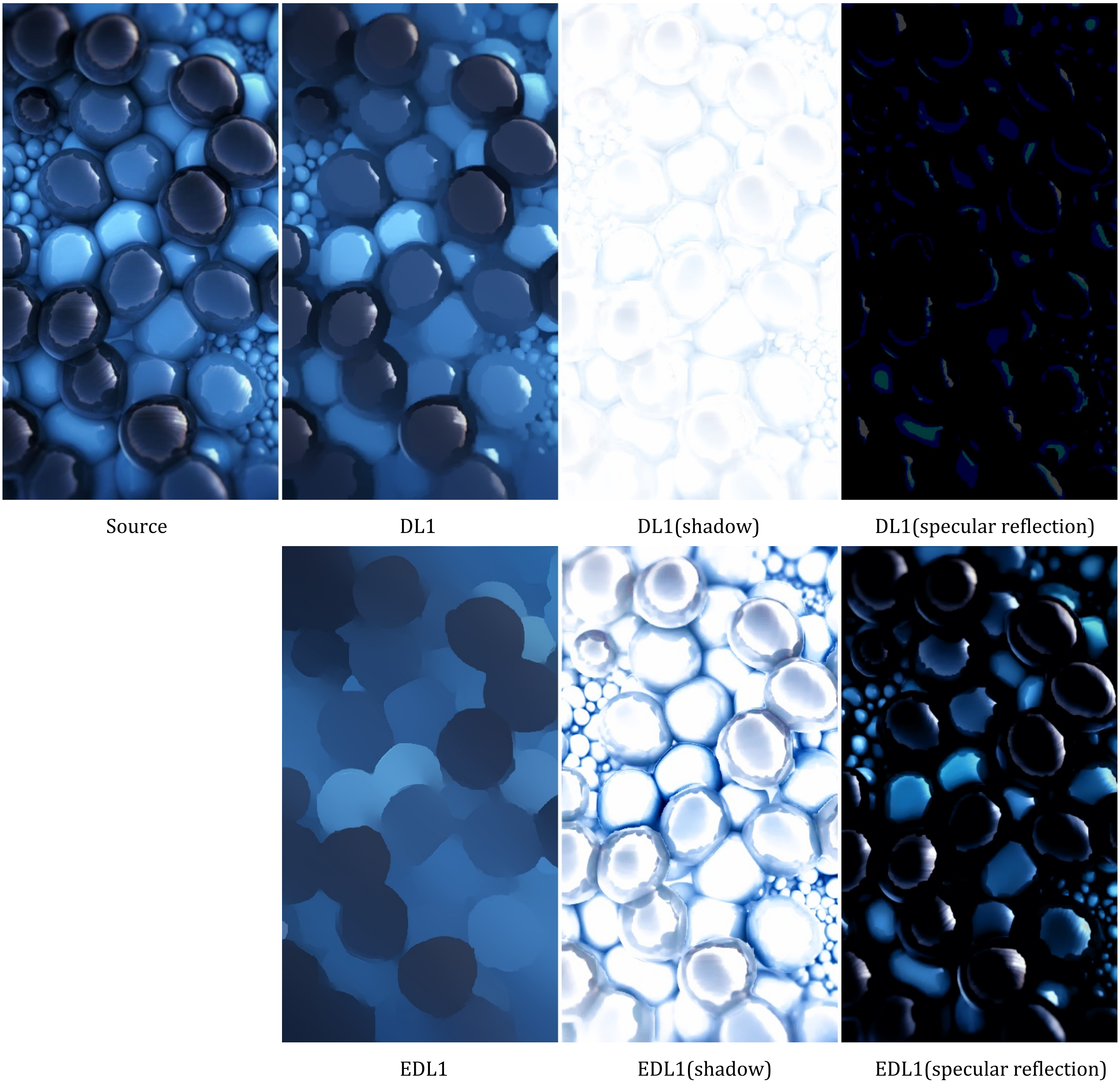

Figure 33: **Layer decomposition.** Visual comparison of layer decomposition with or without the EAP framework.

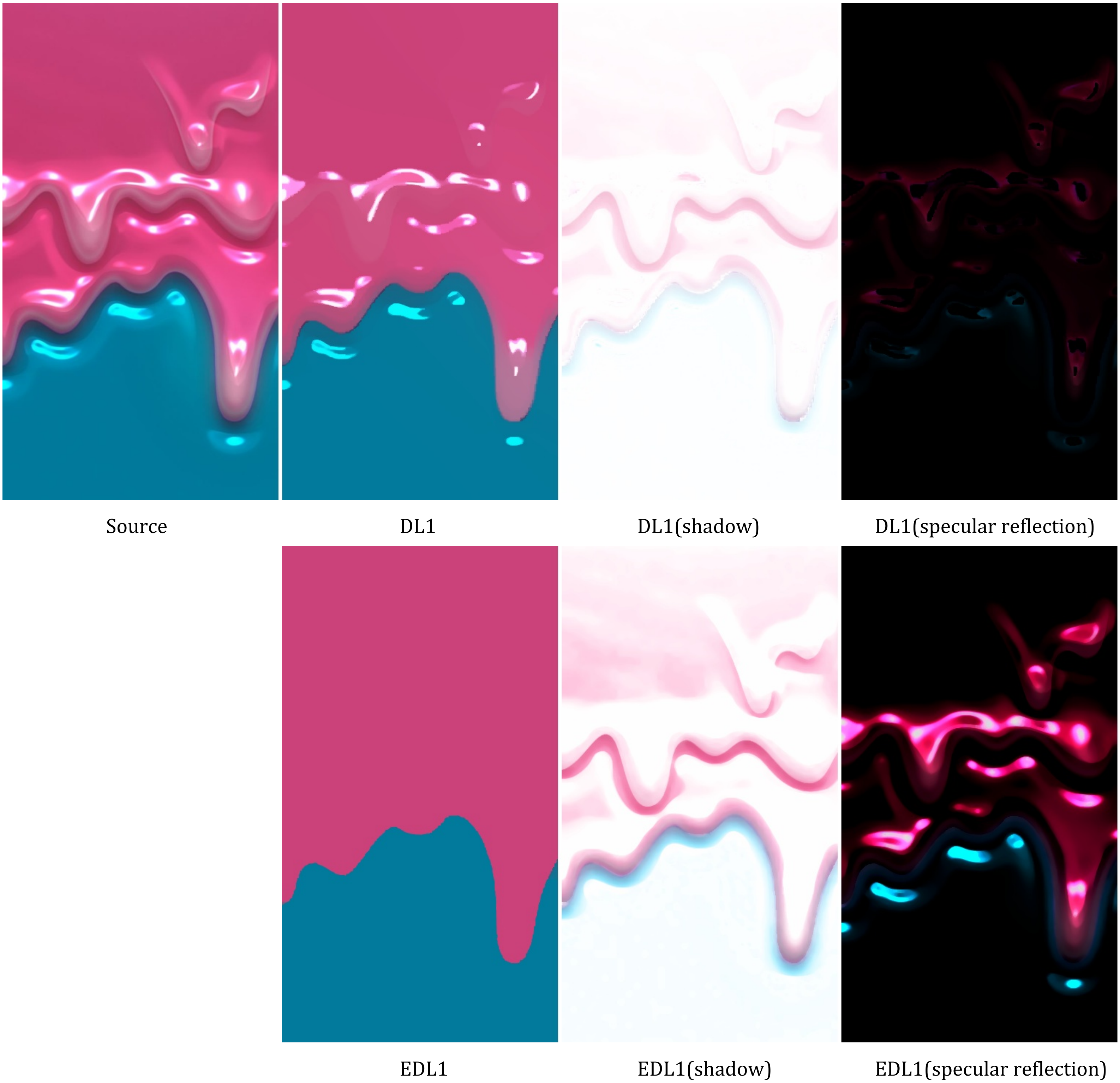

Figure 34: **Layer decomposition.** Visual comparison of layer decomposition with or without the EAP framework.

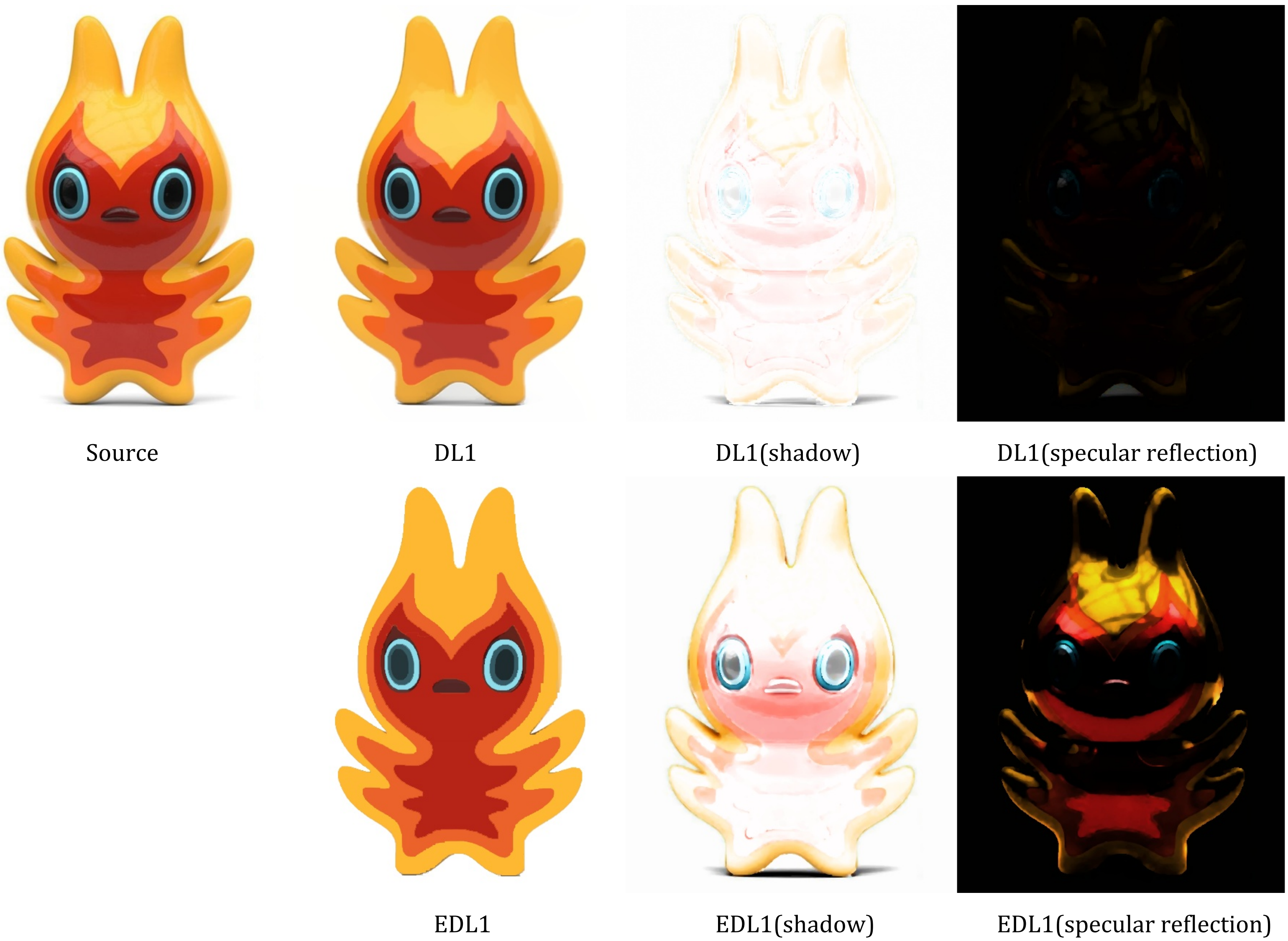

Figure 35: **Layer decomposition.** Visual comparison of layer decomposition with or without the EAP framework.

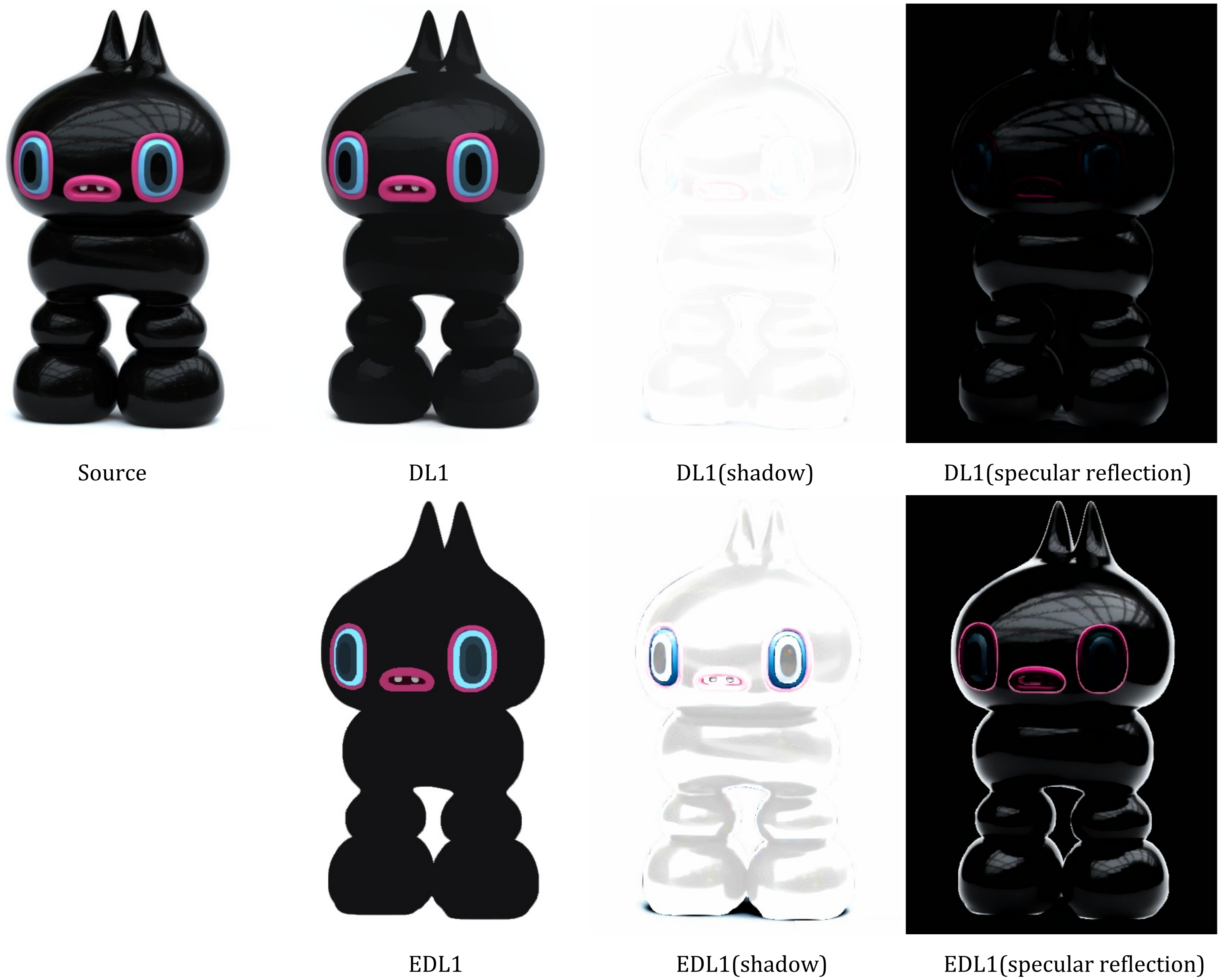

Figure 36: **Layer decomposition.** Visual comparison of layer decomposition with or without the EAP framework.

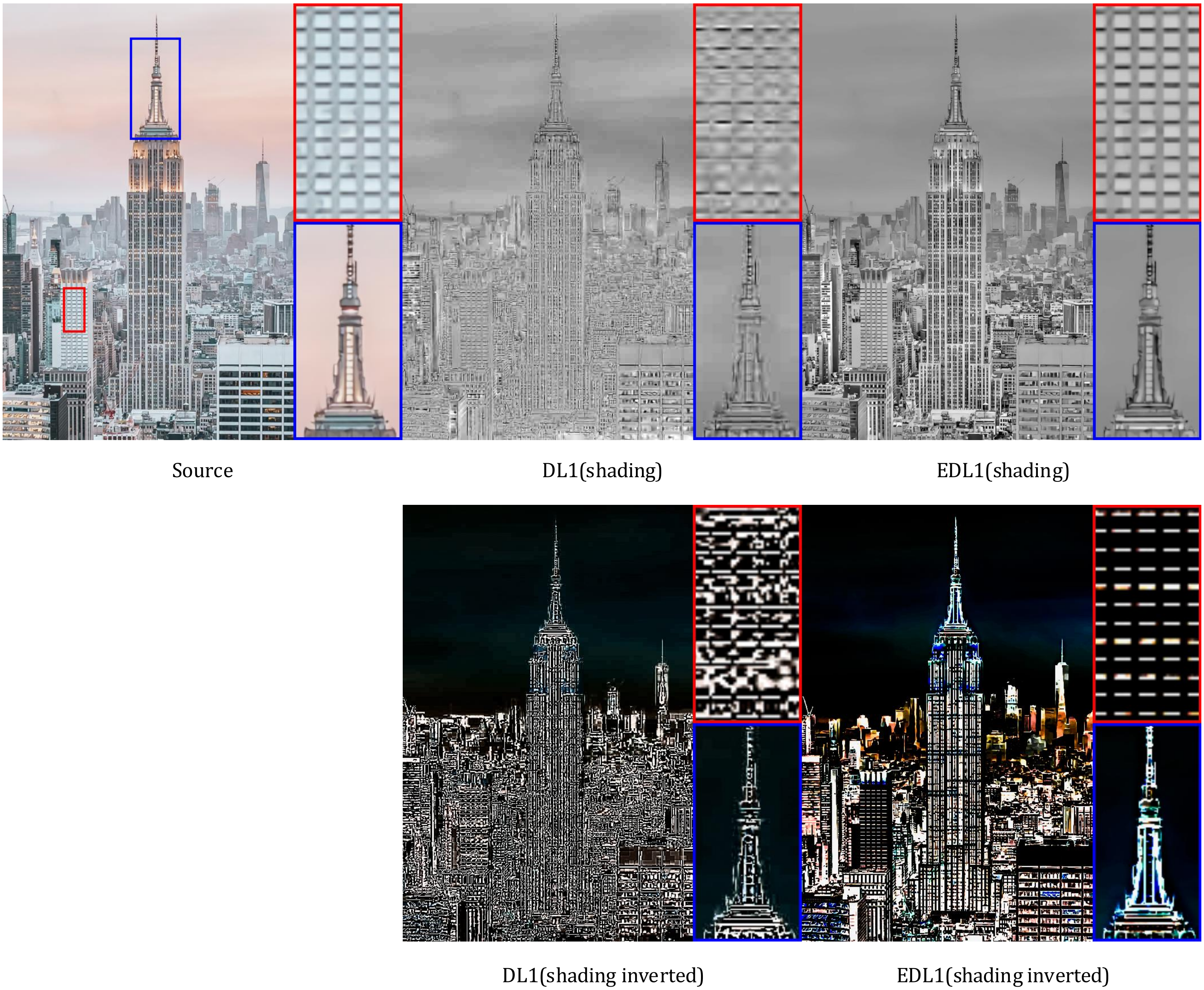

Figure 37: **Illumination manipulation.** Visual comparison of gamma corrected shading map inverting with or without the EAP framework.

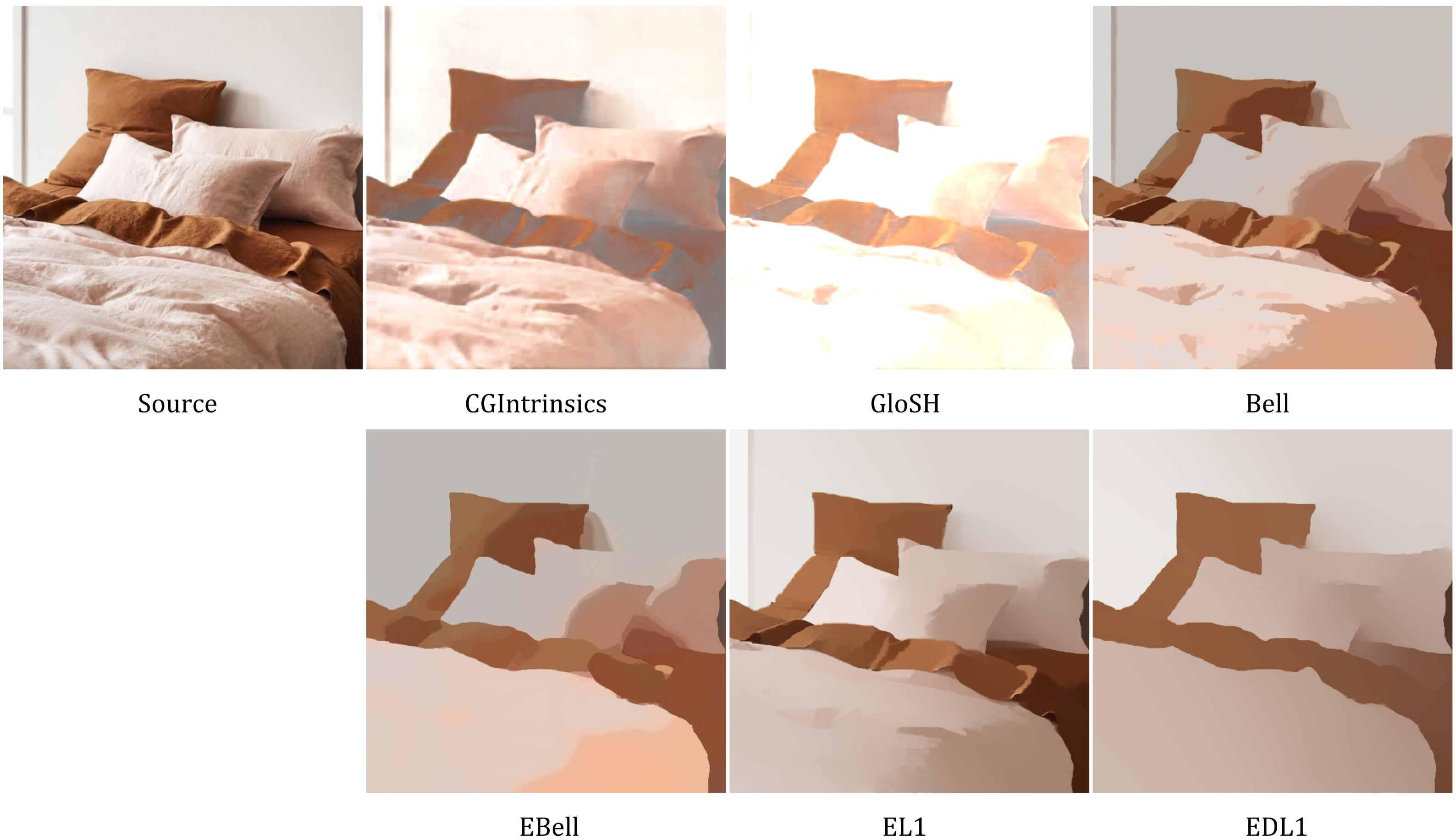

Figure 38: **Intrinsic decomposition.** Visual comparison of intrinsic reflectance extraction with or without the EAP framework, **without** hue/saturation constraint. (The reflectance is **not** required to have same pixel hue and saturation with source images.)

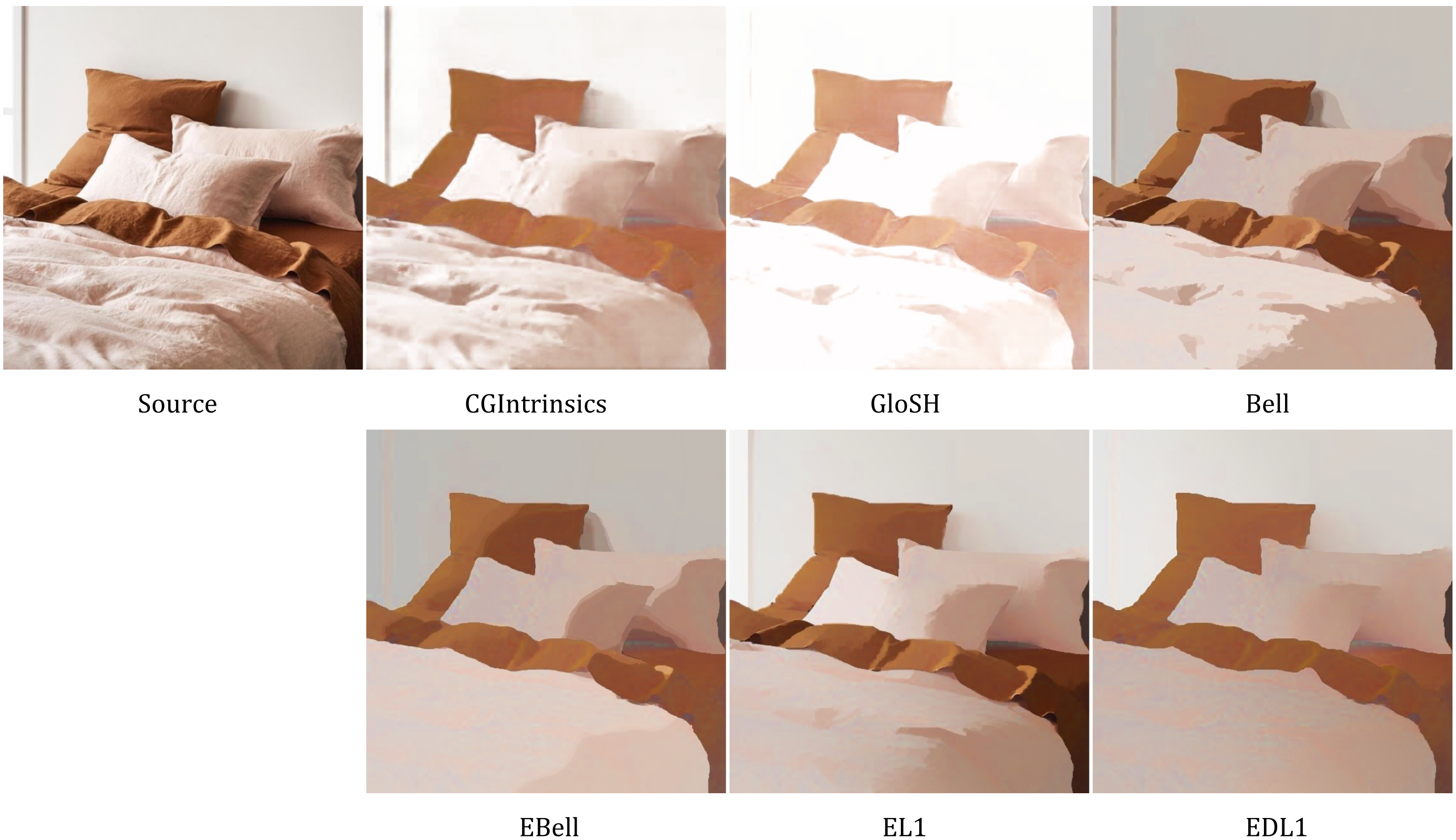

Figure 39: **Intrinsic decomposition.** Visual comparison of intrinsic reflectance extraction with or without the EAP framework, **with** hue/saturation constraint. (The reflectance is required to have same pixel hue and saturation with source images. The hue and saturation channels in all reflectance maps are replaced by the original hue and saturation in the source image.)

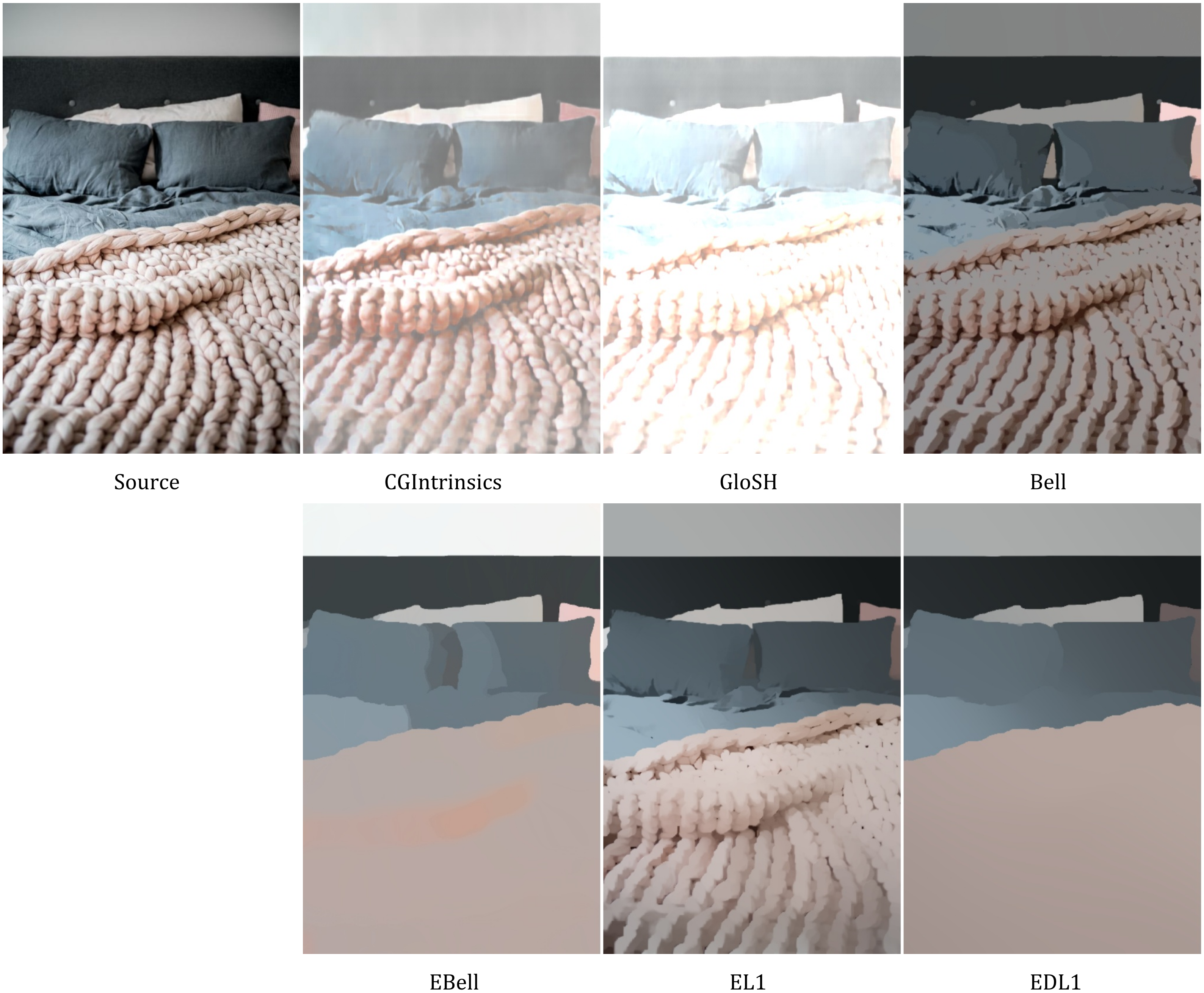

Figure 40: **Intrinsic decomposition.** Visual comparison of intrinsic reflectance extraction with or without the EAP framework, **without** hue/saturation constraint. (The reflectance is **not** required to have same pixel hue and saturation with source images.)

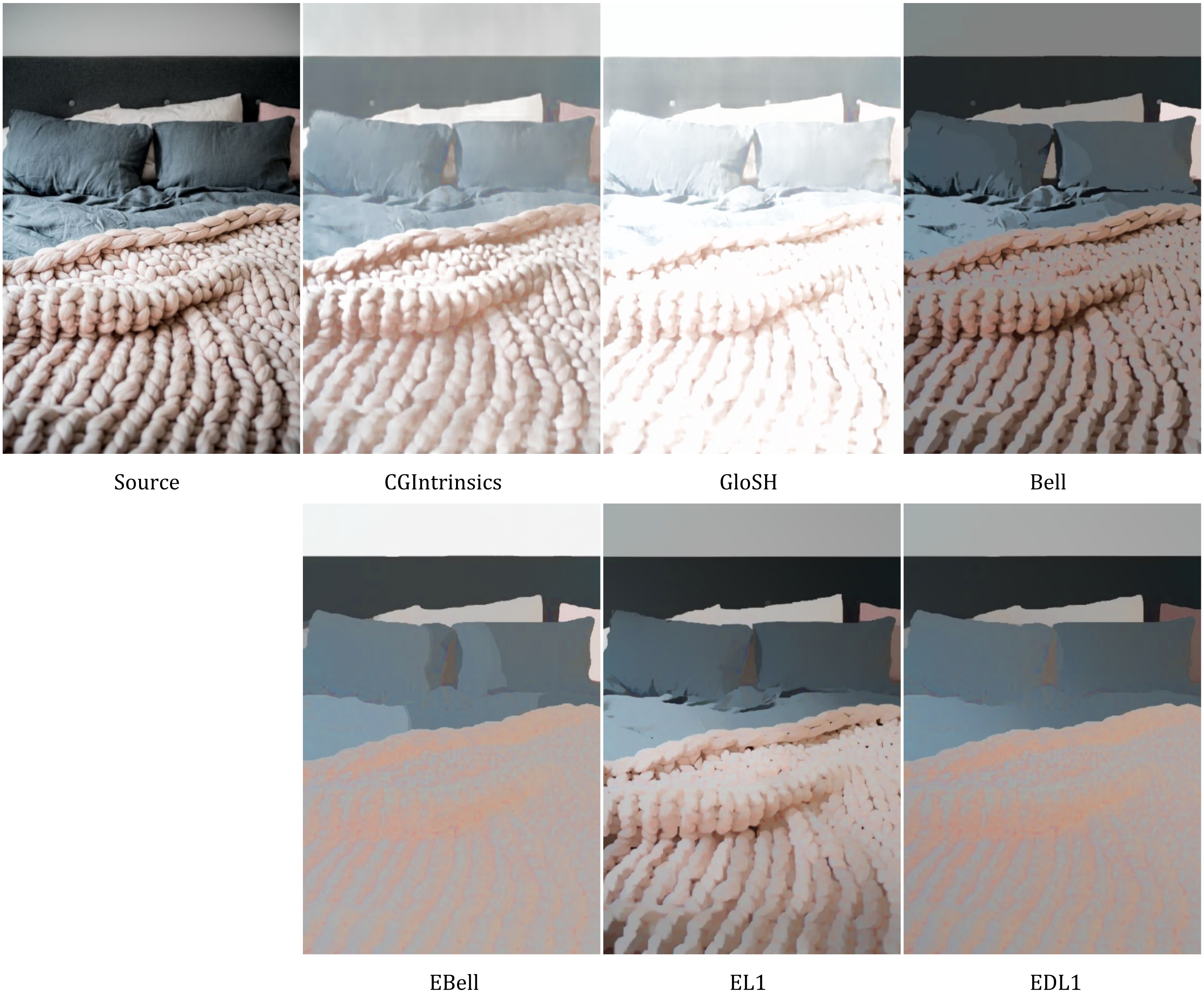

Figure 41: **Intrinsic decomposition.** Visual comparison of intrinsic reflectance extraction with or without the EAP framework, **with** hue/saturation constraint. (The reflectance is required to have same pixel hue and saturation with source images. The hue and saturation channels in all reflectance maps are replaced by the original hue and saturation in the source image.)

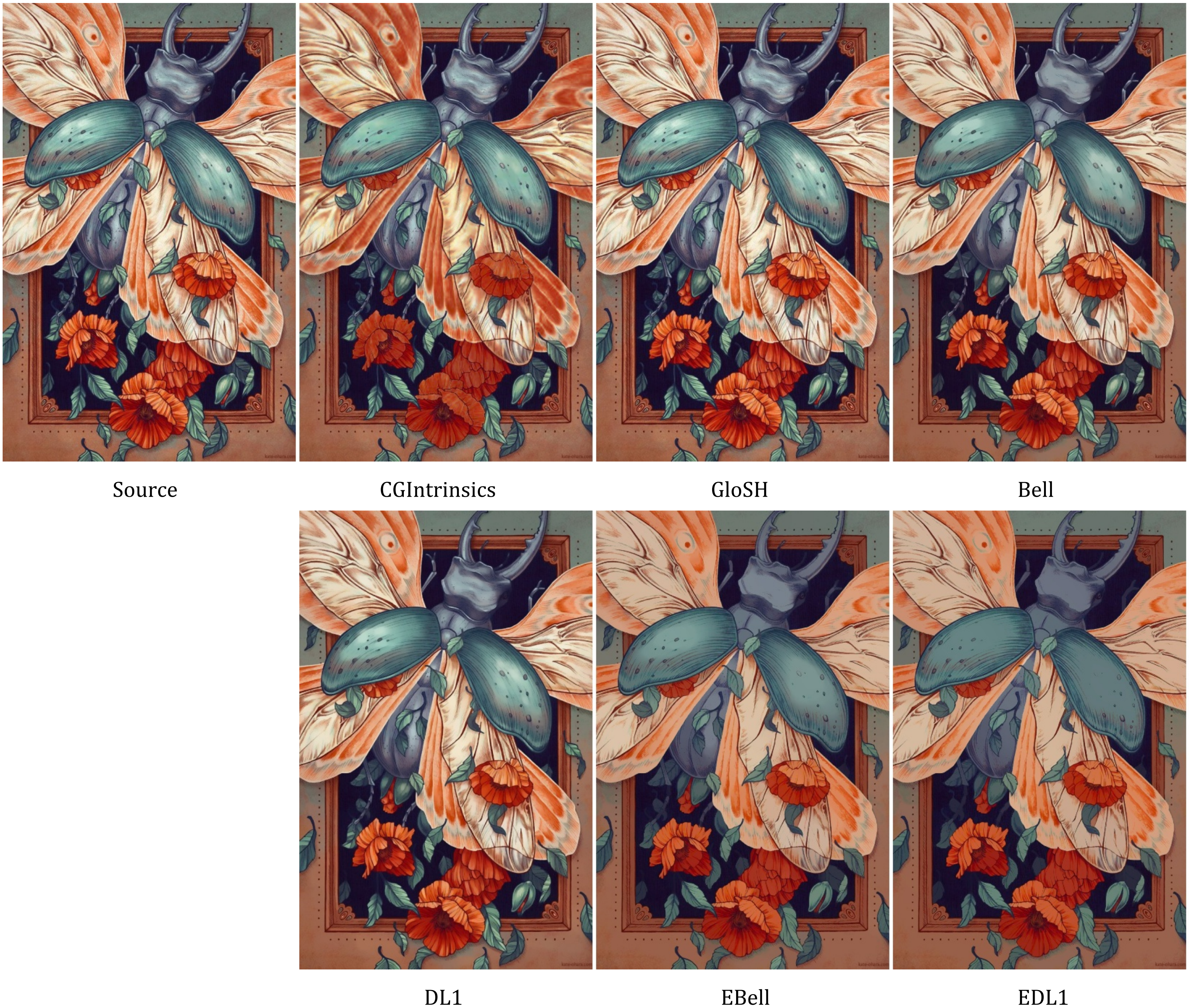

Figure 42: **Specular reflection removal.** Visual comparison of specular reflection removal using different intrinsic decomposition methods.

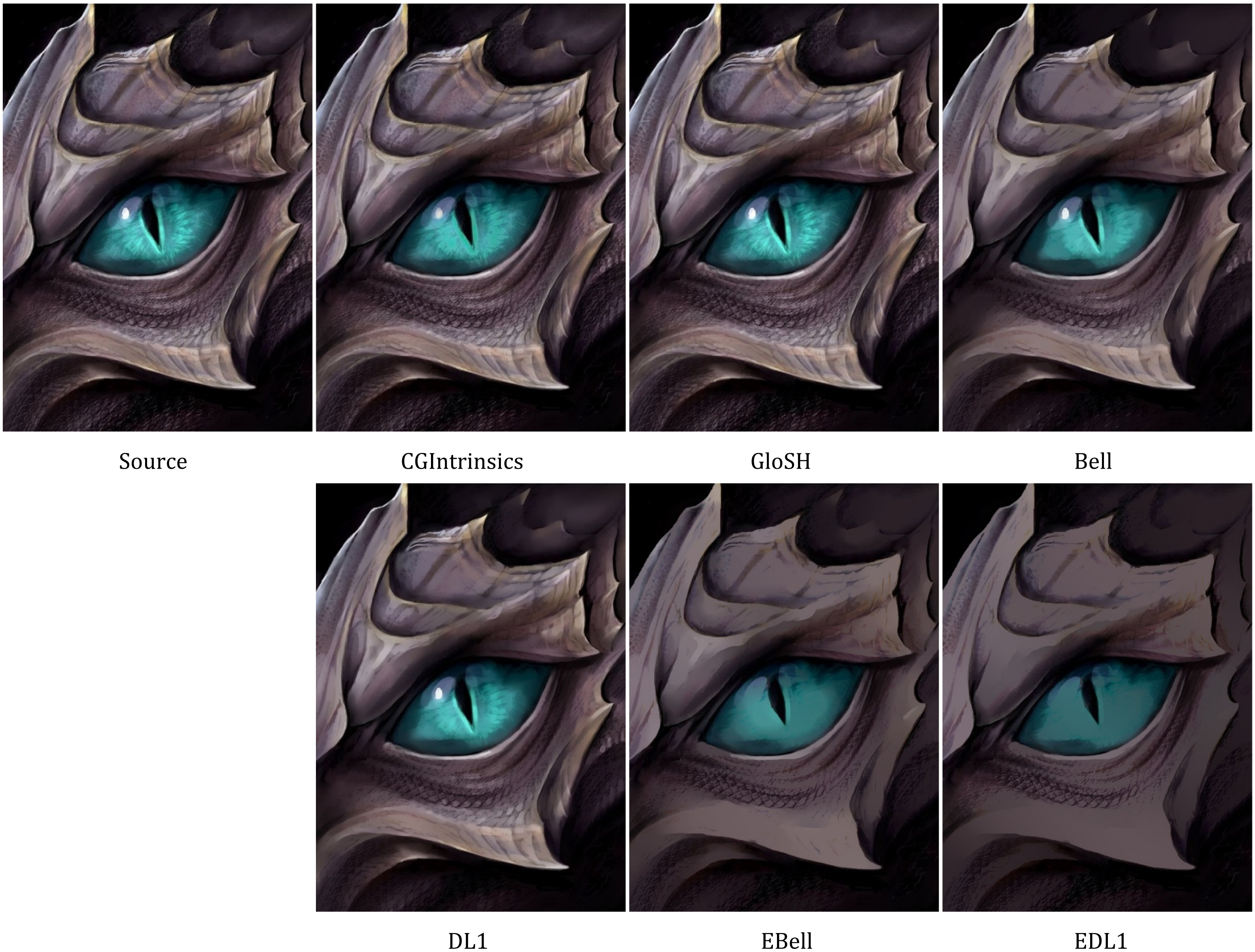

Figure 43: **Specular reflection removal.** Visual comparison of specular reflection removal using different intrinsic decomposition methods.

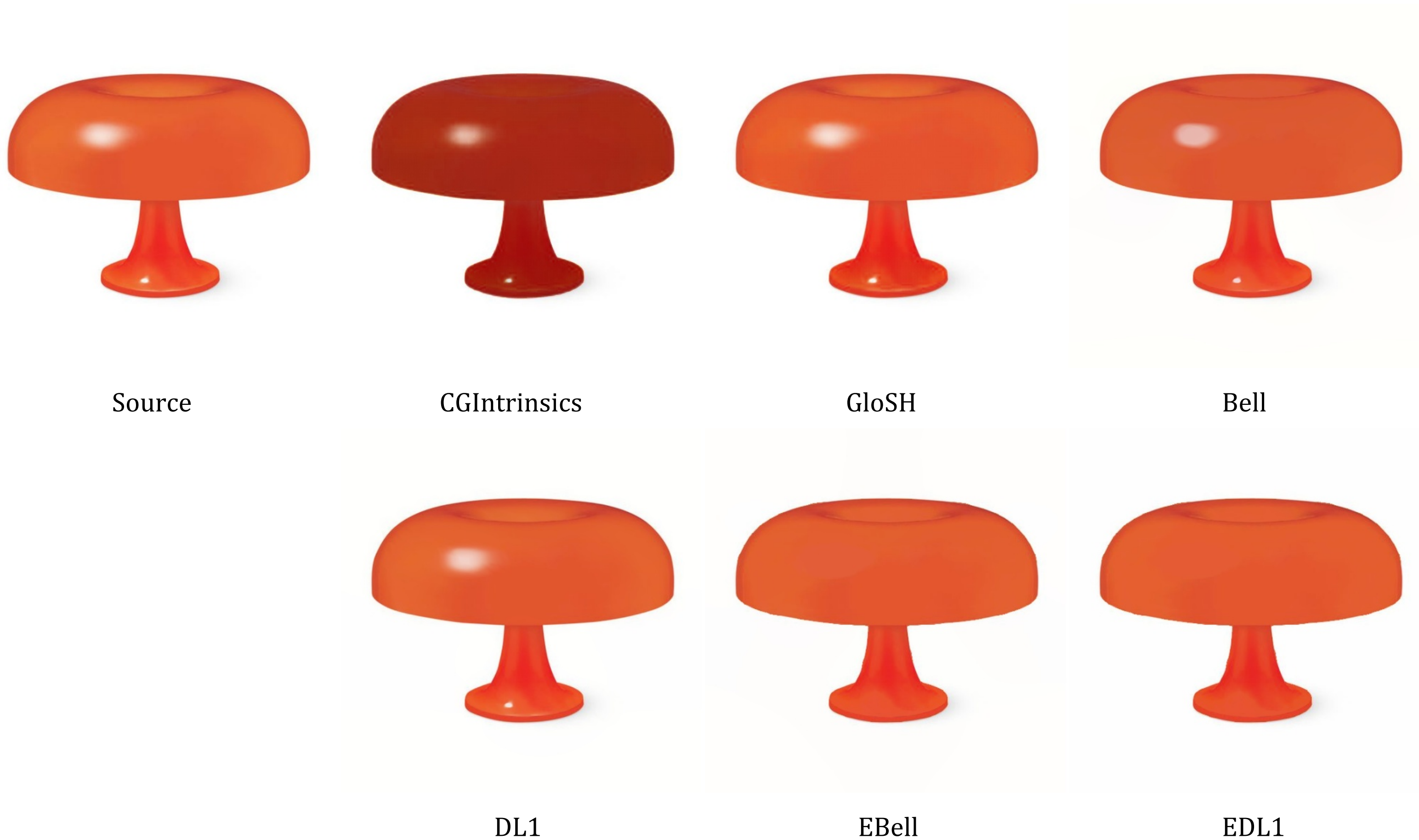

Figure 44: **Specular reflection removal.** Visual comparison of specular reflection removal using different intrinsic decomposition methods.

Figure 45: **Specular reflection removal.** Visual comparison of specular reflection removal using different intrinsic decomposition methods.

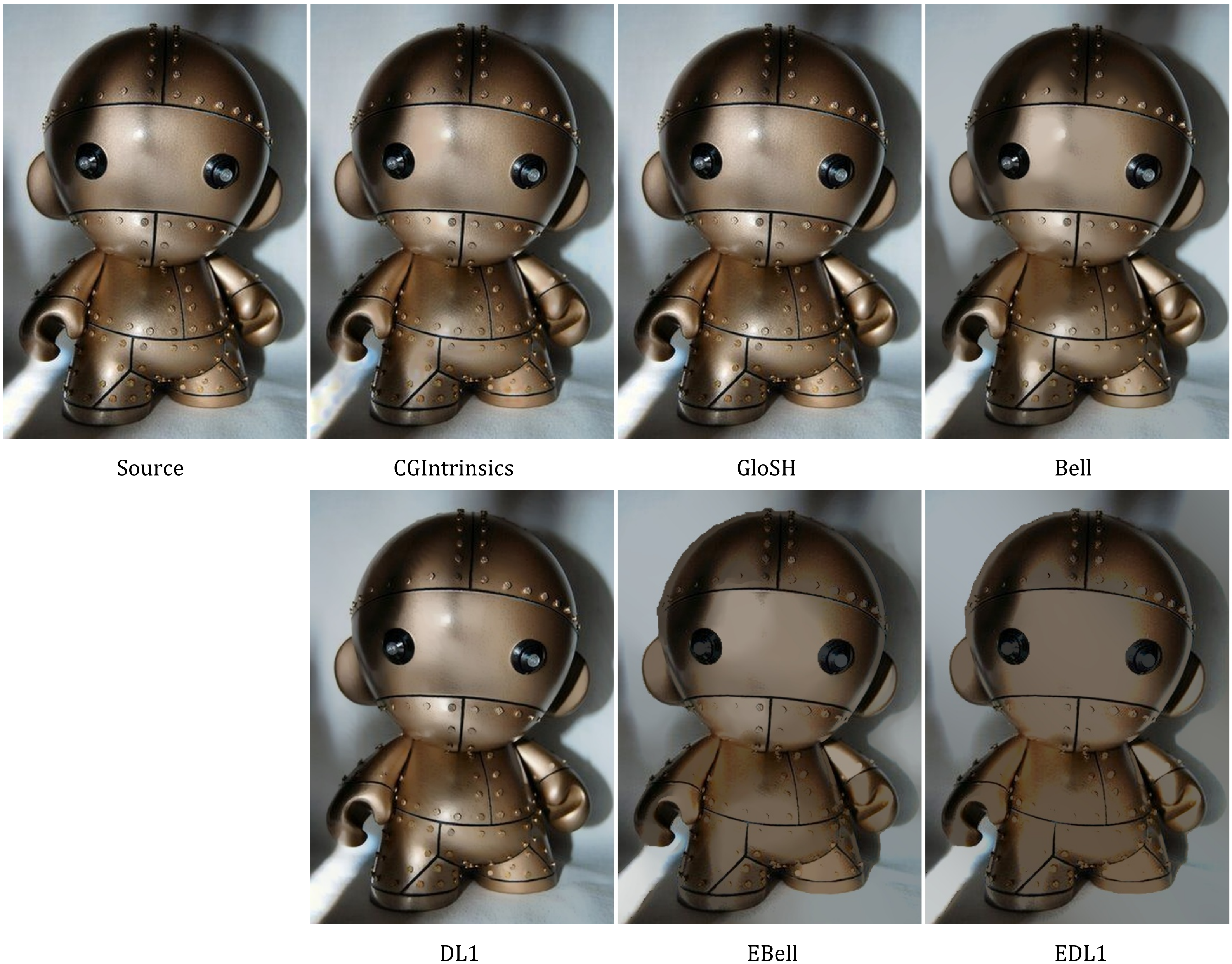

Figure 46: **Specular reflection removal.** Visual comparison of specular reflection removal using different intrinsic decomposition methods.

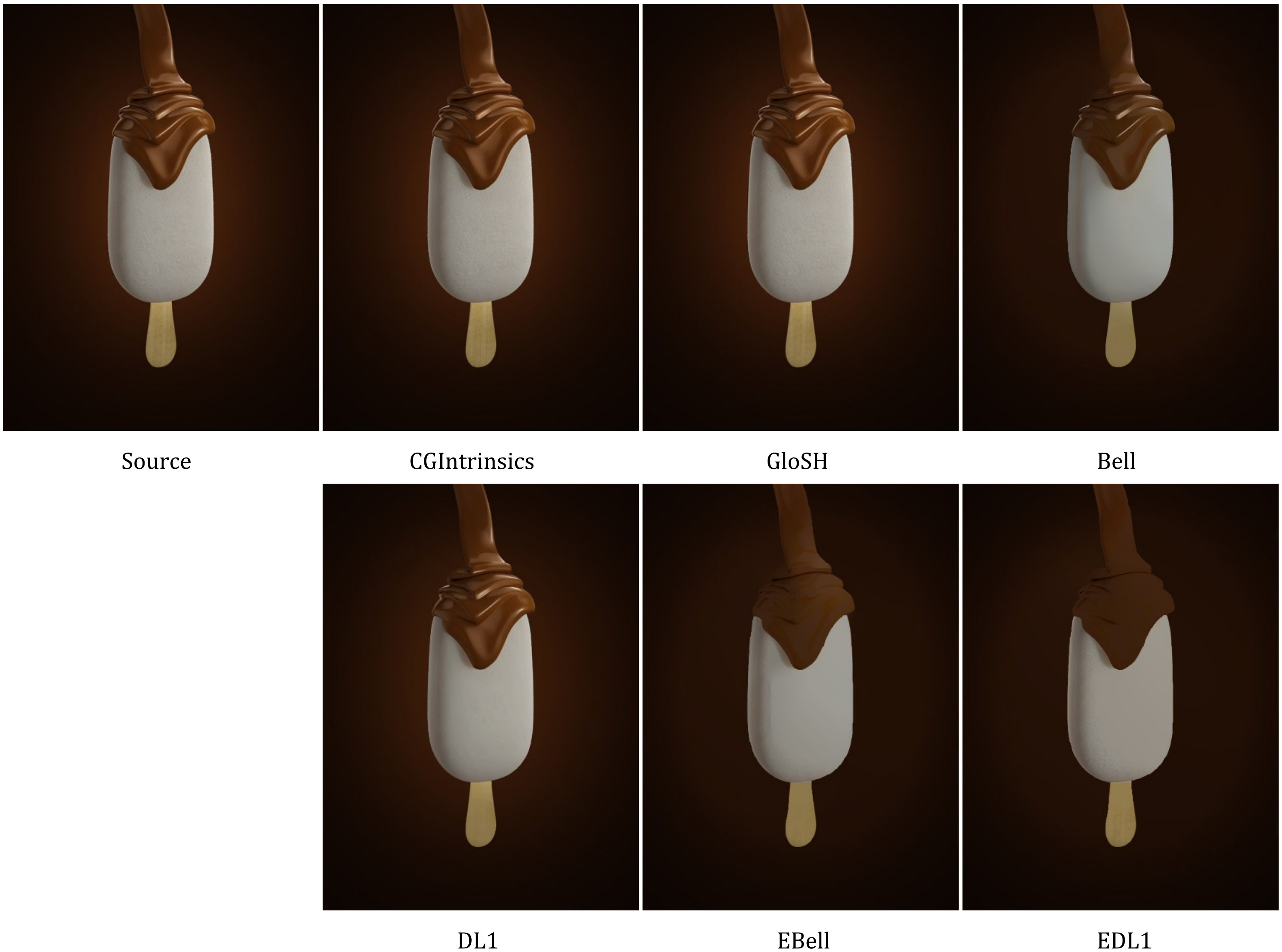

Figure 47: **Specular reflection removal.** Visual comparison of specular reflection removal using different intrinsic decomposition methods.

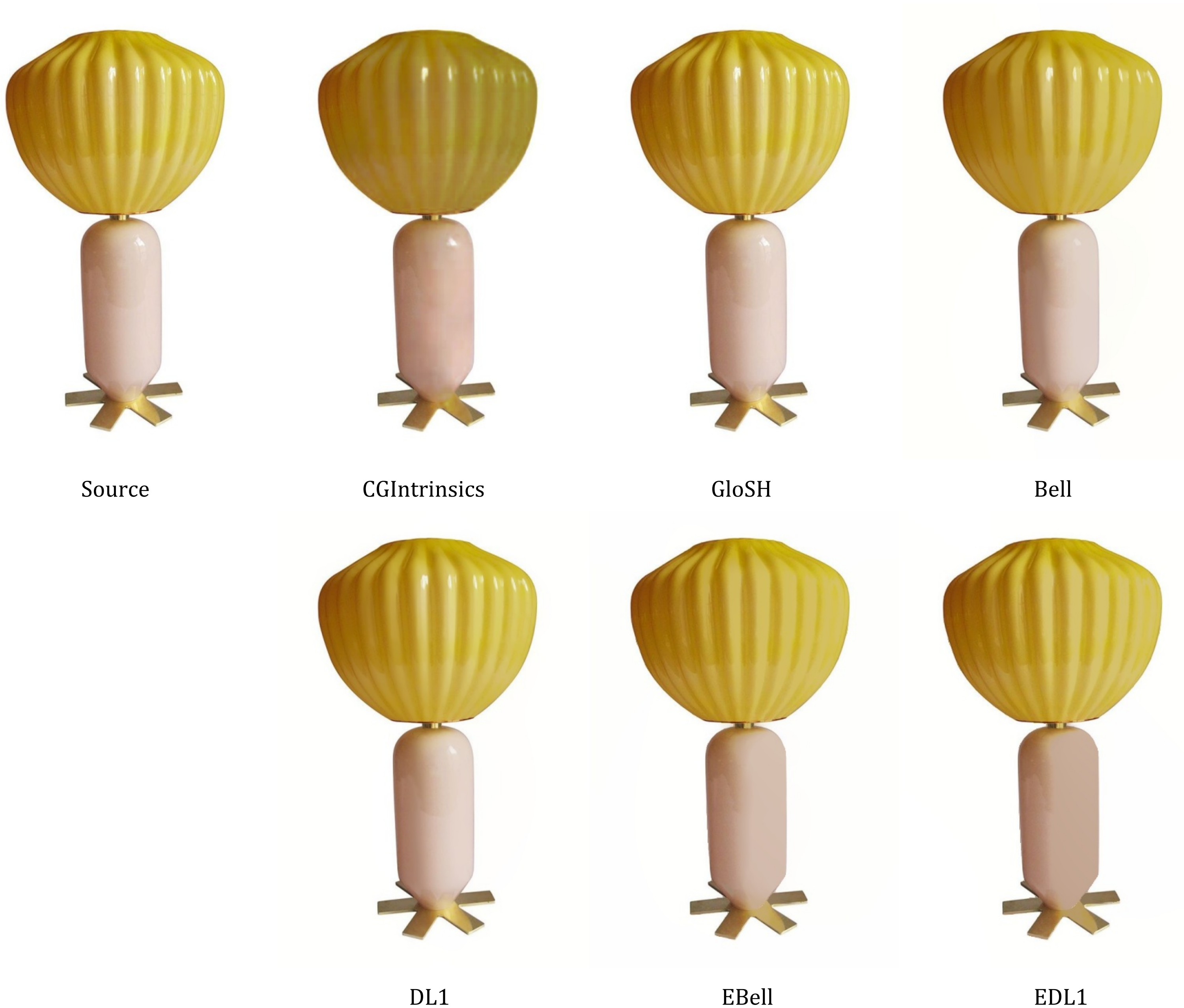

Figure 48: **Specular reflection removal.** Visual comparison of specular reflection removal using different intrinsic decomposition methods.

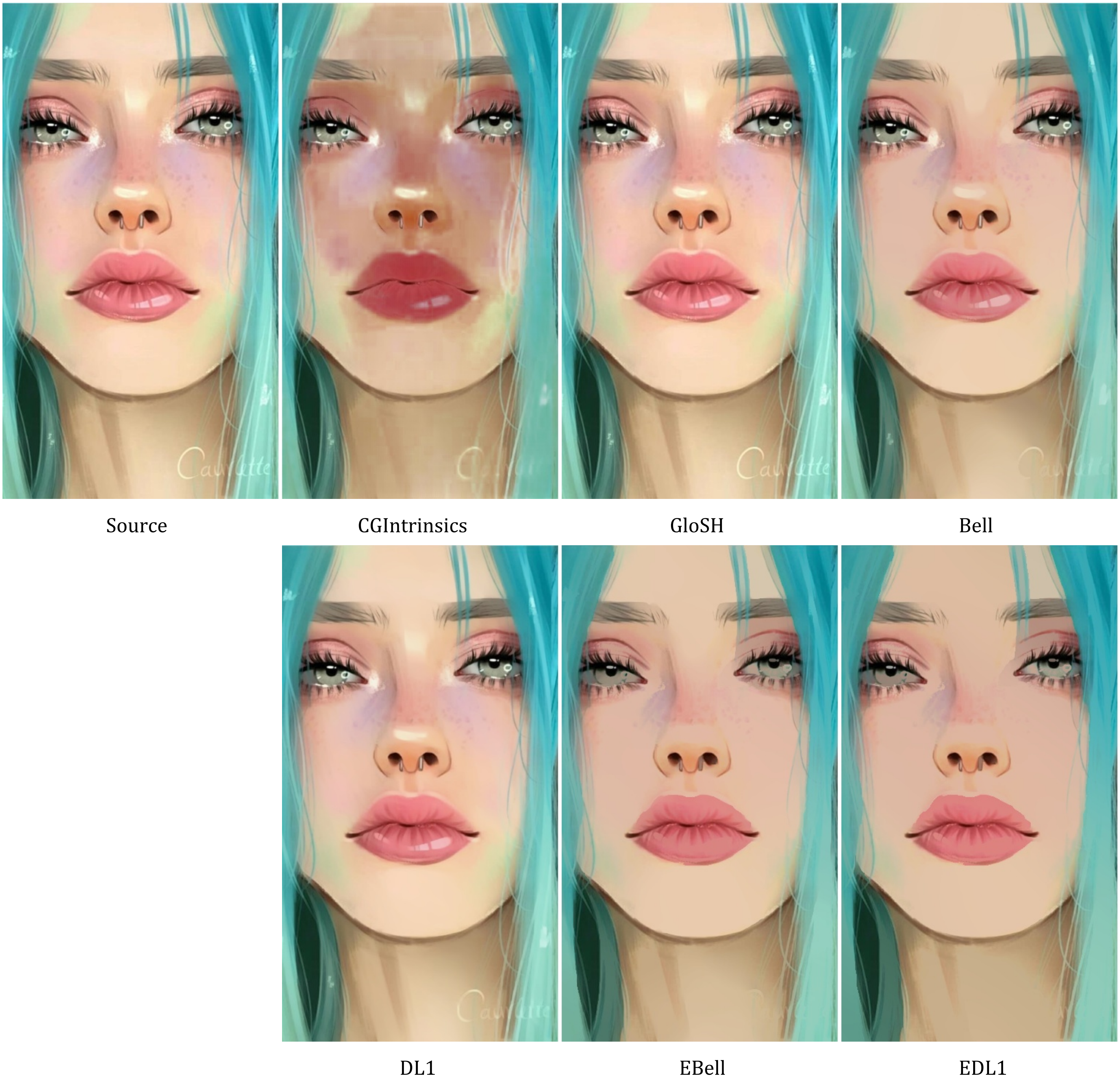

Figure 49: **Specular reflection removal.** Visual comparison of specular reflection removal using different intrinsic decomposition methods.

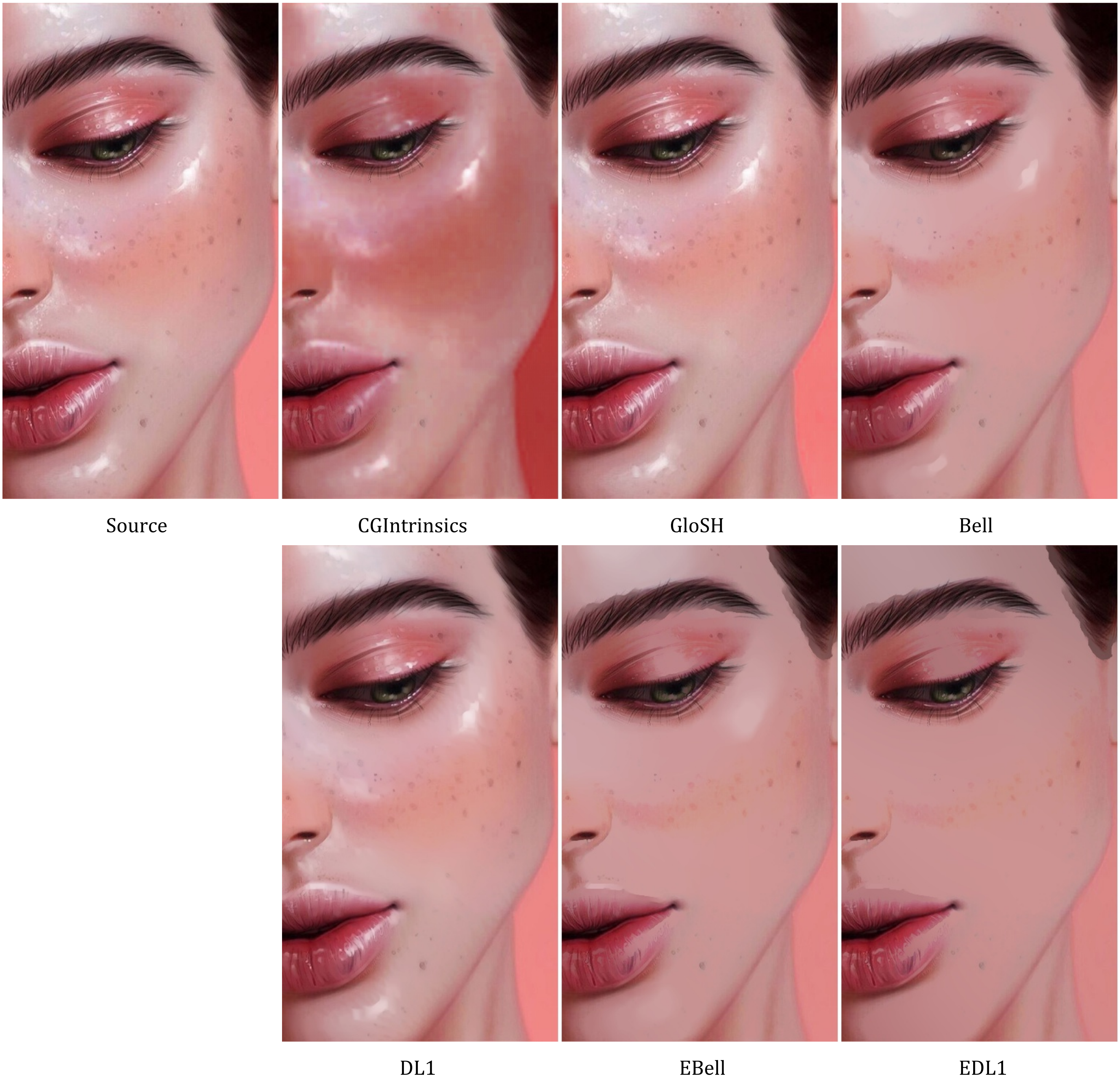

Figure 50: **Specular reflection removal.** Visual comparison of specular reflection removal using different intrinsic decomposition methods.

Figure 51: **Specular reflection removal.** Visual comparison of specular reflection removal using different intrinsic decomposition methods.

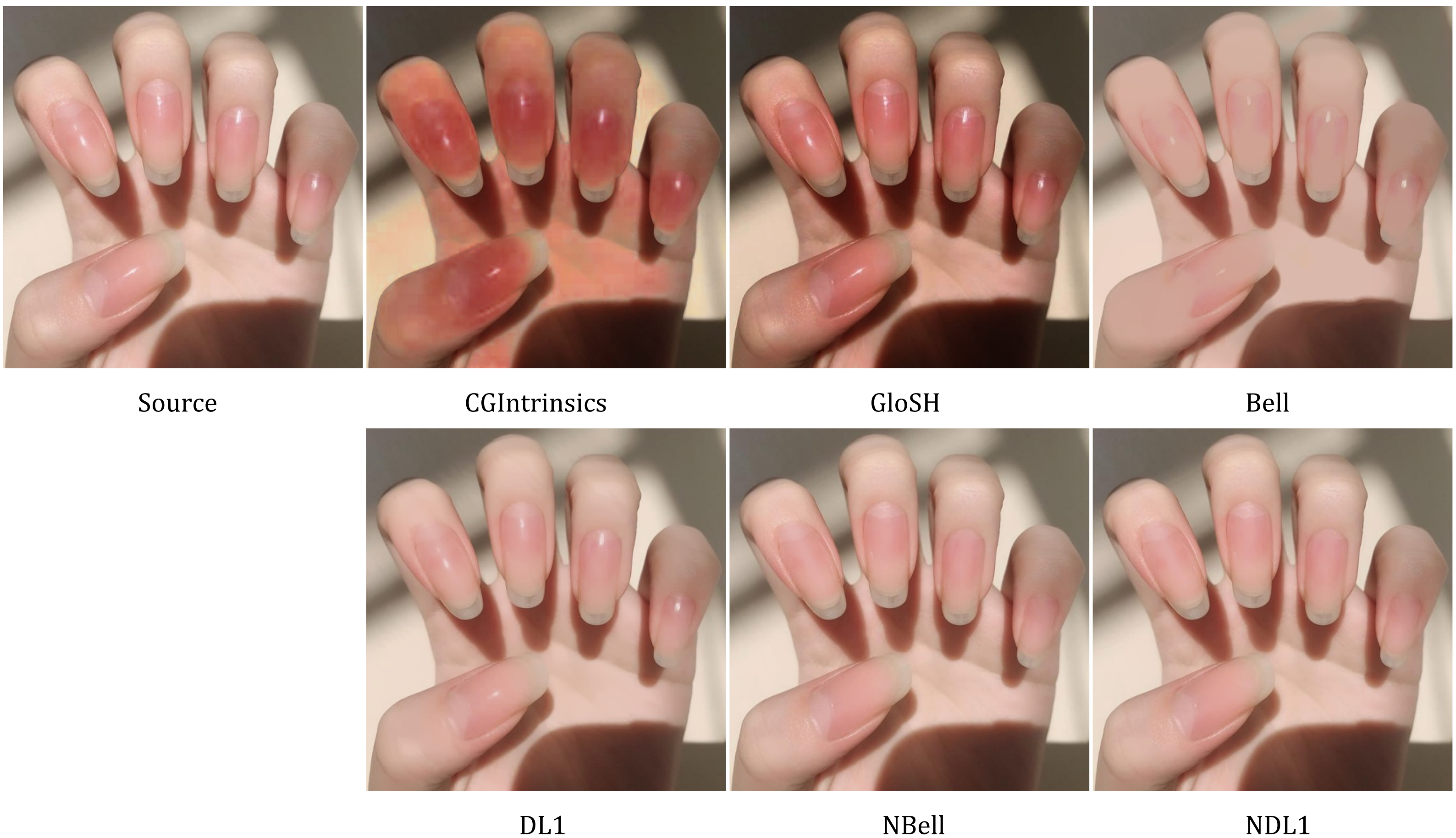

Figure 52: **Specular reflection removal.** Visual comparison of specular reflection removal using different intrinsic decomposition methods.

Figure 53: **Shadow enhancement.** Visual comparison of shadow enhancement using different intrinsic decomposition methods.

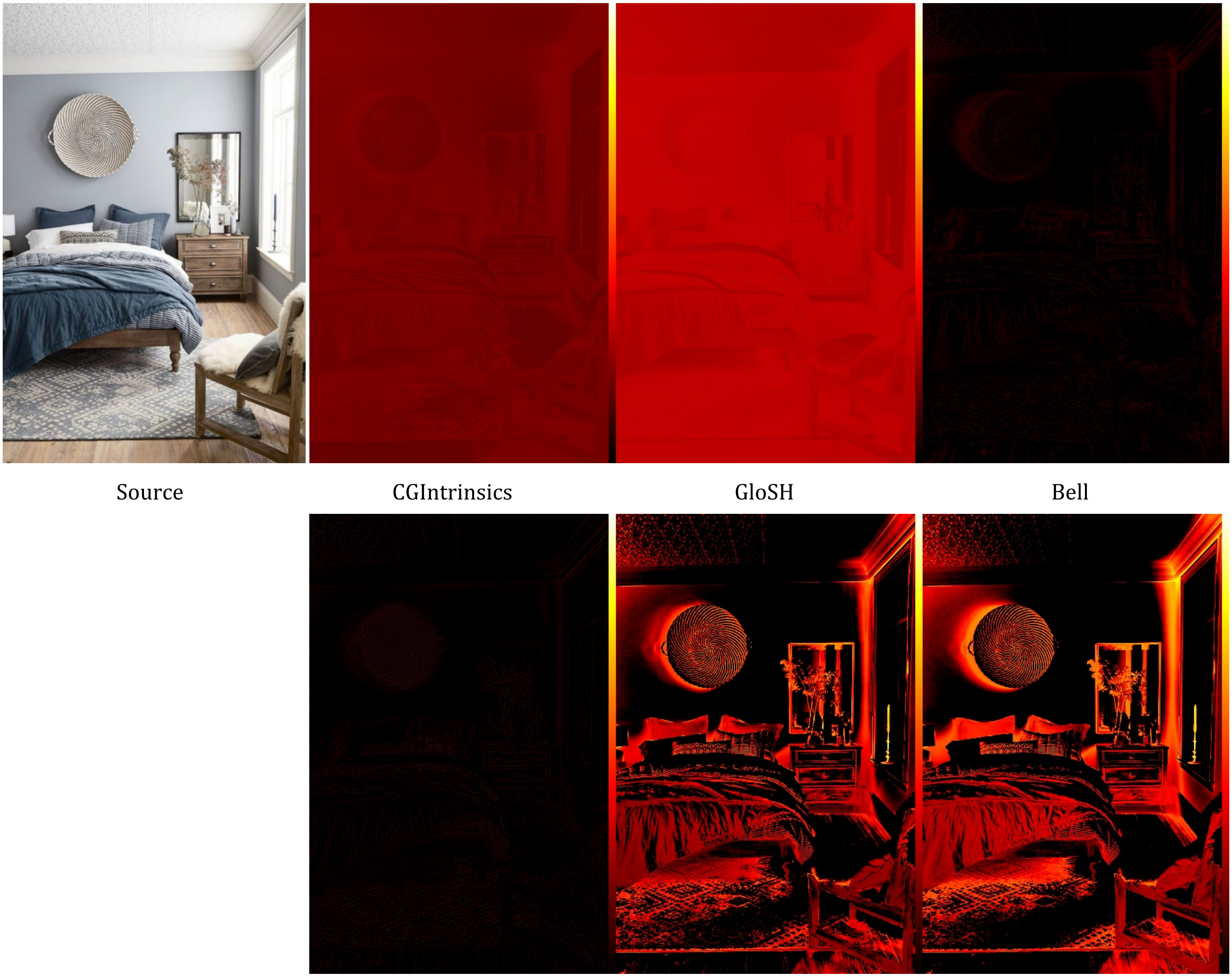

Figure 54: **Shadow enhancement.** Log-distance map between shadow enhancement map and naive gamma correction map. This distance is computed as the log-space distance $D_{\text{distance}} = \log Y_{\text{enhanced}} - \log X^k$ with Y_{enhanced} being the shadow-enhanced map and X^k is the k-gamma-corrected image ($k = 2.0$). This distance reflects to what extant the enhanced shadows are meaningful and different from the naive gamma corrections. Working in log space prevents zero division and yields results faithful to the irradiance models in previous intrinsic literature. We visualize the negative part (or called the shadow part) of this measurement.

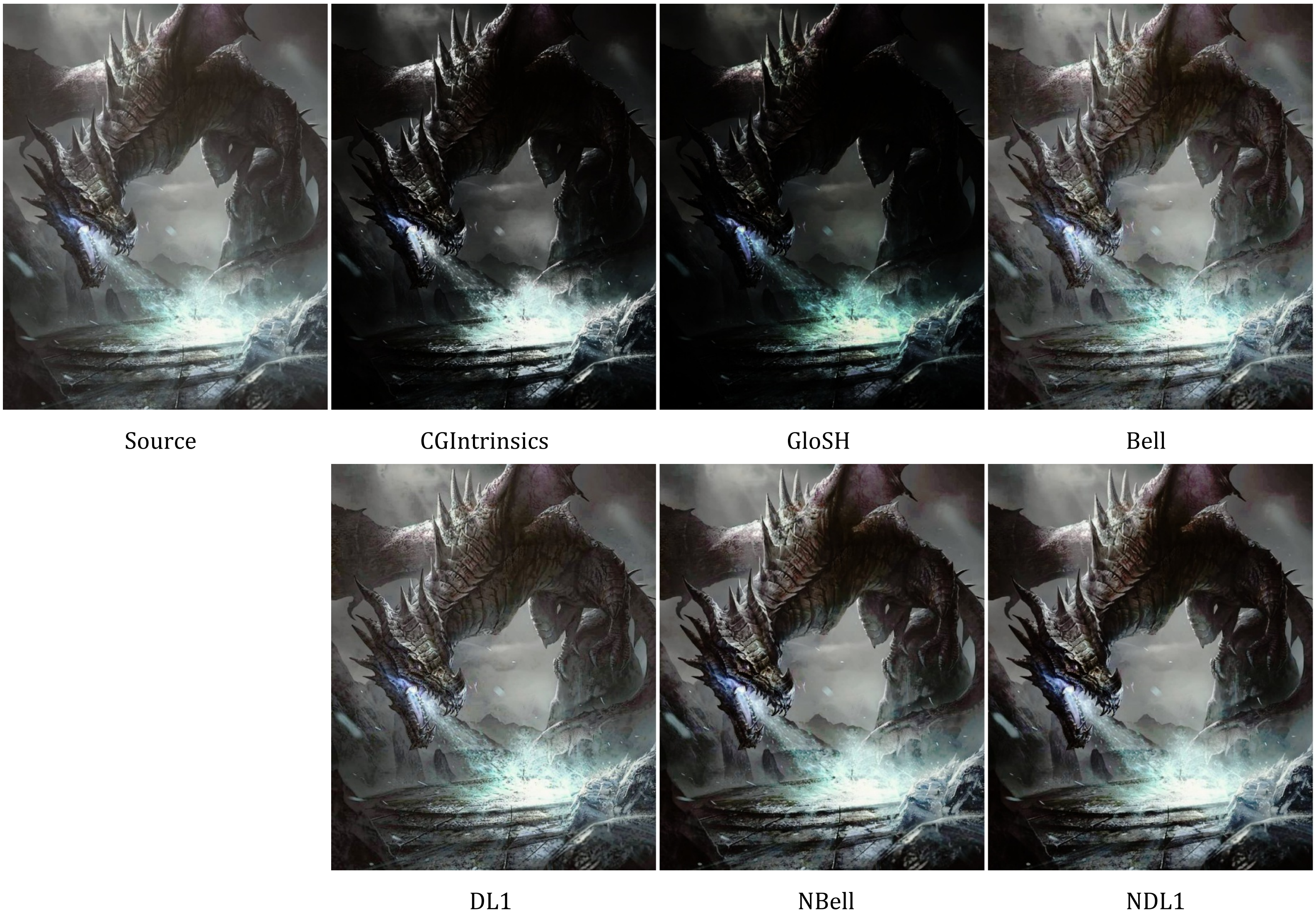

Figure 55: **Shadow enhancement.** Visual comparison of shadow enhancement using different intrinsic decomposition methods.

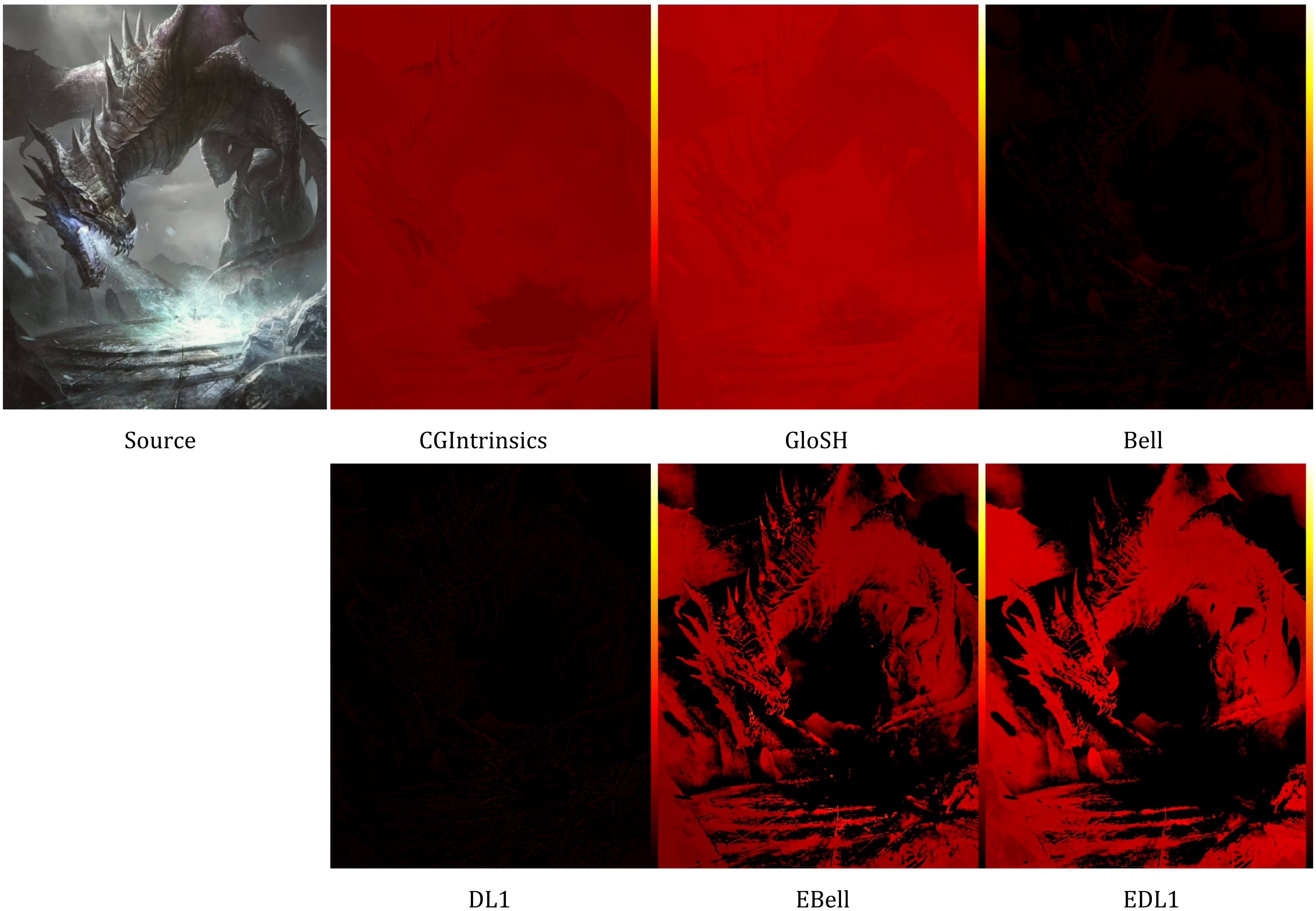

Figure 56: **Shadow enhancement.** Log-distance map between shadow enhancement map and naive gamma correction map. This distance is computed as the log-space distance $D_{\text{distance}} = \log Y_{\text{enhanced}} - \log X^k$ with Y_{enhanced} being the shadow-enhanced map and X^k is the k-gamma-corrected image ($k = 2.0$). This distance reflects to what extant the enhanced shadows are meaningful and different from the naive gamma corrections. Working in log space prevents zero division and yields results faithful to the irradiance models in previous intrinsic literature. We visualize the negative part (or called the shadow part) of this measurement.

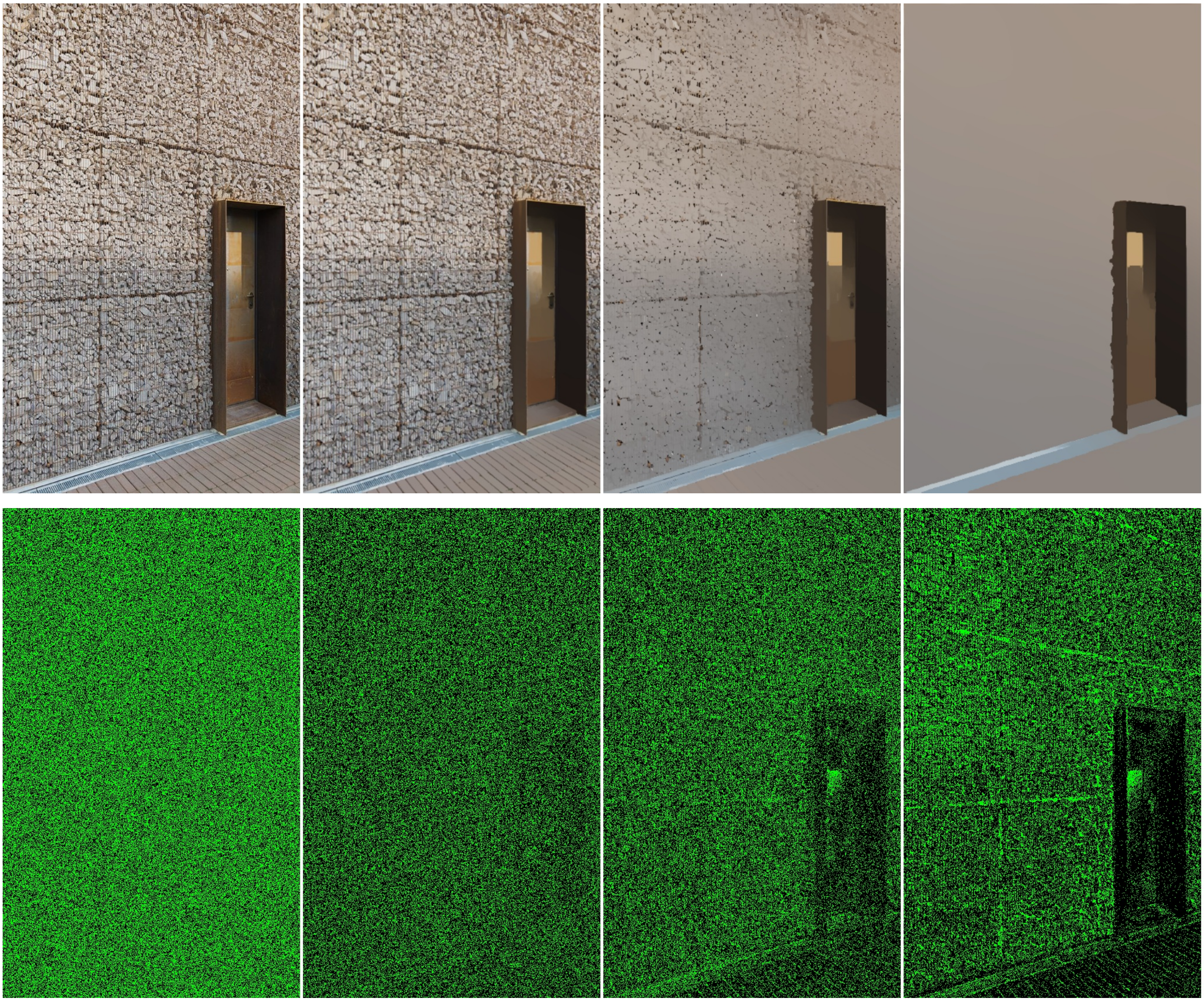

Figure 57: **Erasing position visualization.** Extended $\mathcal{E}$ visualization as in the main paper. In the first row are the smoothed results using EDL1 at iteration 0, 1, 3, and 7. In the second row is the visualized erasing sets $\mathcal{E}$, marked in green. Please refer to main article for expositions.

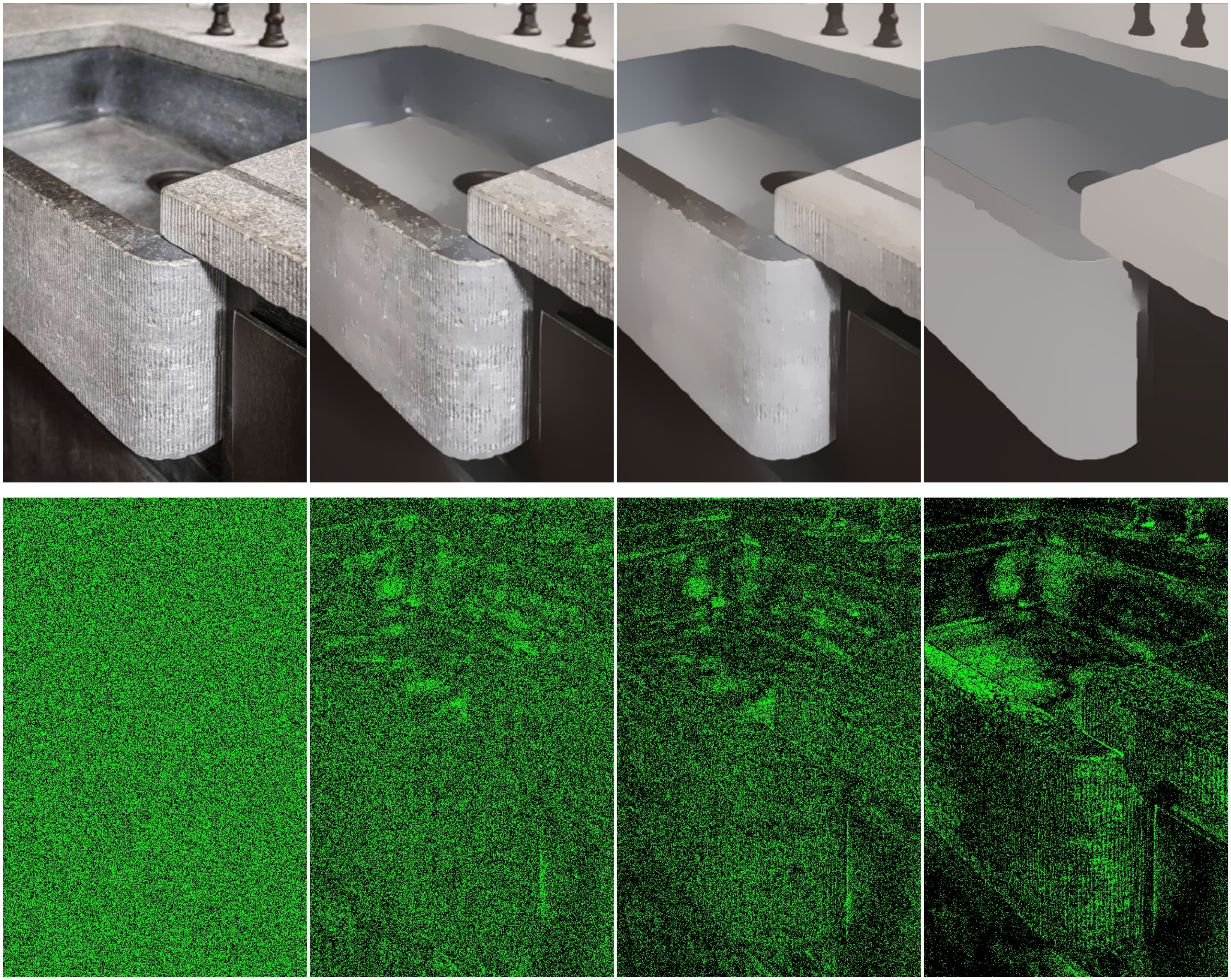

Figure 58: **Erasing position visualization.** Extended $\mathcal{E}$ visualization as in the main paper. In the first row are the smoothed results using EDL1 at iteration 0, 1, 3, and 7. In the second row is the visualized erasing sets $\mathcal{E}$, marked in green. Please refer to main article for expositions.

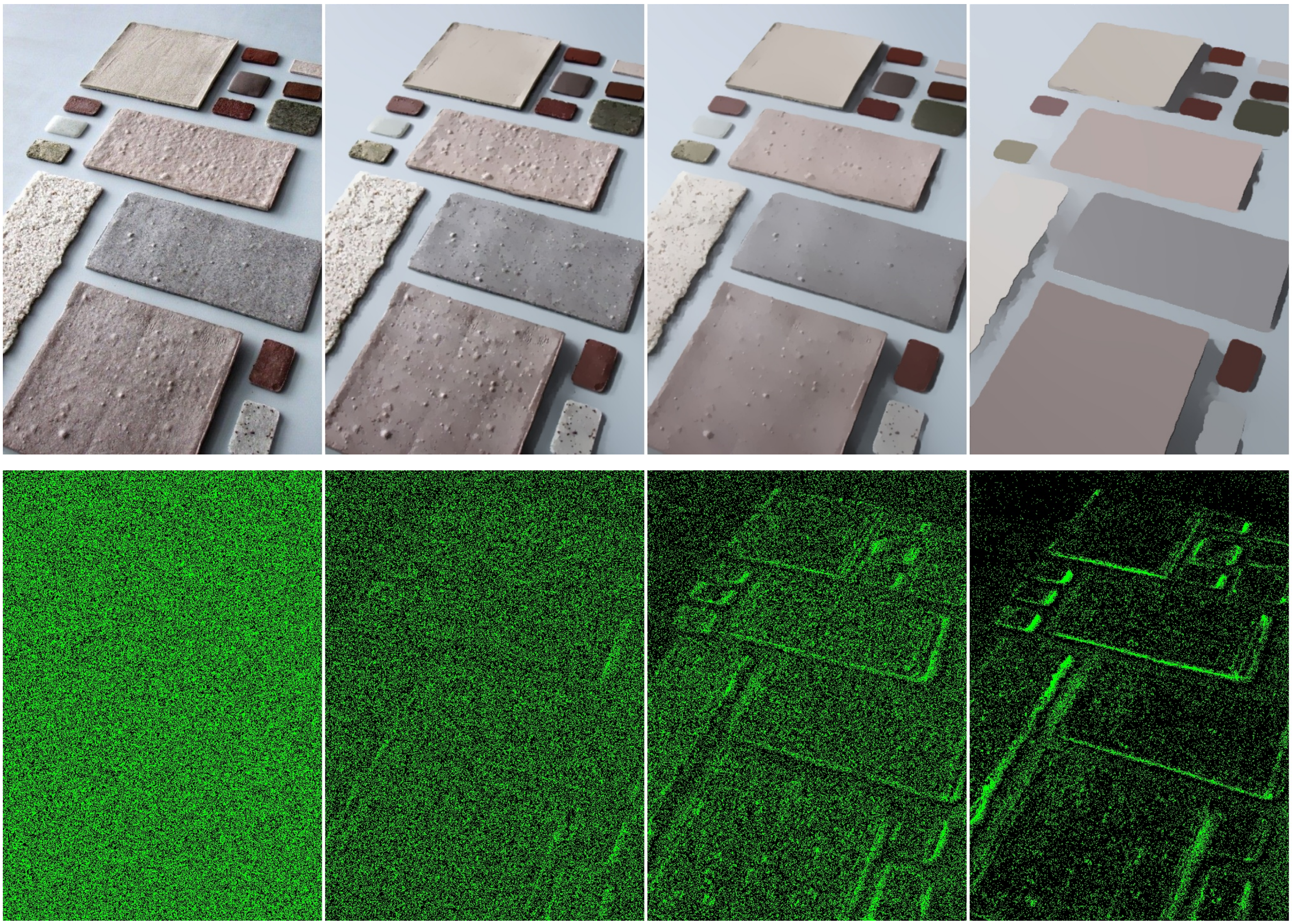

Figure 59: **Erasing position visualization.** Extended $\mathcal{E}$ visualization as in the main paper. In the first row are the smoothed results using EDL1 at iteration 0, 1, 3, and 7. In the second row is the visualized erasing sets $\mathcal{E}$, marked in green. Please refer to main article for expositions.

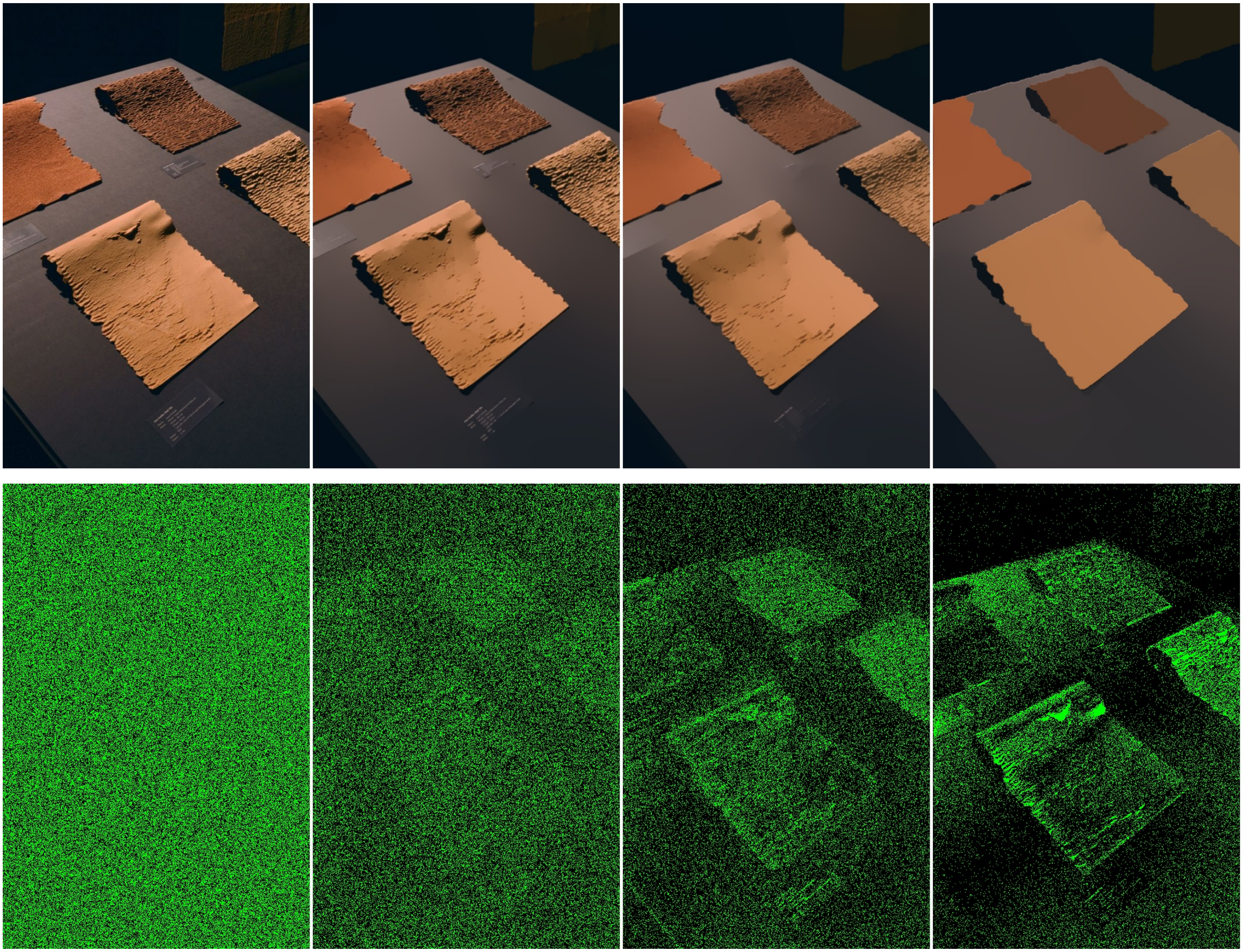

Figure 60: **Erasing position visualization.** Extended $\mathcal{E}$ visualization as in the main paper. In the first row are the smoothed results using EDL1 at iteration 0, 1, 3, and 7. In the second row is the visualized erasing sets $\mathcal{E}$, marked in green. Please refer to main article for expositions.

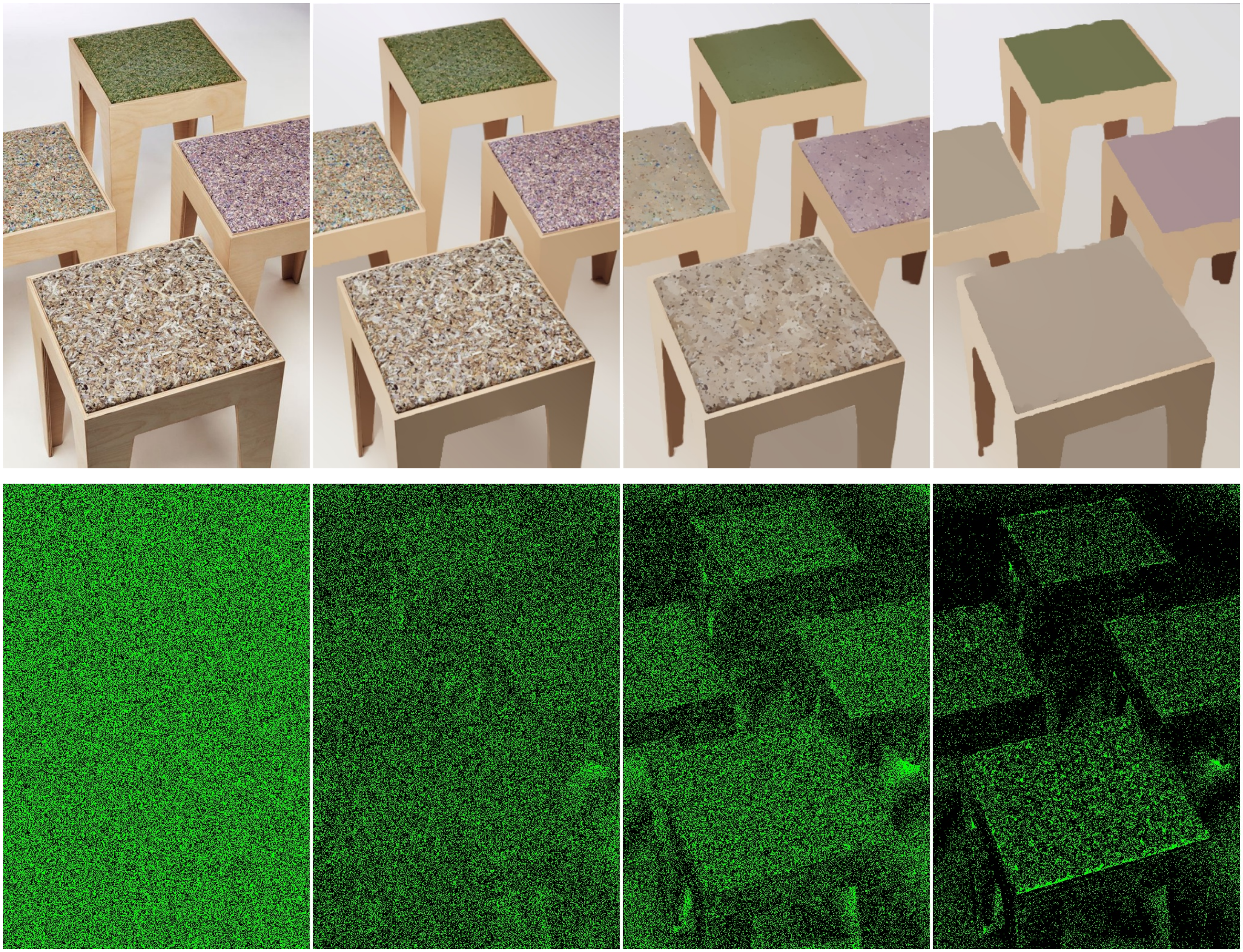

Figure 61: **Erasing position visualization.** Extended $\mathcal{E}$ visualization as in the main paper. In the first row are the smoothed results using EDL1 at iteration 0, 1, 3, and 7. In the second row is the visualized erasing sets $\mathcal{E}$, marked in green. Please refer to main article for expositions.

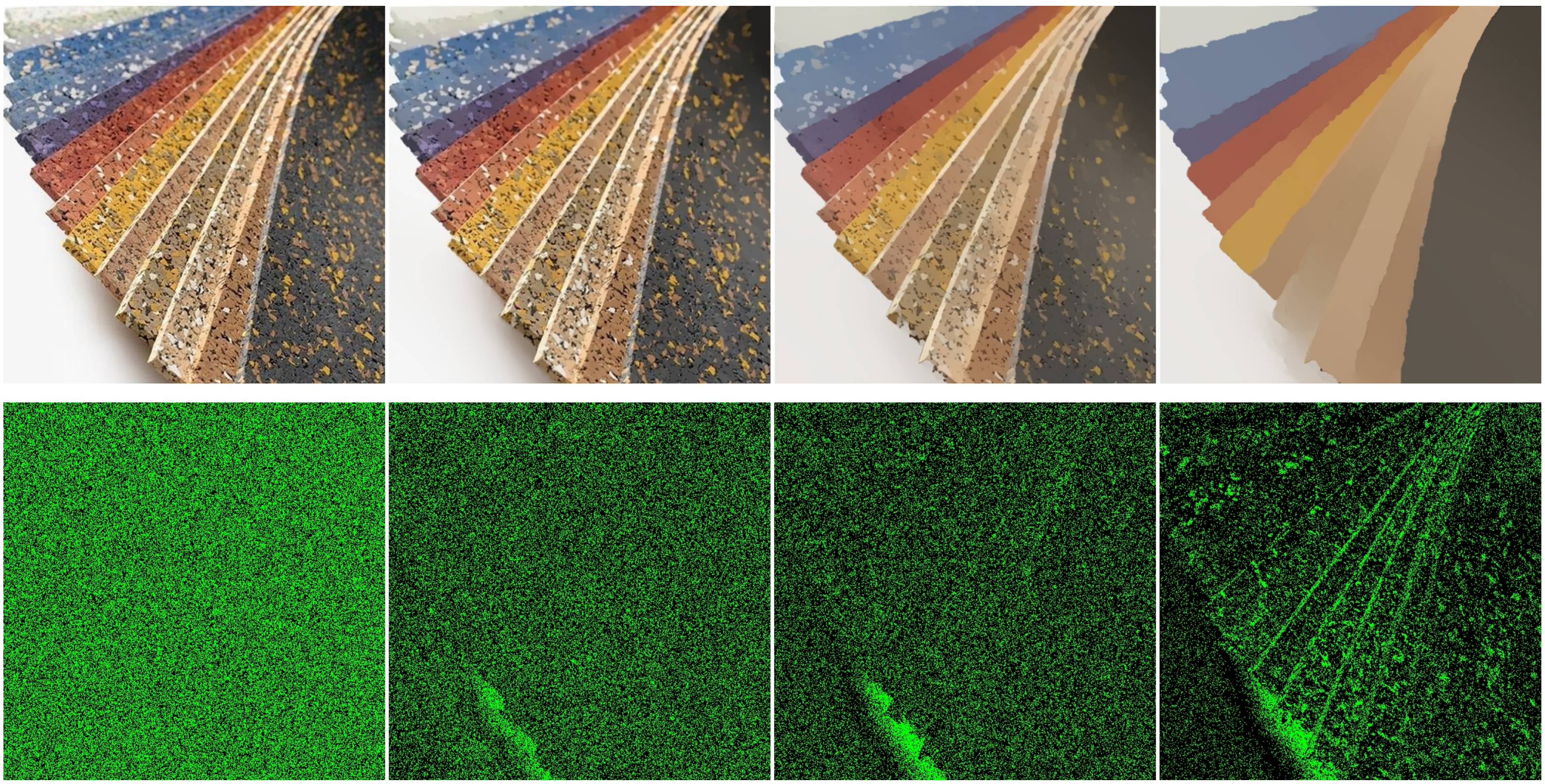

Figure 62: **Erasing position visualization.** Extended $\mathcal{E}$ visualization as in the main paper. In the first row are the smoothed results using EDL1 at iteration 0, 1, 3, and 7. In the second row is the visualized erasing sets $\mathcal{E}$, marked in green. Please refer to main article for expositions.

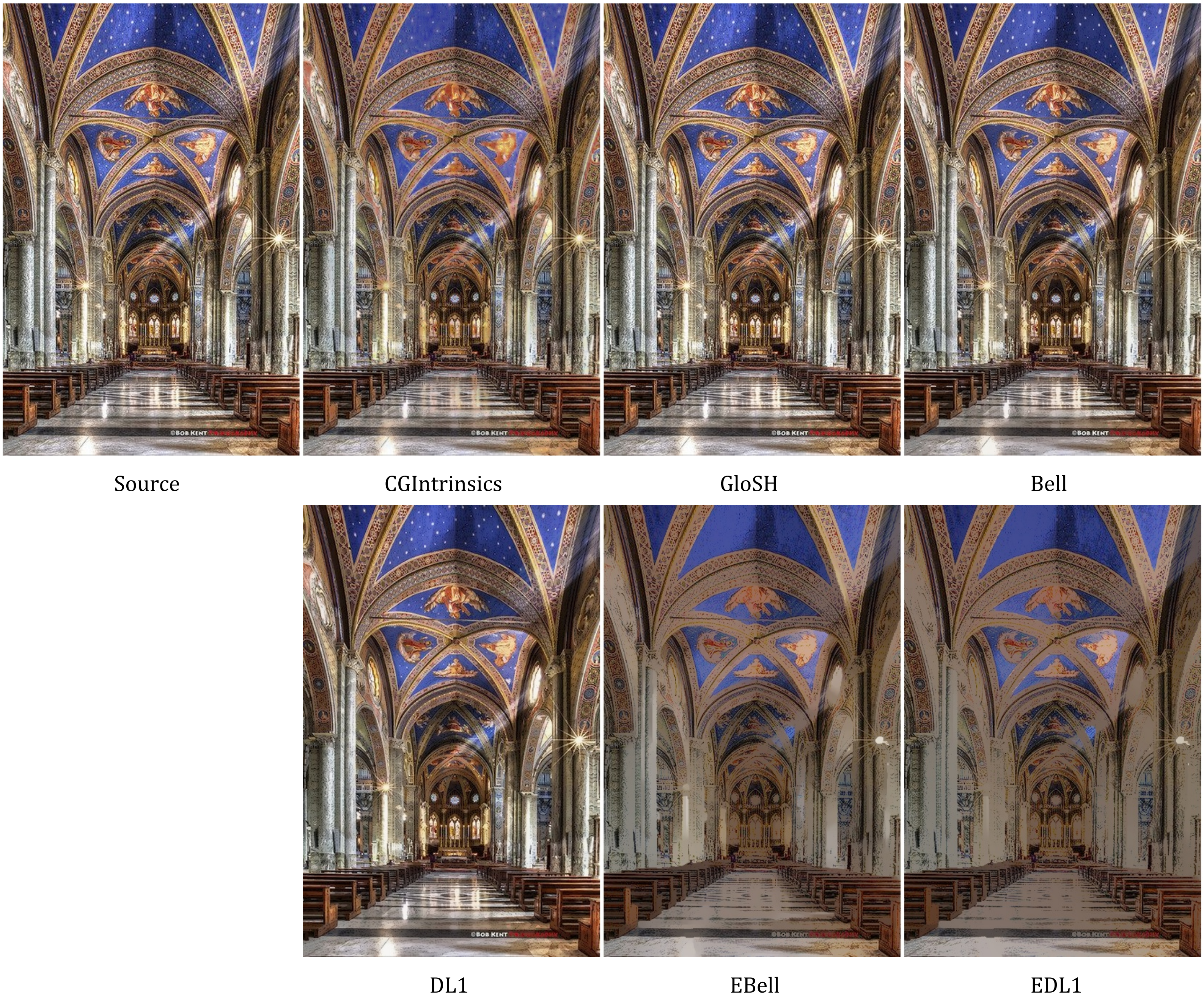

Figure 63: **Limitation.** The specular reflection removal causes unrealistic result in this example.