Decomposing Images into Layers with Advanced Color Blending

Yuki Koyama & Masataka Goto

Quick Summary

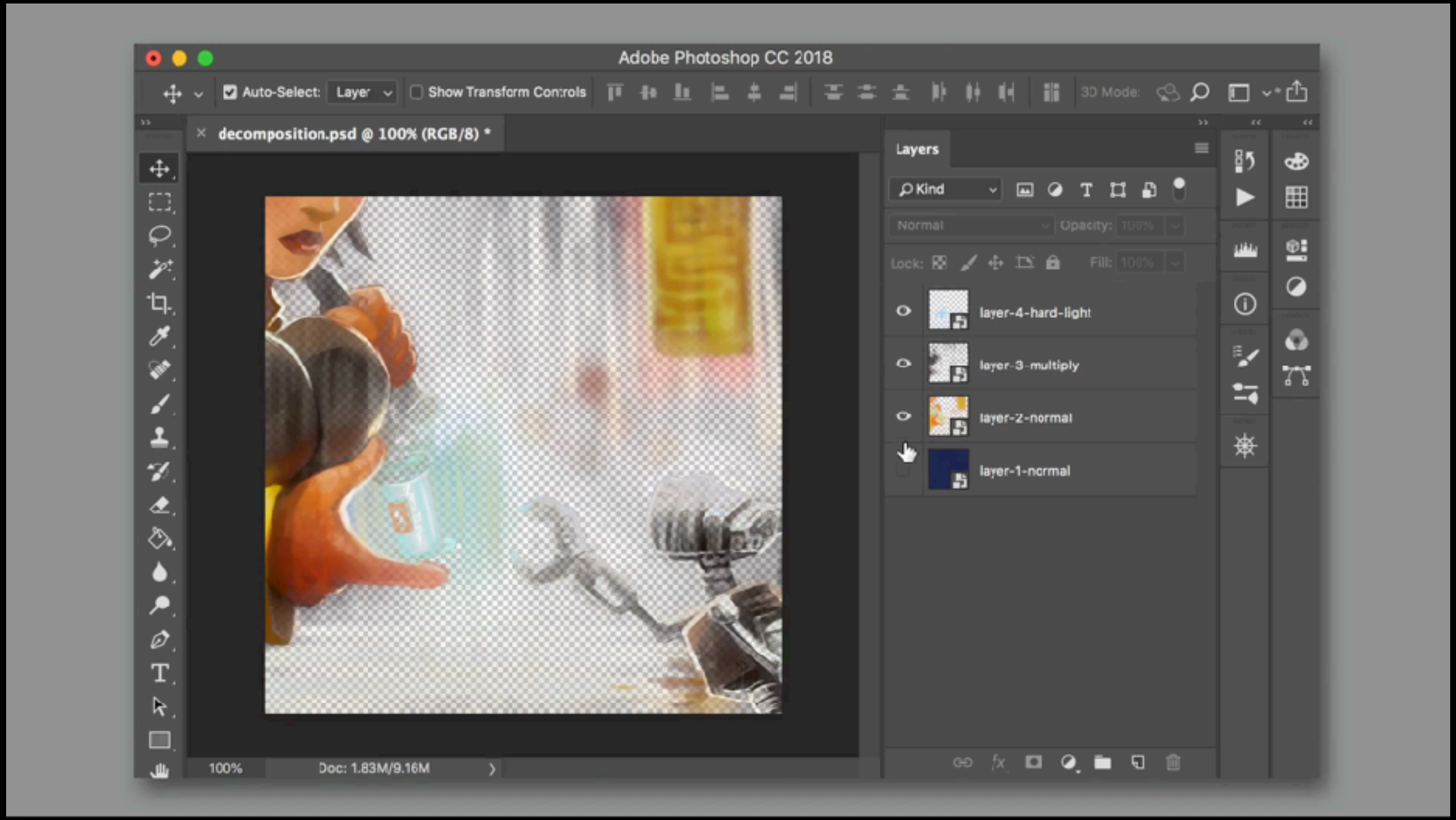
Our method decomposes a target image into semi-transparent layers ...

Input image courtesy of David Revoy | www.davidrevoy.com

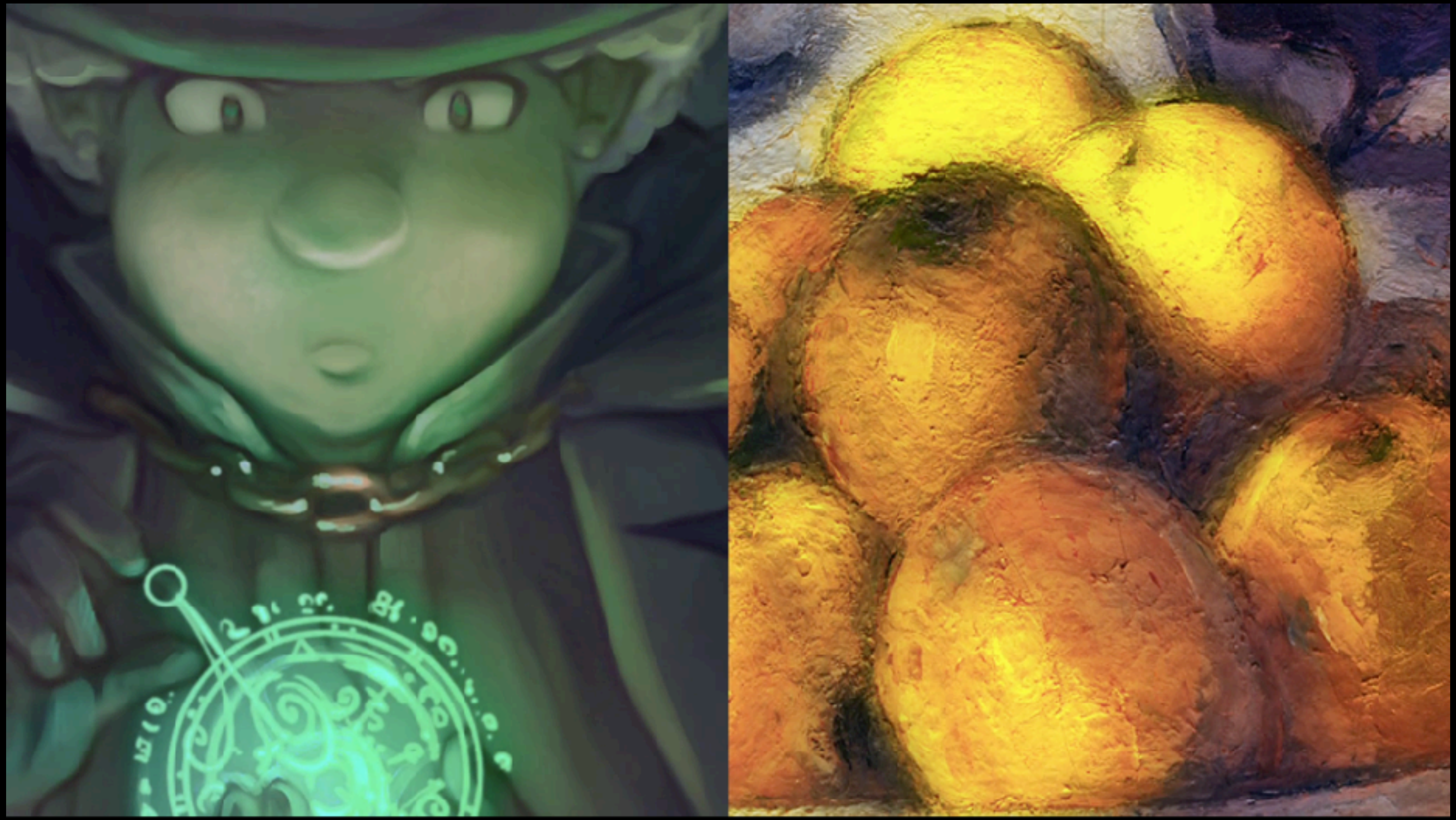
... that are associated with advanced color-blend modes such as **hard-light** and **multiply**.

Input image courtesy of David Revoy | www.davidrevoy.com

Usage Scenario



Import the resulting layers
to Photoshop etc.

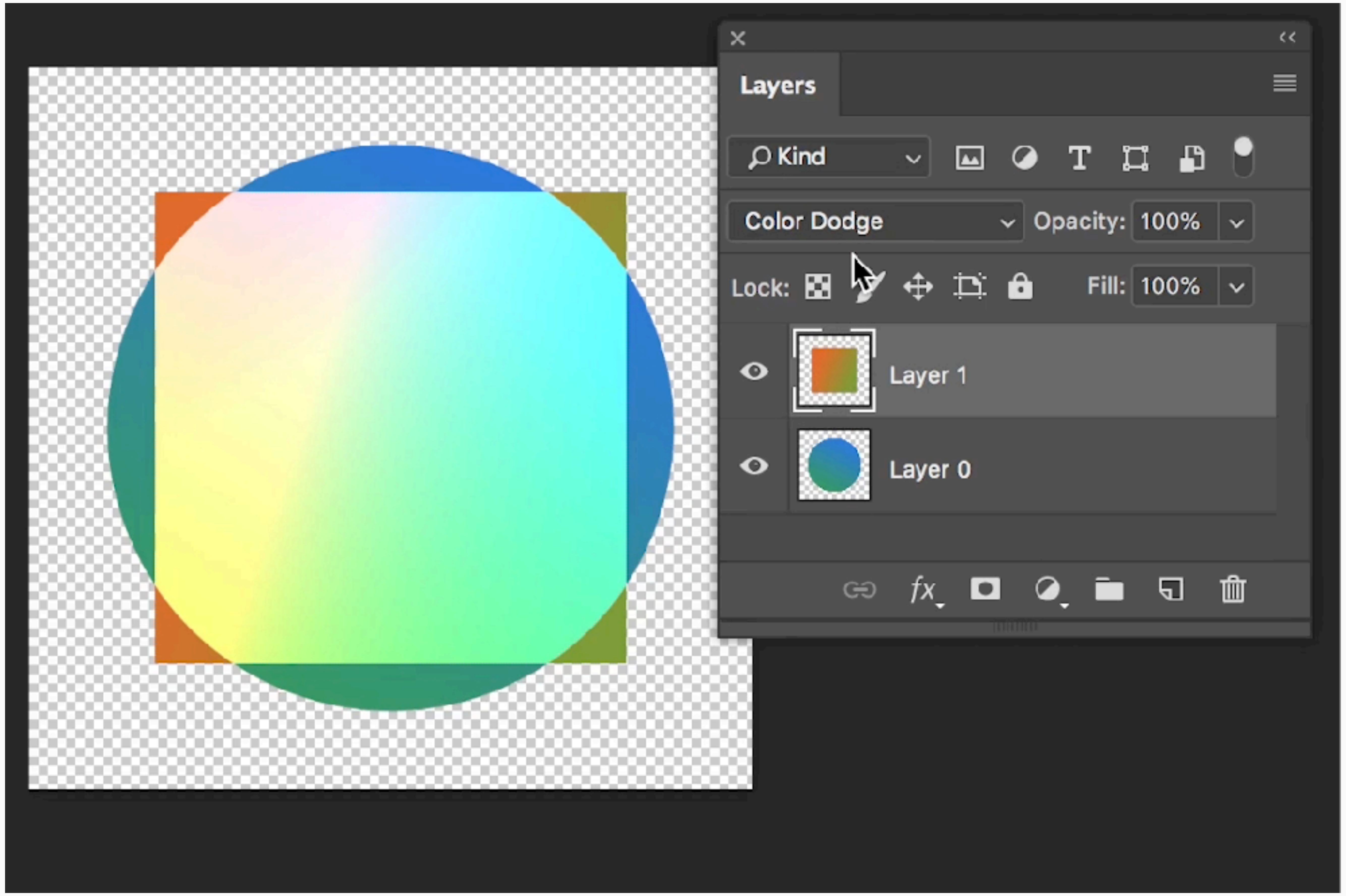


Perform non-trivial edits
(e.g., lighting-aware hue change)

Input images except for the rightmost one courtesy of David Revoy

Background
What is Advanced Color Blending?

Color Blending & Blend Mode

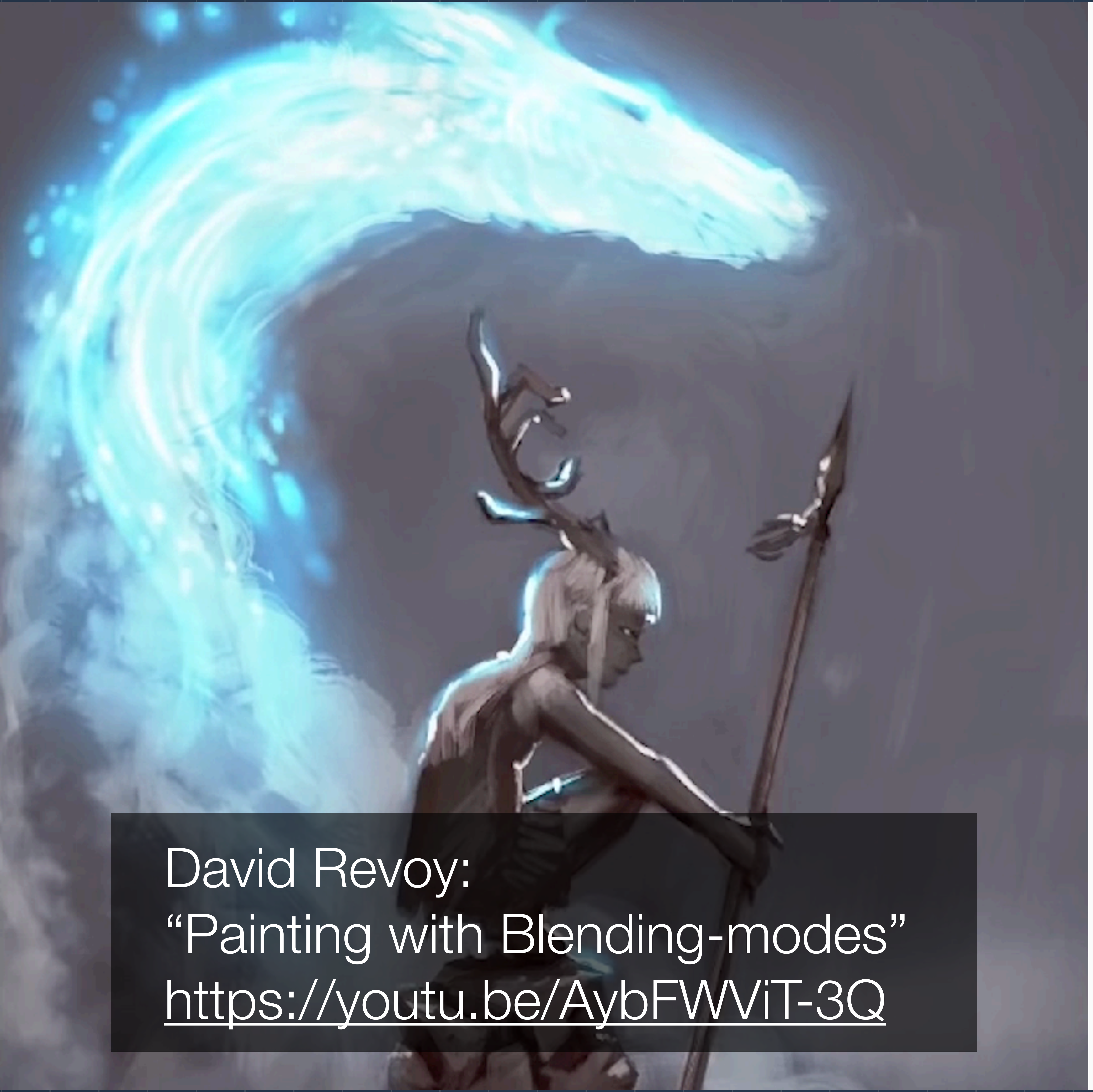


A ***color-blend mode*** defines the mapping from the colors of the layers to the color that will be rendered

Examples: **multiply**, **overlay**, **color-dodge**, **normal**, etc.

Available in …

Digital Drawing & Effects with Color Blending



David Revoy:
“Painting with Blending-modes”
https://youtu.be/AybFWViT-3Q

Advanced (non-linear) color blending is used for creating interesting color effects

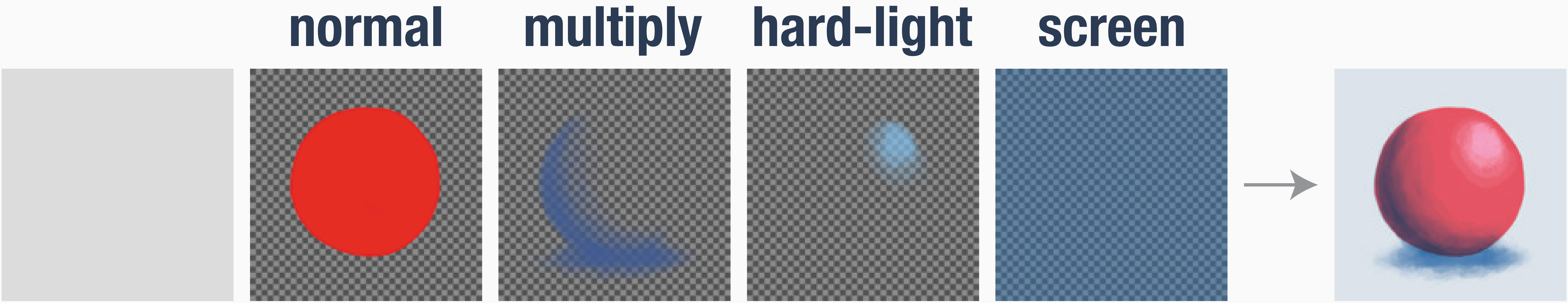
Example:
digital drawing using the **color-dodge** mode

Digital Drawing & Effects with Color Blending

It is important to combine various blend modes

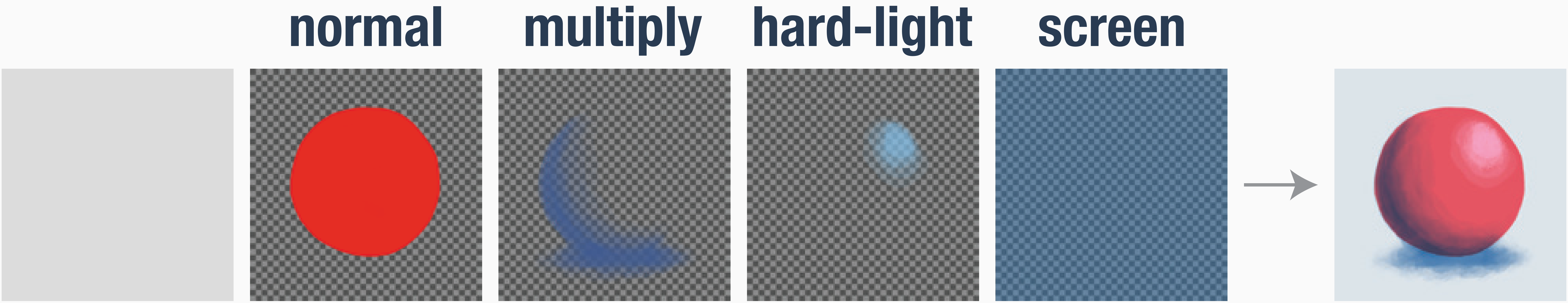
Digital Drawing & Effects with Color Blending

It is important to combine various blend modes



Digital Drawing & Effects with Color Blending

It is important to combine various blend modes



Note: every digital artist has his/her own way to combine blend modes

Our Motivation

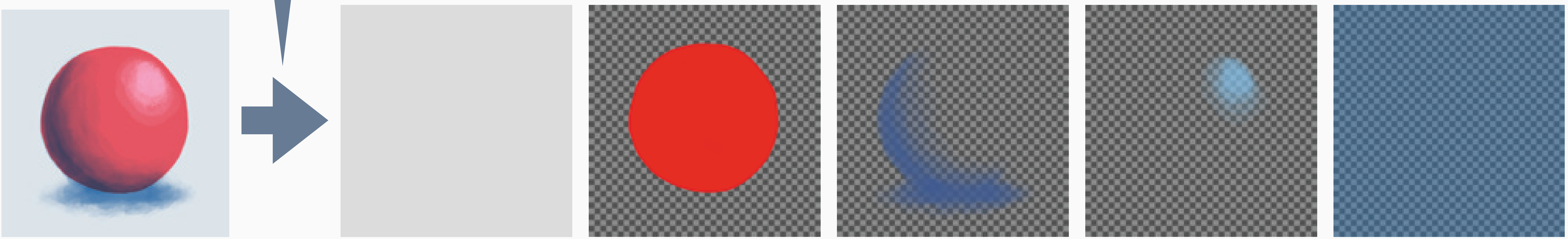
Motivation: Color Unblending

Arbitrary combination of blend modes

It should be useful if an existing image can be decomposed into layers with an arbitrary combination of advanced color-blend modes

Unblending

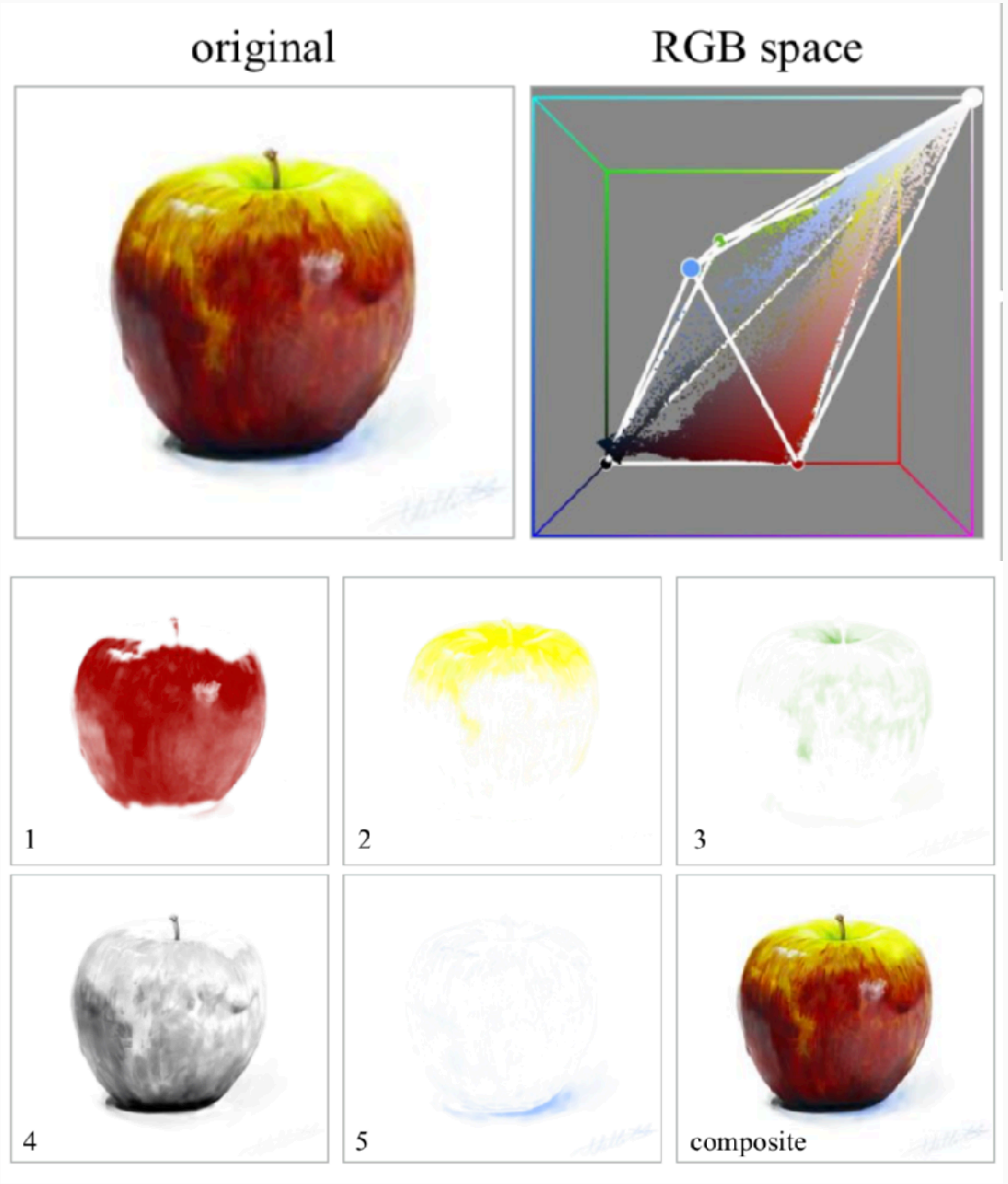
normal **multiply** **hard-light** **screen**



Previous Work

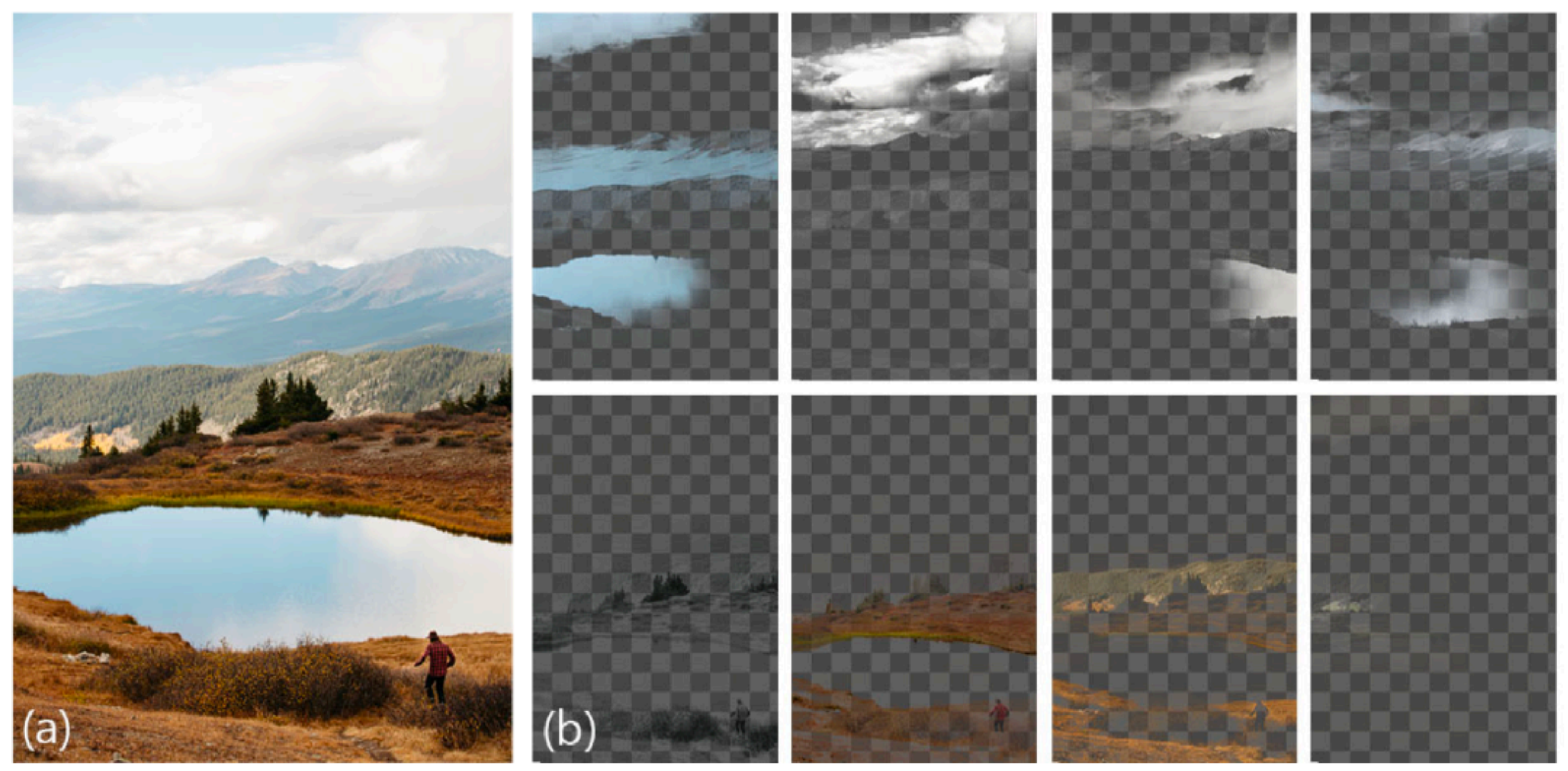
Layer Decomposition by Color Unblending

RGB-Space Geometry [Tan+, TOG 16]



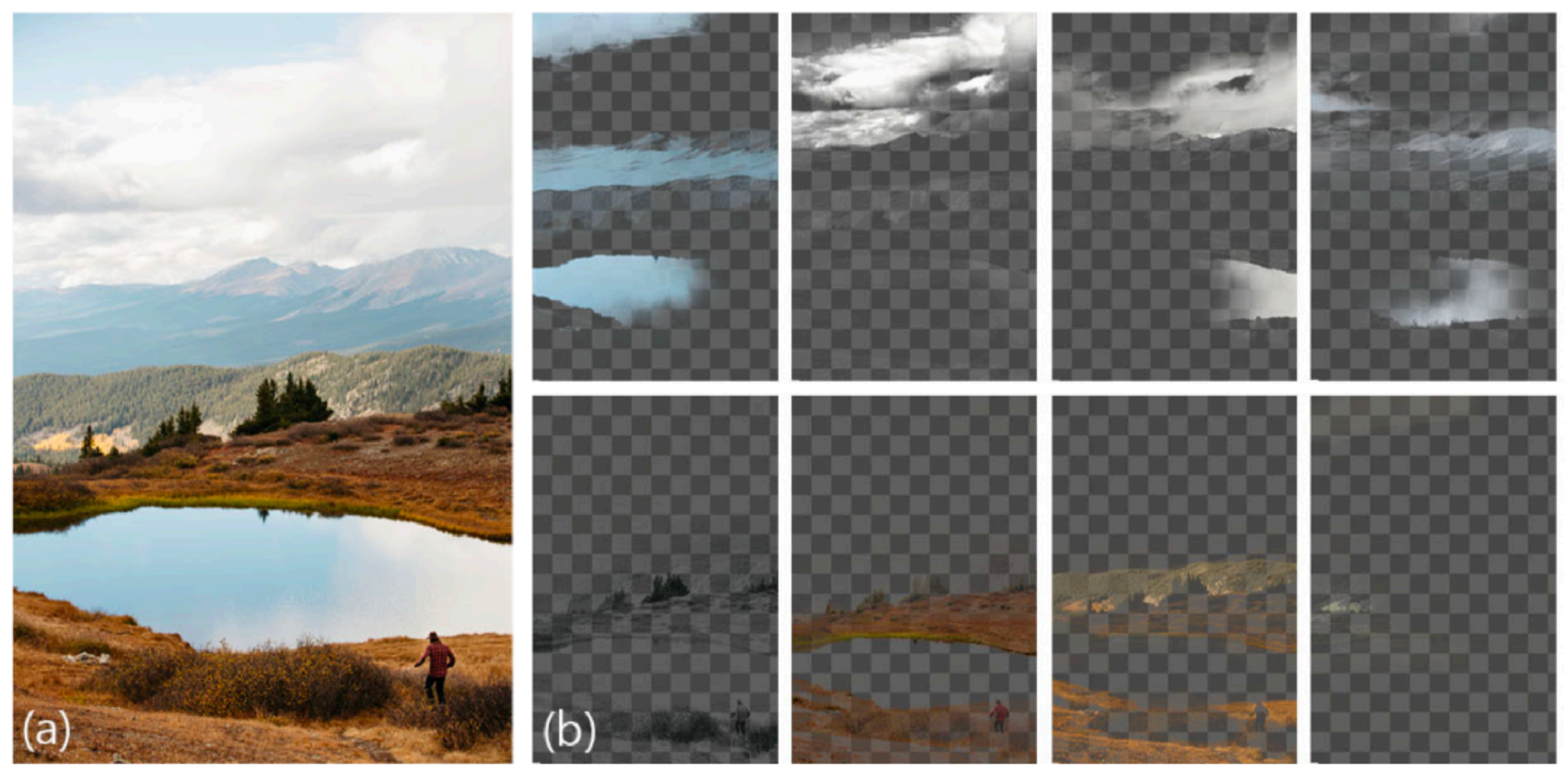
- Support **only a specific linear blend mode** (i.e., "normal")
- Heavily **rely on the linearity** of the blend mode
- Cannot be (easily) extended for non-linear color-blend modes

Per-Pixel Optimization [Aksoy+, TOG 16; TOG 17]



- Support **only a specific linear blend mode** (and a linear alpha addition)

Per-Pixel Optimization [Aksoy+, TOG 16; TOG 17]



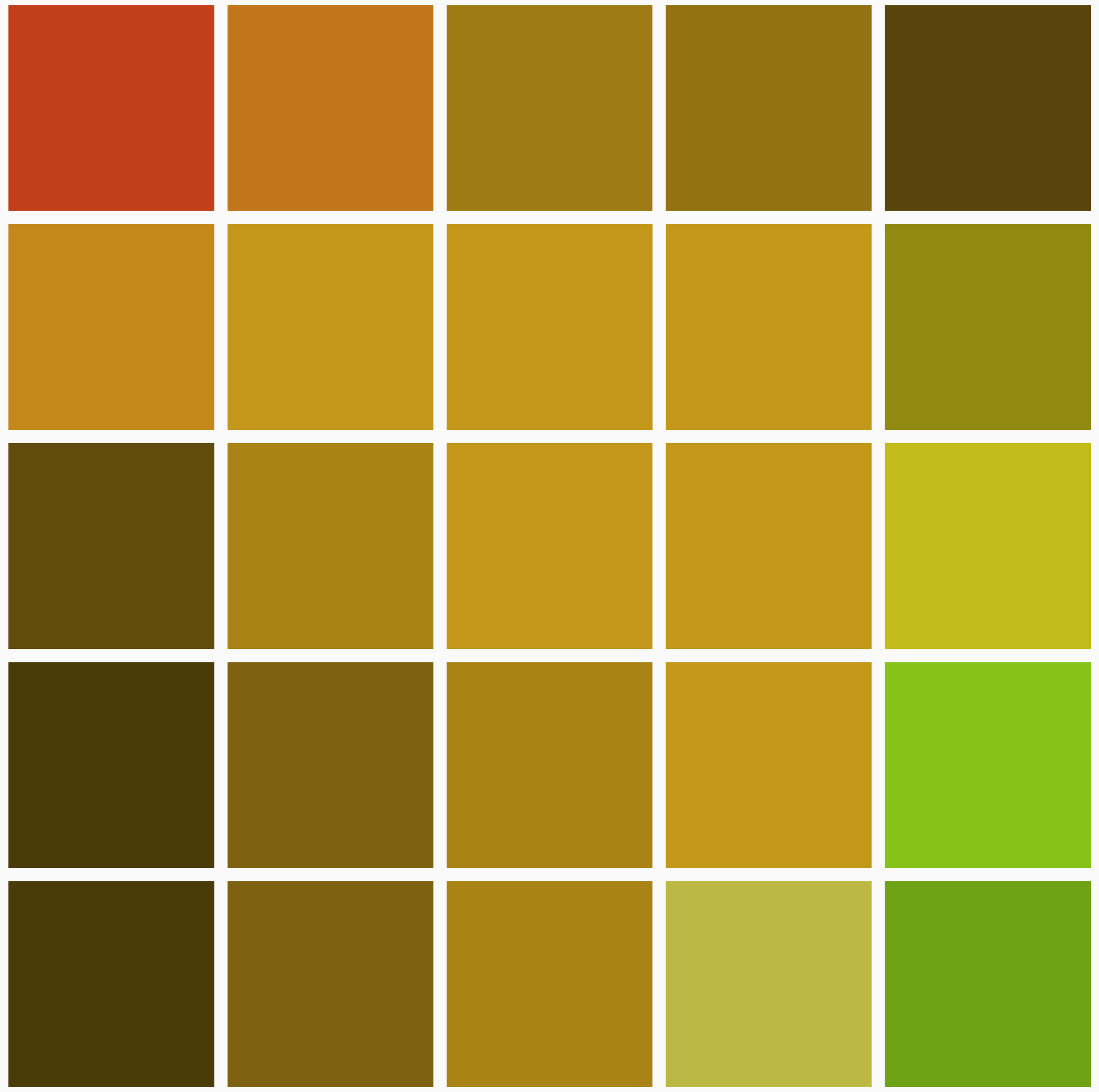
- Support **only a specific linear blend mode** (and a linear alpha addition)

➡ We **generalize** this method to support advanced (non-linear) color-blend modes

Details of Our Method [1/2]

Per-Pixel Unblending Optimization
(Concept-Level Explanation)

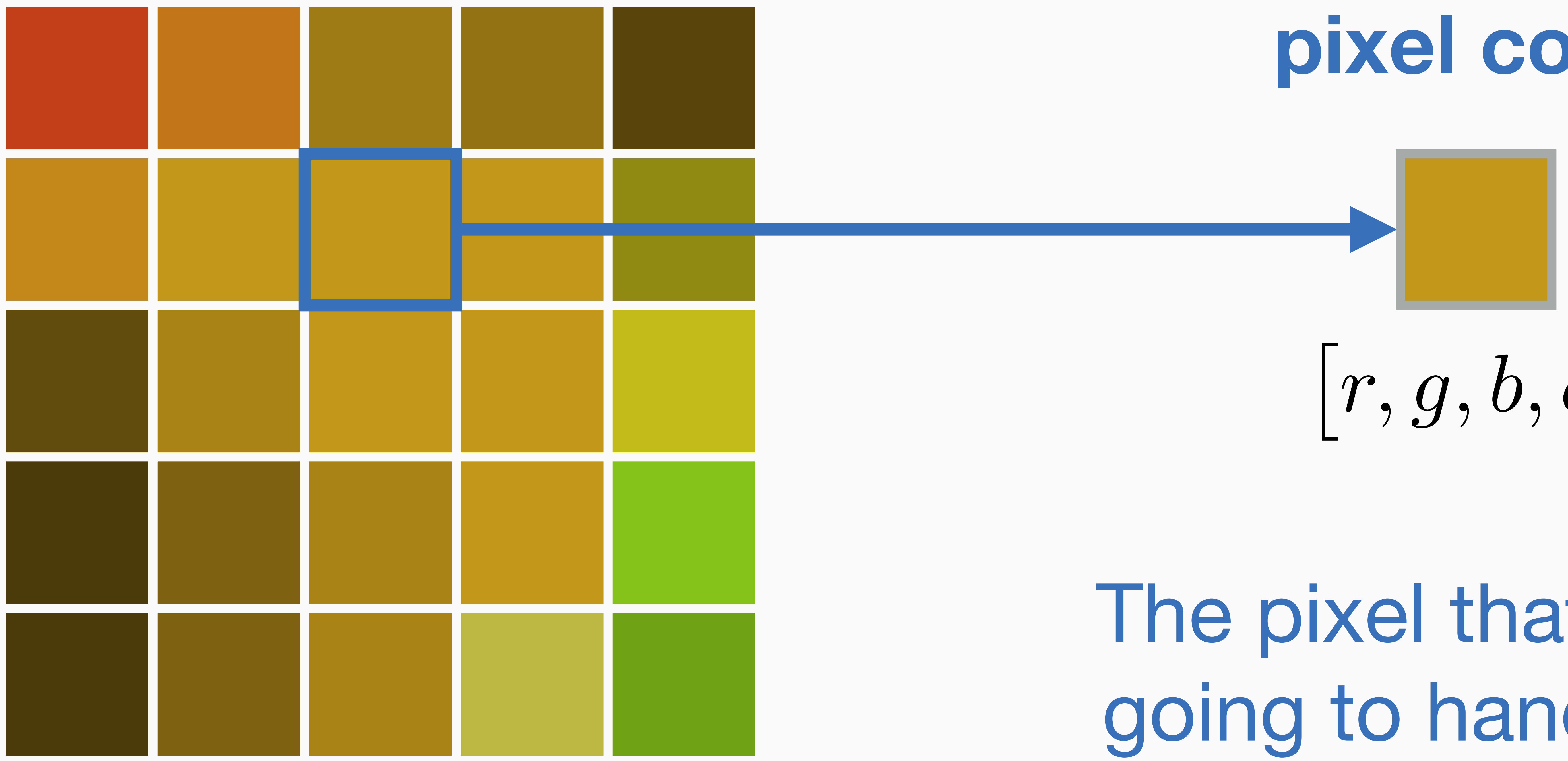
Problem Setting: Per-Pixel Unblending



Problem Setting: Per-Pixel Unblending

Target image

Target pixel color

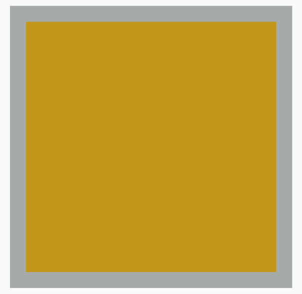


$$[r, g, b, \alpha]$$

The pixel that we are going to handle now

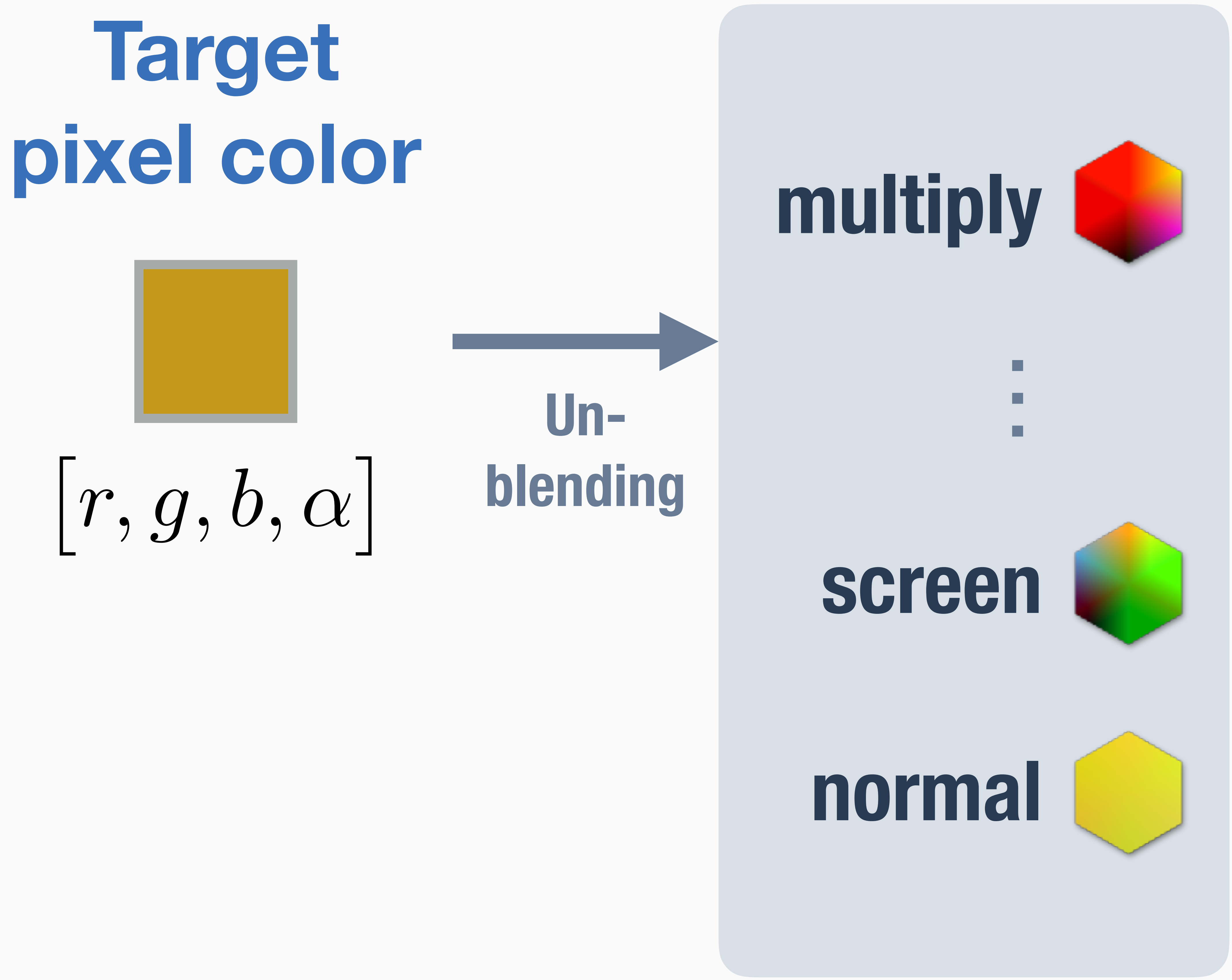
Problem Setting: Per-Pixel Unblending

Target pixel color



$$[r, g, b, \alpha]$$

Problem Setting: Per-Pixel Unblending



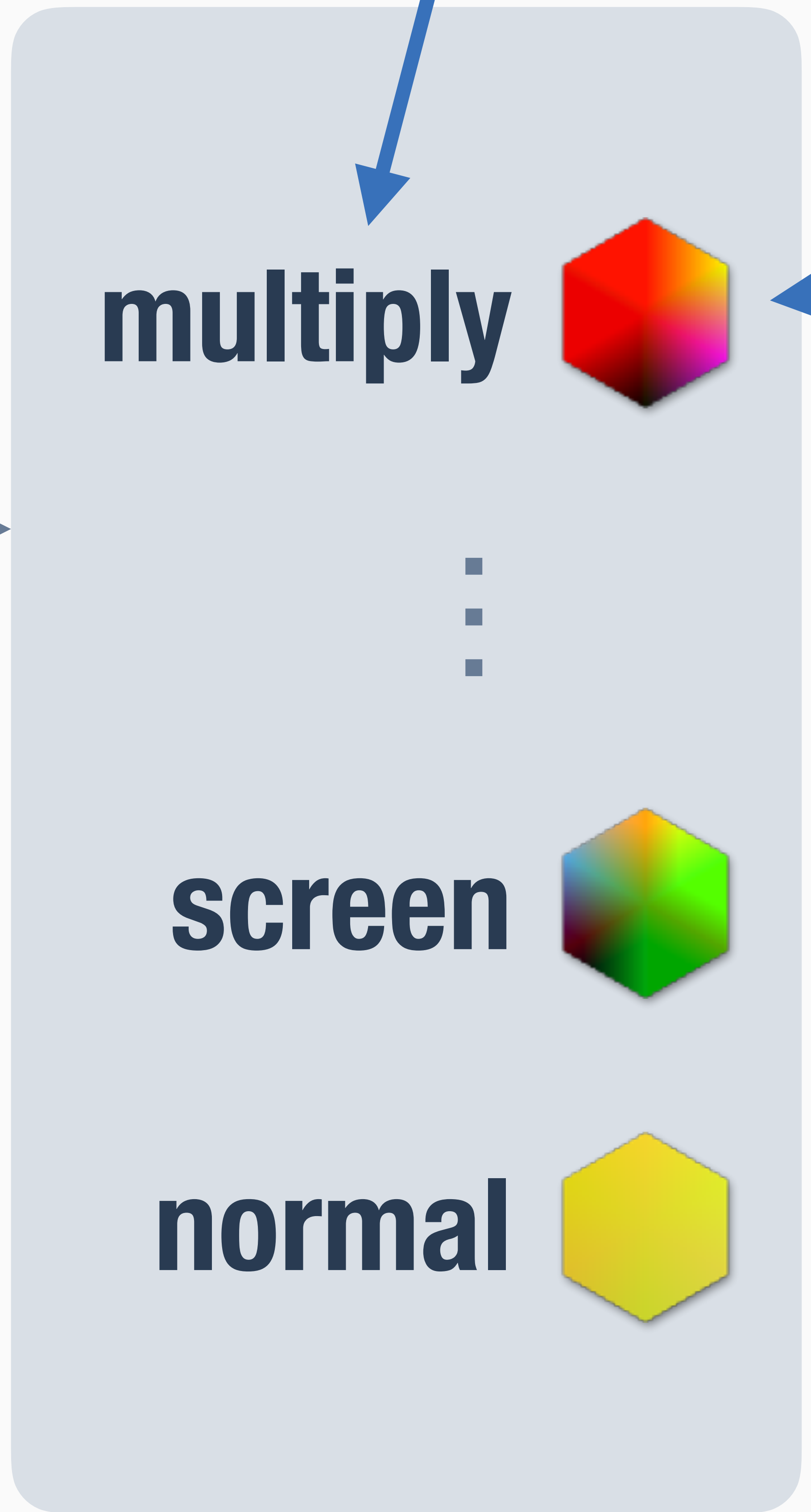
Problem Setting: Per-Pixel Unblending

Blend mode

Target pixel color

$[r, g, b, \alpha]$

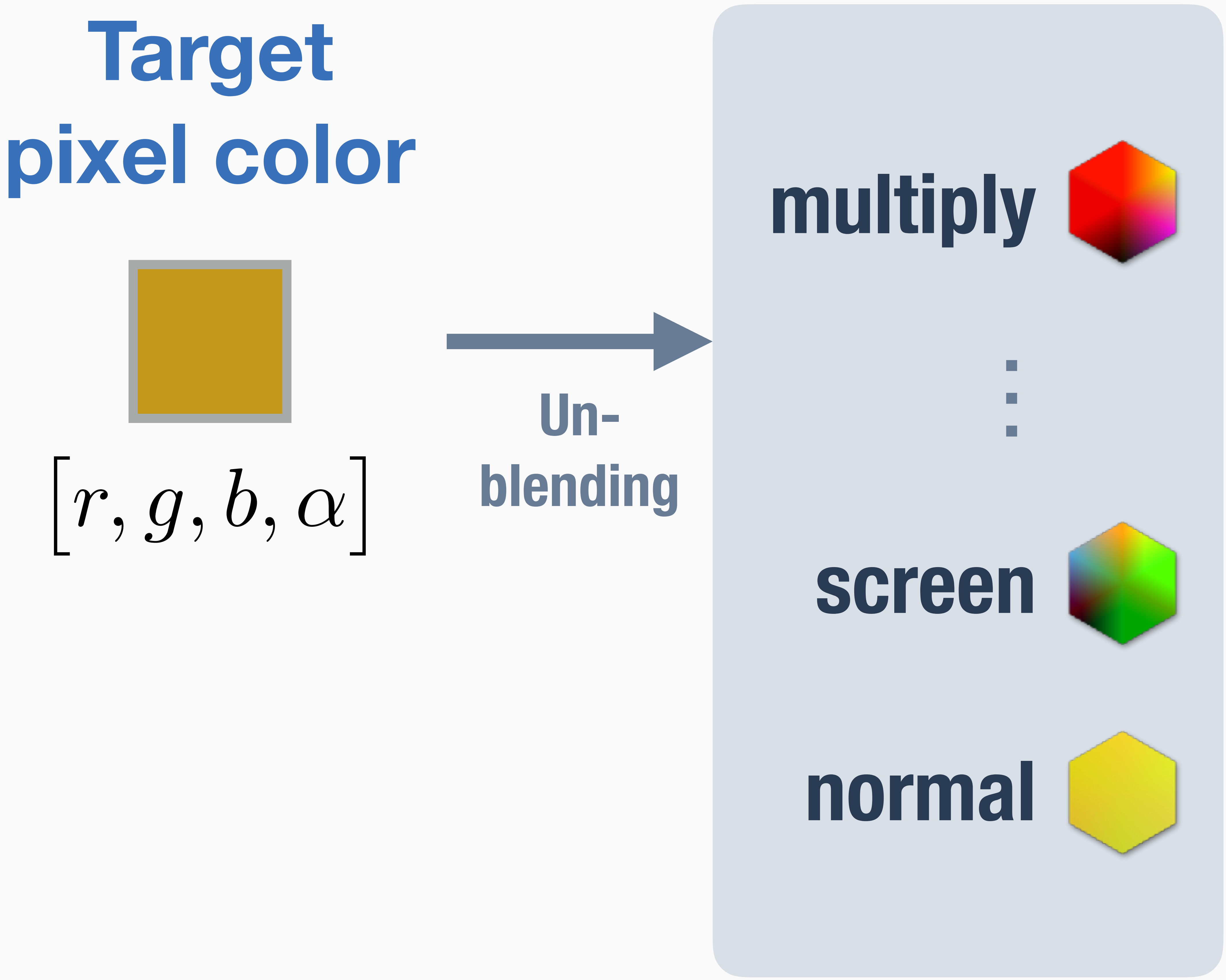
Un-blending



Desirable color distribution:
the distribution of RGB values that the decomposed layer contains

Problem Setting: Per-Pixel Unblending

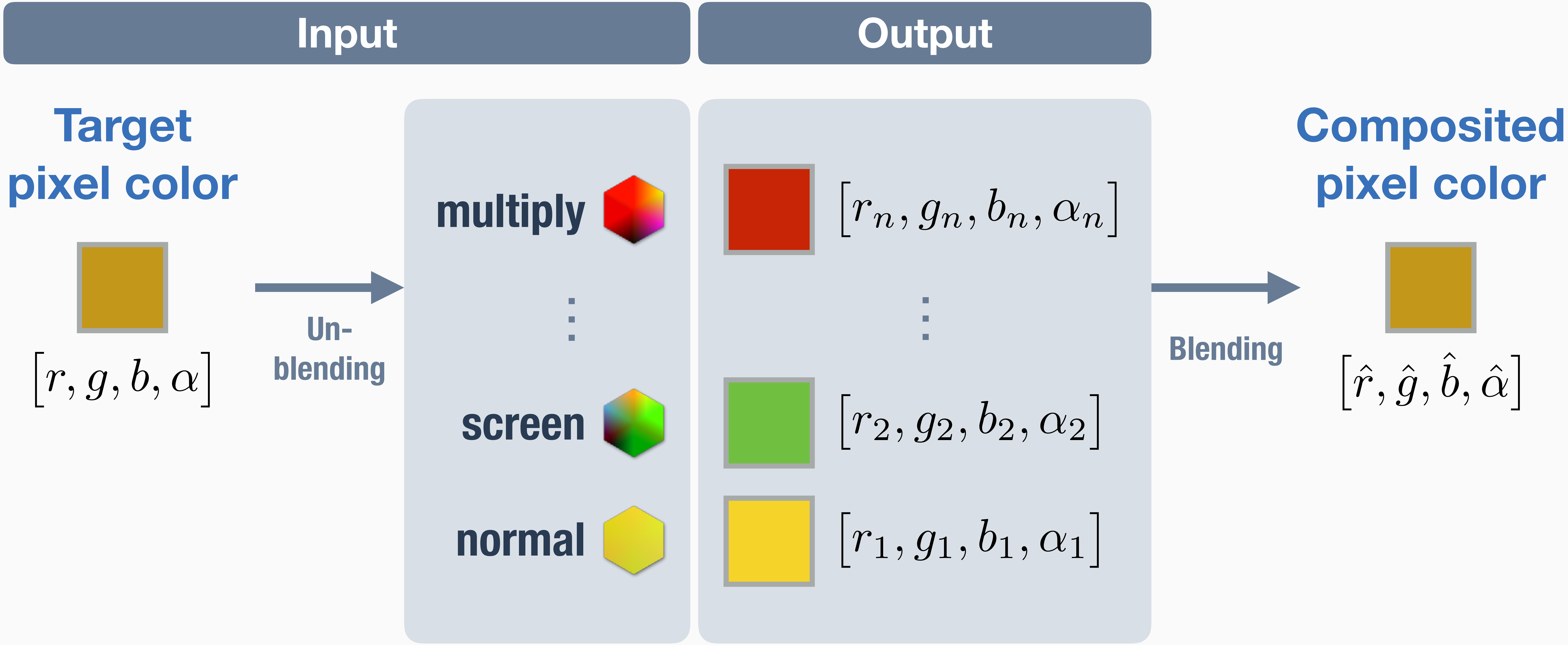
Input



Problem Setting: Per-Pixel Unblending

Input		Output
Target pixel color $[r, g, b, \alpha]$ → Un-blending	multiply	$[r_n, g_n, b_n, \alpha_n]$
	⋮	⋮
	screen	$[r_2, g_2, b_2, \alpha_2]$
	normal	$[r_1, g_1, b_1, \alpha_1]$

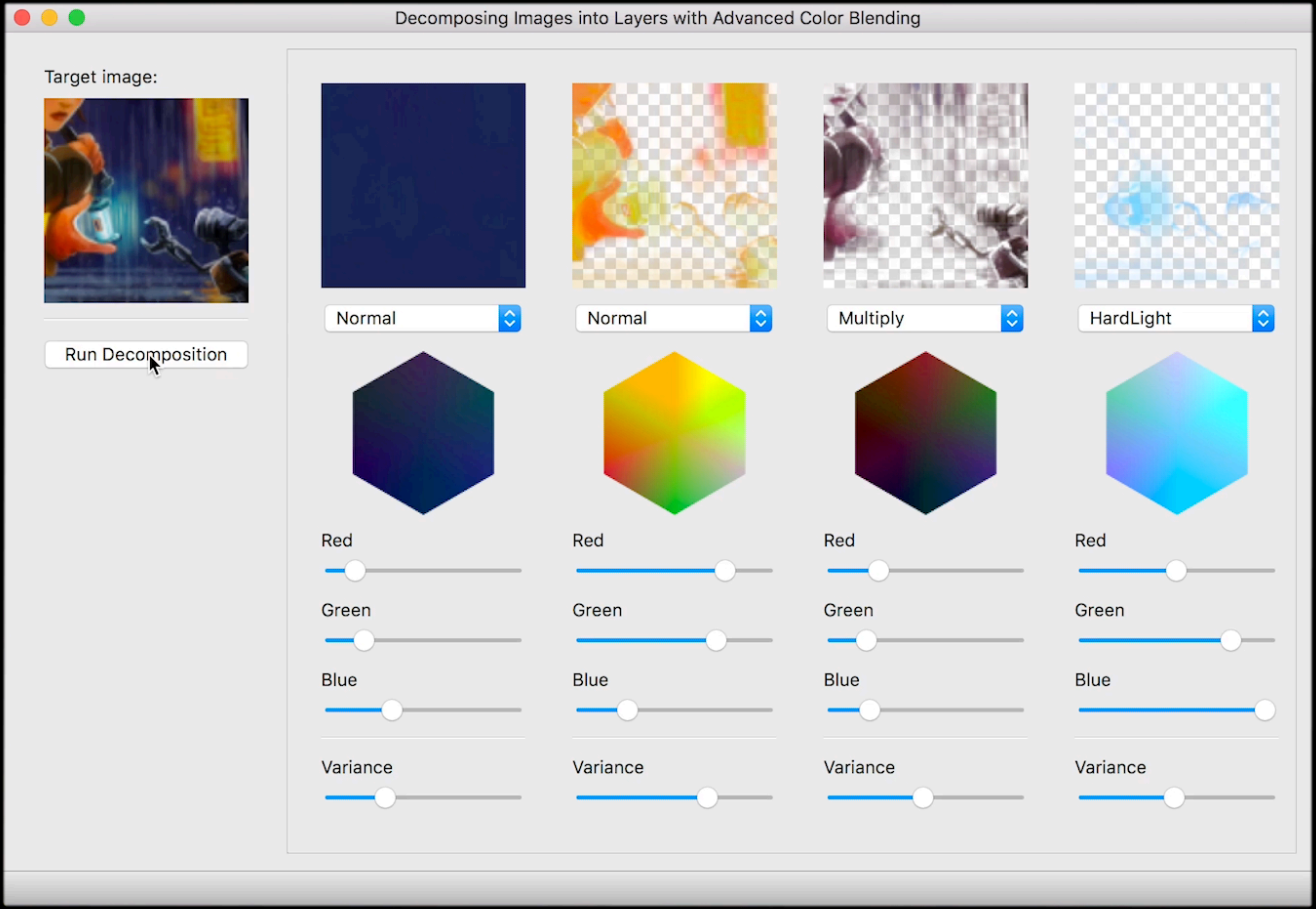
Problem Setting: Per-Pixel Unblending



Problem Setting: Per-Pixel Unblending

Problem Setting: Per-Pixel Unblending

Example User Interaction

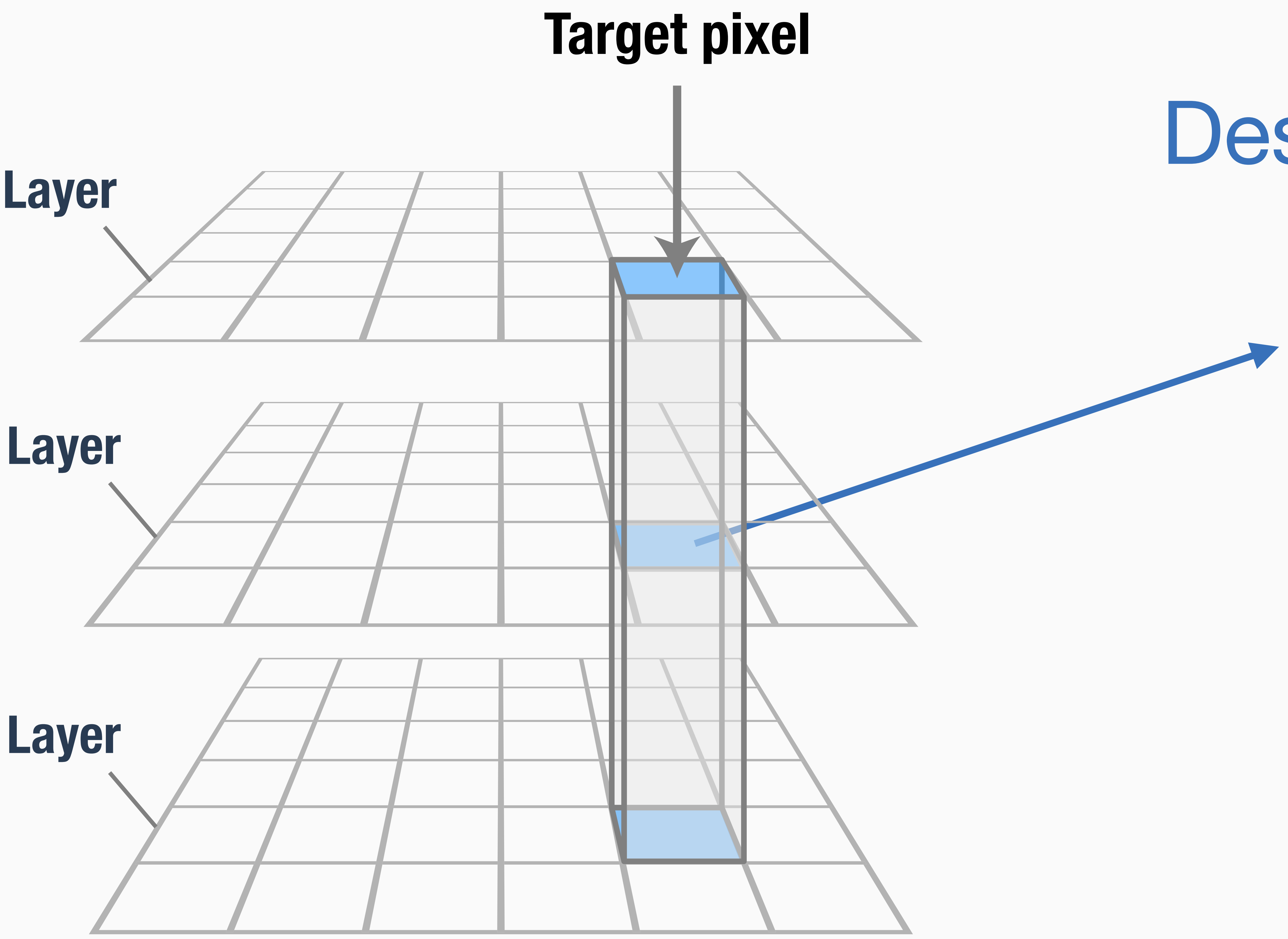


Input image courtesy of David Revoy

Basics (Not Our Method)

Math of Advanced Color Blending

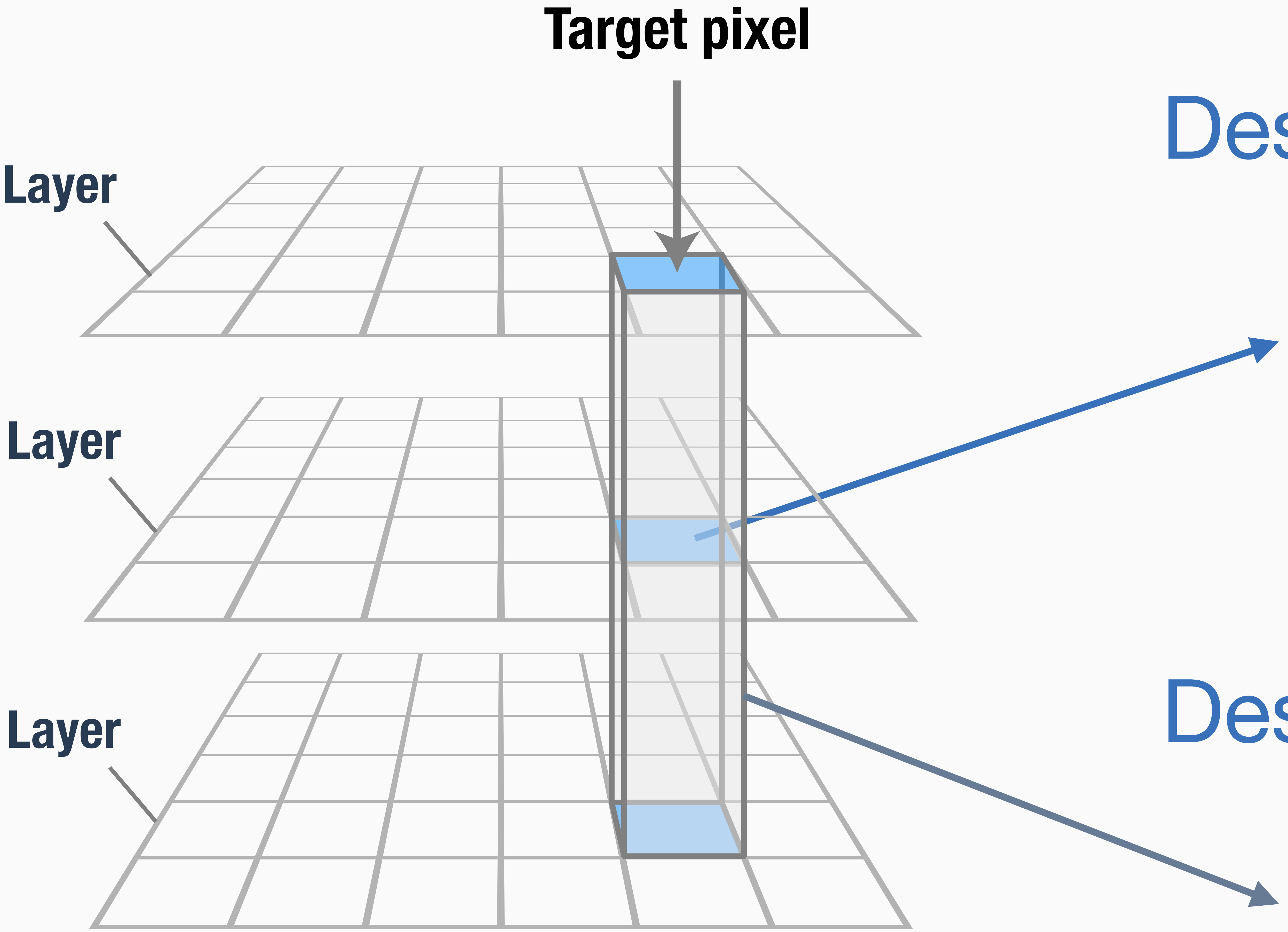
Descriptor of a Pixel Color



Descriptor of the *i*-th layer's pixel

$$\mathbf{x}_i = \begin{bmatrix} r_i & g_i & b_i & \alpha_i \end{bmatrix}^\mathrm{T} = \begin{bmatrix} \mathbf{c}_i^\mathrm{T} & \alpha_i \end{bmatrix}^\mathrm{T} \in \mathbb{R}^4$$

Descriptor of a Pixel Color



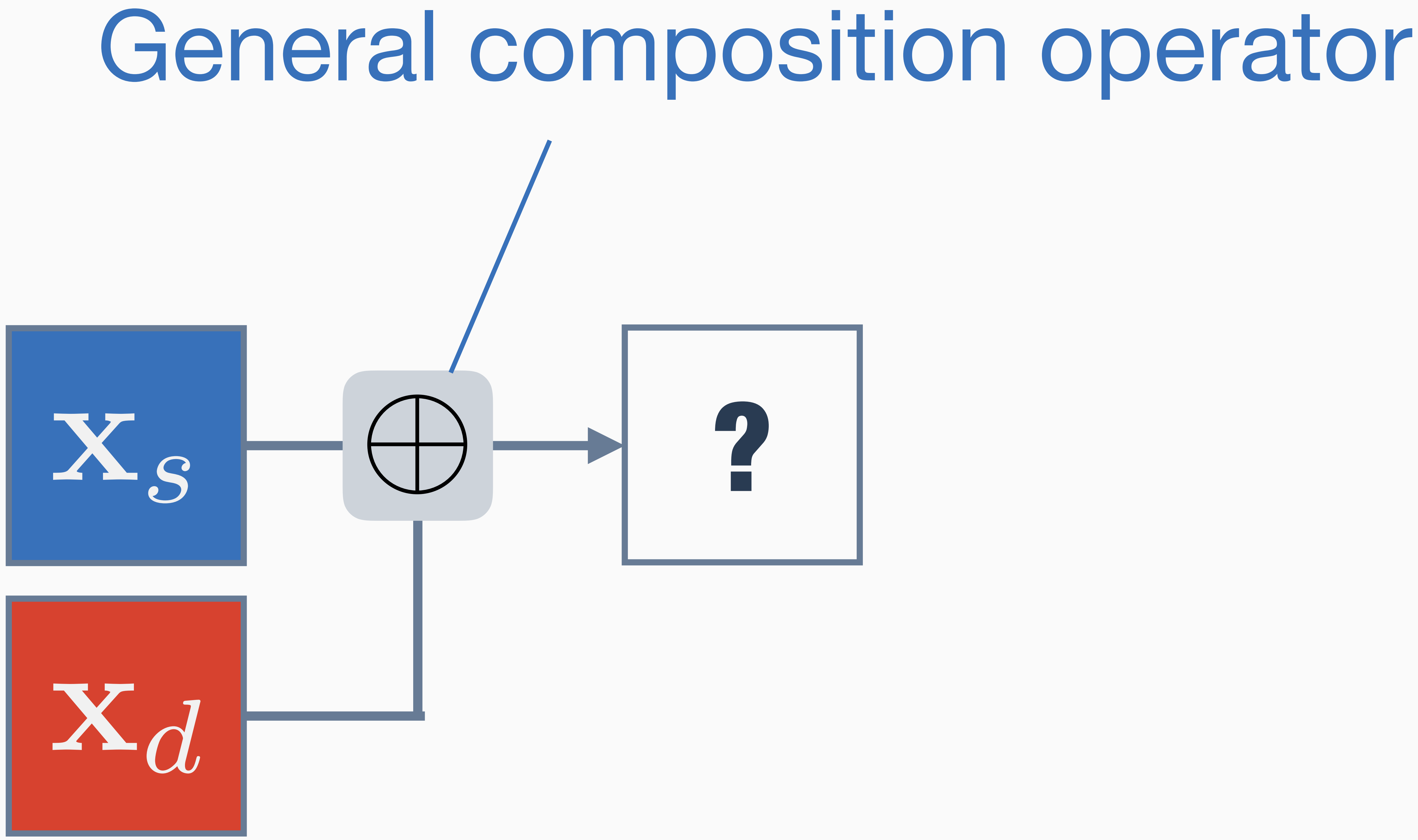
Descriptor of the *i*-th layer's pixel

$$\mathbf{x}_i = \begin{bmatrix} r_i & g_i & b_i & \alpha_i \end{bmatrix}^\mathrm{T} = \begin{bmatrix} \mathbf{c}_i^\mathrm{T} & \alpha_i \end{bmatrix}^\mathrm{T} \in \mathbb{R}^4$$

Descriptor of all the layers' pixels

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^\mathrm{T} & \cdots & \mathbf{x}_n^\mathrm{T} \end{bmatrix}^\mathrm{T} \in \mathbb{R}^{4n}$$

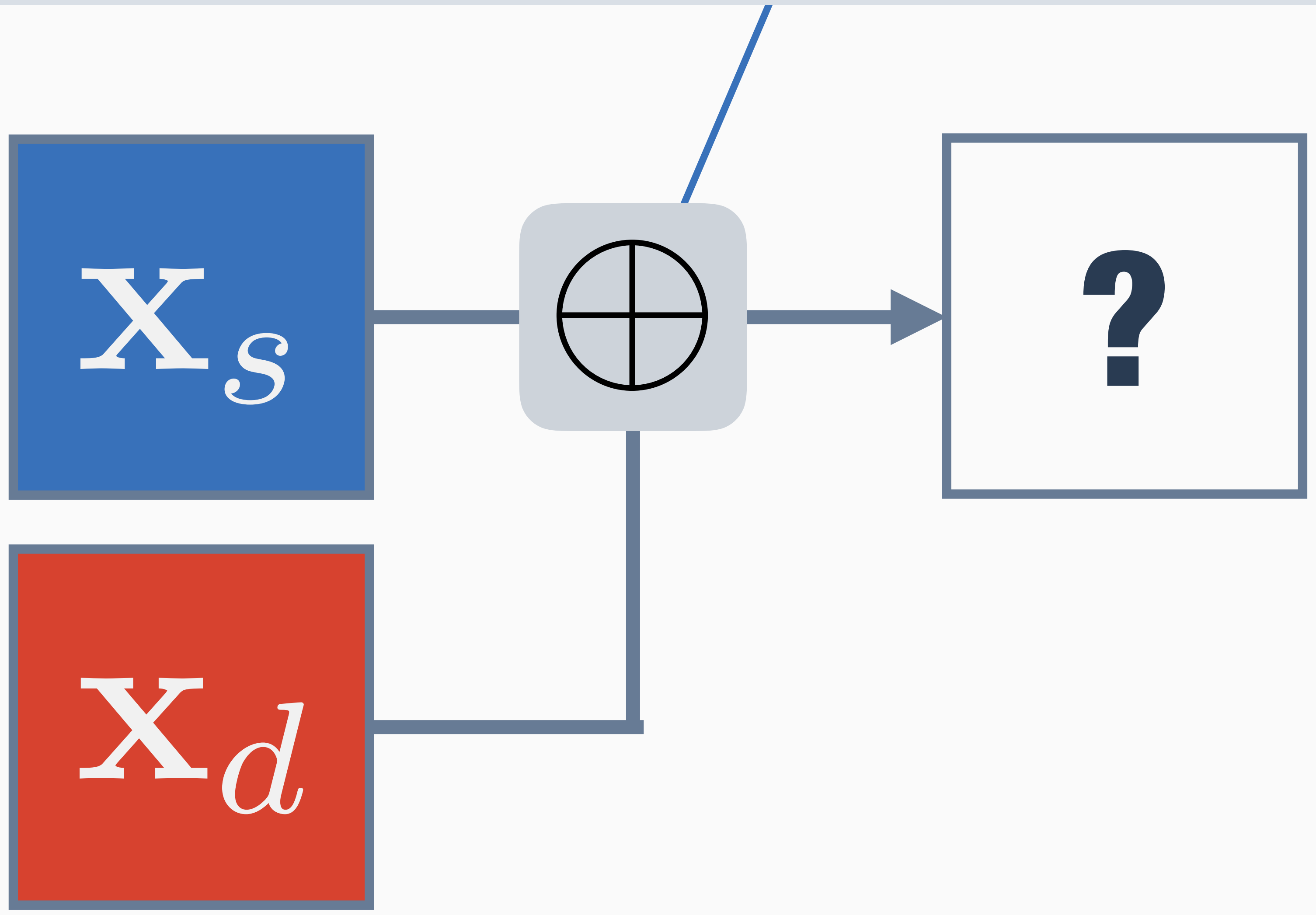
General Composition Operator ($\oplus$) for Compositing Two Layers



General Composition Operator ($\oplus$) for Compositing Two Layers

$$\mathbf{x}_s \oplus^{\mathrm{rgb}} \mathbf{x}_d = \frac{f(\mathbf{c}_s, \mathbf{c}_d)\alpha_s\alpha_d + Y\alpha_s(1-\alpha_d)\mathbf{c}_s + Z\alpha_d(1-\alpha_s)\mathbf{c}_d}{\mathbf{x}_s \oplus^{\alpha} \mathbf{x}_d}$$

$$\mathbf{x}_s \oplus^{\alpha} \mathbf{x}_d = X\alpha_s\alpha_d + Y\alpha_s(1-\alpha_d) + Z\alpha_d(1-\alpha_s)$$



“SVG Compositing Specification” by W3C: https://www.w3.org/TR/2011/WD-SVGCompositing-20110315/

General Composition Operator ($\oplus$) for Compositing Two Layers

$$\mathbf{x}_s \oplus^{\mathrm{rgb}} \mathbf{x}_d = \frac{f(\mathbf{c}_s, \mathbf{c}_d)\alpha_s\alpha_d + Y\alpha_s(1-\alpha_d)\mathbf{c}_s + Z\alpha_d(1-\alpha_s)\mathbf{c}_d}{\mathbf{x}_s \oplus^{\alpha} \mathbf{x}_d}$$

$$\mathbf{x}_s \oplus^{\alpha} \mathbf{x}_d = X\alpha_s\alpha_d + Y\alpha_s(1-\alpha_d) + Z\alpha_d(1-\alpha_s)$$

Blend function: Mode-specific function (can be non-linear)

$$f : [0,1]^3 \times [0,1]^3 \to [0,1]^3$$

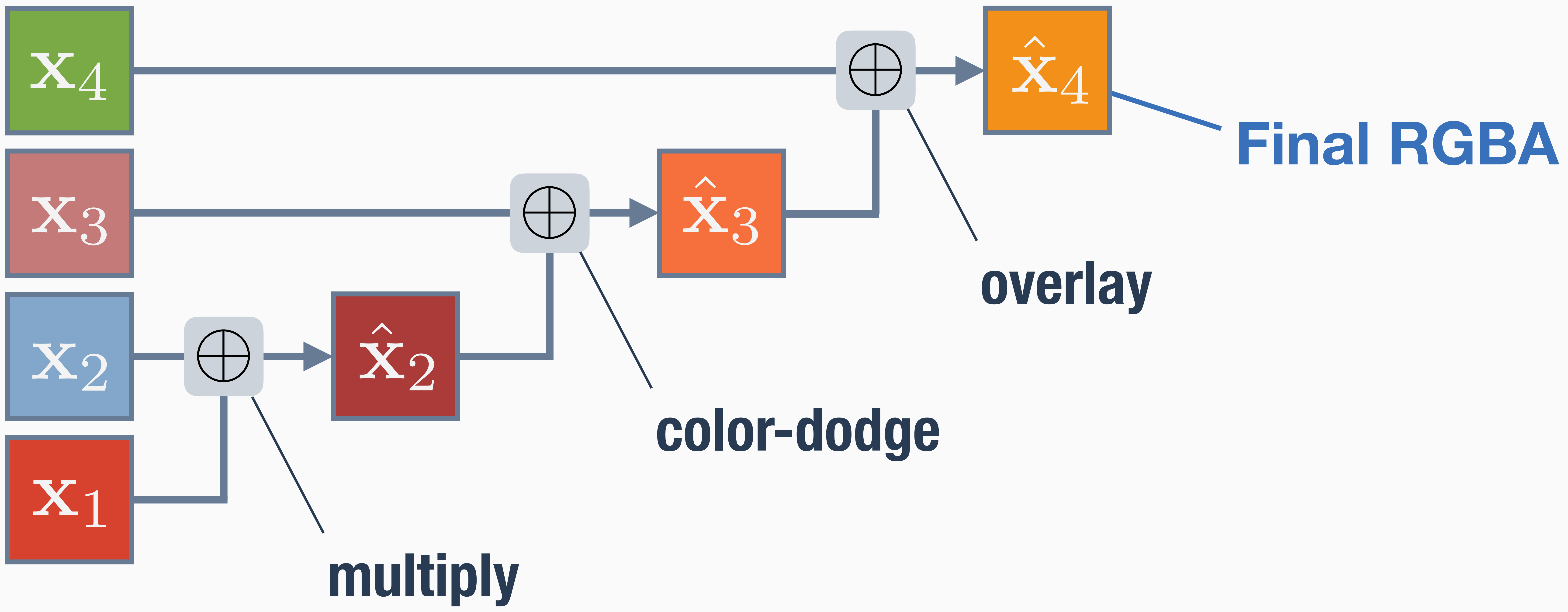
For example: $f_{\texttt{normal}}, f_{\texttt{multiply}}, f_{\texttt{color-dodge}}, \cdots$

“SVG Compositing Specification” by W3C: https://www.w3.org/TR/2011/WD-SVGCompositing-20110315/

More-Than-Two-Layers Case: Recursive Layering

Recursive rule:

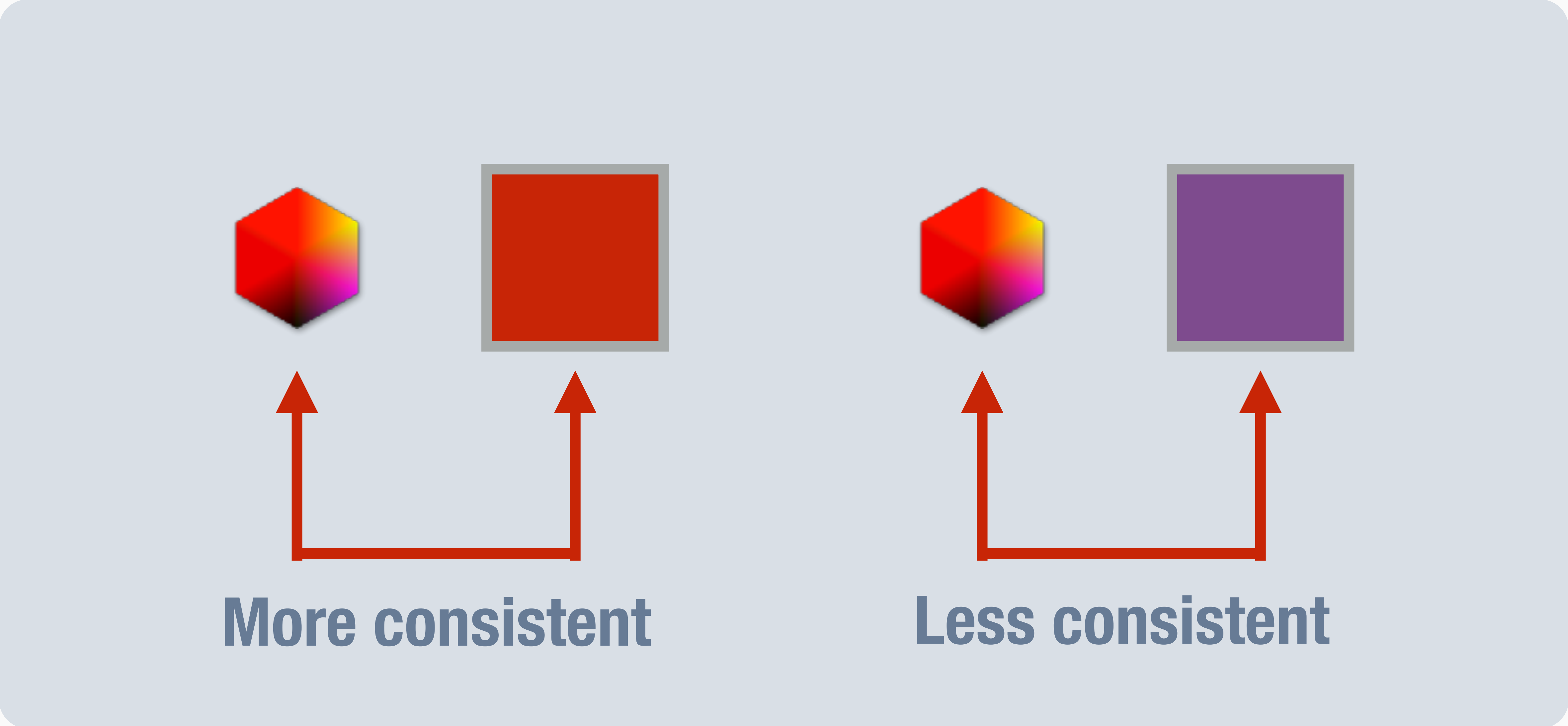
$$\hat{\mathbf{x}}_k = \begin{cases} \mathbf{x}_1 & (k = 1) \\ \mathbf{x}_k \oplus_k \hat{\mathbf{x}}_{k-1} & (\text{otherwise}) \end{cases}$$

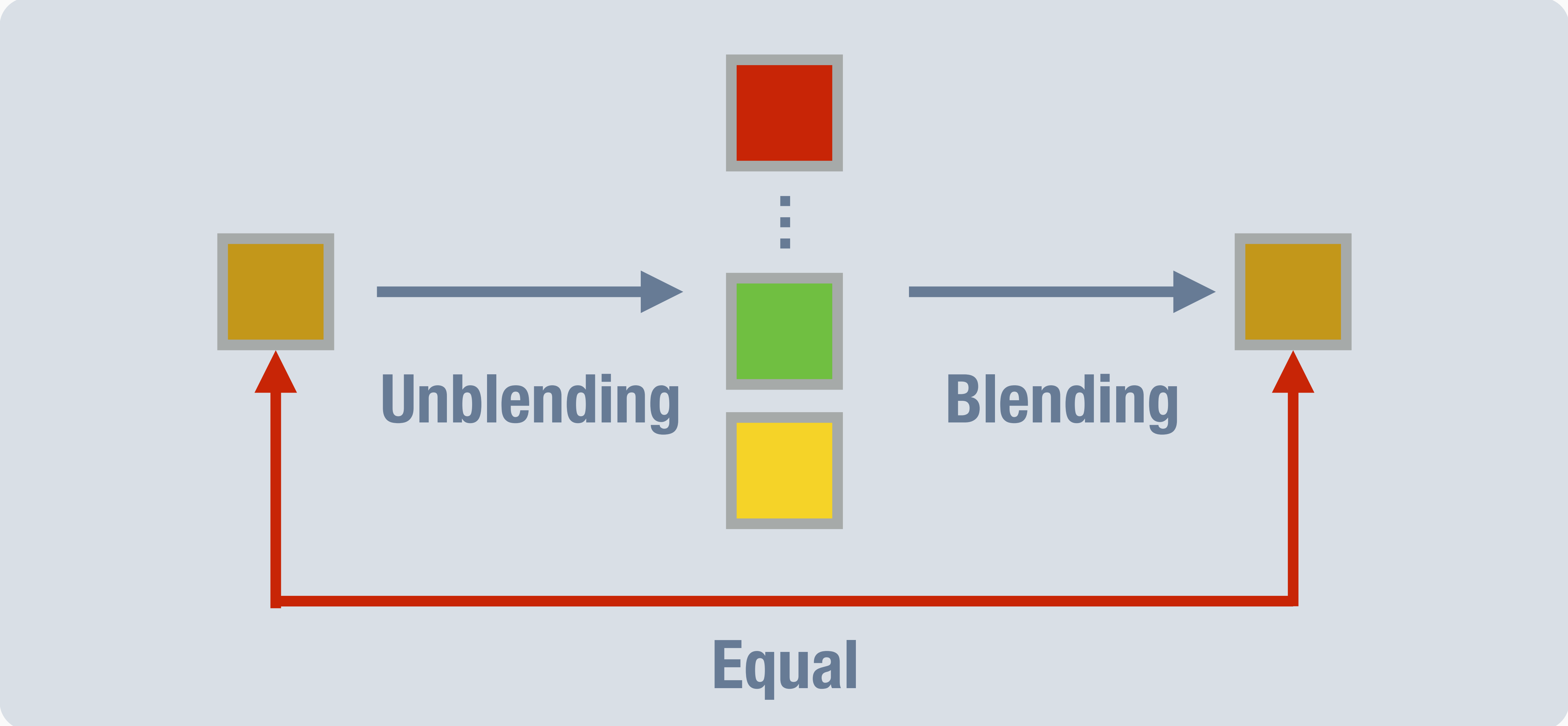


Details of Our Method [2/2]

Per-Pixel Unblending Optimization (Equation-Level Explanation)

Unblending Optimization





Color consistency:
Each pixel color should be as consistent to the specified color distribution as possible

Color reproducibility:
The target pixel color should be reproduced by compositing the resulting layer colors

Unblending Optimization

Equality-constrained optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^{4n}} \; E(\mathbf{x}) \;\; \text{s.t.} \;\; \mathbf{C}(\mathbf{x}) = \mathbf{0}$$

Objective:
Each pixel color should be as consistent to the specified color distribution as possible

Equality constraint:
The target pixel color should be reproduced by compositing the resulting layer colors

Unblending Optimization

Equality-constrained optimization problem

$$\min_{\mathbf{x} \in \mathbb{R}^{4n}} E(\mathbf{x}) \text{ s.t. } \mathbf{C}(\mathbf{x}) = \mathbf{0}$$

Objective:
Each pixel color should be as consistent to the specified color distribution as possible

Equality constraint:
The target pixel color should be reproduced by compositing the resulting layer colors

Constraint Function Definition

$$\mathbf{C}(\mathbf{x}) = \left(\mathbf{x}_n \oplus_n \left(\mathbf{x}_{n-1} \oplus_{n-1} \left(\mathbf{x}_{n-2} \oplus_{n-2} \left(\ldots\right)\right)\right)\right) - \mathbf{c}^{\mathrm{target}} = \mathbf{0}$$

E.g., multiply color-dodge overlay ⋯

Constraint Function Definition

$$\mathbf{C}(\mathbf{x}) = (\mathbf{x}_n \oplus_n (\mathbf{x}_{n-1} \oplus_{n-1} (\mathbf{x}_{n-2} \oplus_{n-2} (\ldots)))) - \mathbf{c}^{\mathrm{target}} = \mathbf{0}$$

E.g., multiply color-dodge overlay ...

C.f., [Aksoy+16; 17]: Assumed a **special case** of a linear model

$$\mathbf{C}(\mathbf{x}) = \sum_i^n \alpha_i \mathbf{c}_i - \mathbf{c}^{\mathrm{target}} = \mathbf{0}$$

Constraint Function Definition

$$\mathbf{C}(\mathbf{x}) = \left(\mathbf{x}_n \oplus_n \left(\mathbf{x}_{n-1} \oplus_{n-1} \left(\mathbf{x}_{n-2} \oplus_{n-2} \left(\ldots\right)\right)\right)\right) - \mathbf{c}^{\text{target}} = \mathbf{0}$$

E.g., multiply color-dodge overlay ⋯

C.f., [Aksoy+16; 17]: Assumed a **special case** of a linear model

$$\mathbf{C}(\mathbf{x}) = \sum_{i}^{n} \alpha_i \mathbf{c}_i - \mathbf{c}^{\text{target}} = \mathbf{0}$$

➡ Our method is a **generalization** of [Aksoy+16; 17]

Solving Equality-Constrained Optimization [1/2]

Algorithm: The augmented Lagrangian method

- Iteratively solve (unconstrained) optimizations

while (not converged) {

solve $\min_{\mathbf{x}} \left\{ E(\mathbf{x}) - \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{C}(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{C}(\mathbf{x})\|^2 \right\}$

update $\boldsymbol{\lambda}$ and ρ

}

Solving Equality-Constrained Optimization [1/2]

Algorithm: The augmented Lagrangian method

- Iteratively solve (unconstrained) optimizations

while (not converged) {

solve $\min_{\mathbf{x}} \left\{ E(\mathbf{x}) - \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{C}(\mathbf{x}) + \frac{\rho}{2} \| \mathbf{C}(\mathbf{x}) \|^2 \right\}$

update $\boldsymbol{\lambda}$ and ρ

}

Need the **derivative of C(x)** to efficiently solve this optimization

Solving Equality-Constrained Optimization [2/2]

It is possible to algorithmically calculate the derivative of C(x) by recursively applying the chain rule

$$\frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}_i} = \begin{cases} \mathbf{I} & (i = k = 1) \\ \frac{\partial}{\partial \mathbf{x}_k}(\mathbf{x}_k \oplus_k \hat{\mathbf{x}}_{k-1}) & (i = k \neq 1) \\ \frac{\partial \hat{\mathbf{x}}_{k-1}}{\partial \mathbf{x}_i} \cdot \frac{\partial}{\partial \hat{\mathbf{x}}_{k-1}}(\mathbf{x}_k \oplus_k \hat{\mathbf{x}}_{k-1}), & (\text{otherwise}) \end{cases}$$

Solving Equality-Constrained Optimization [2/2]

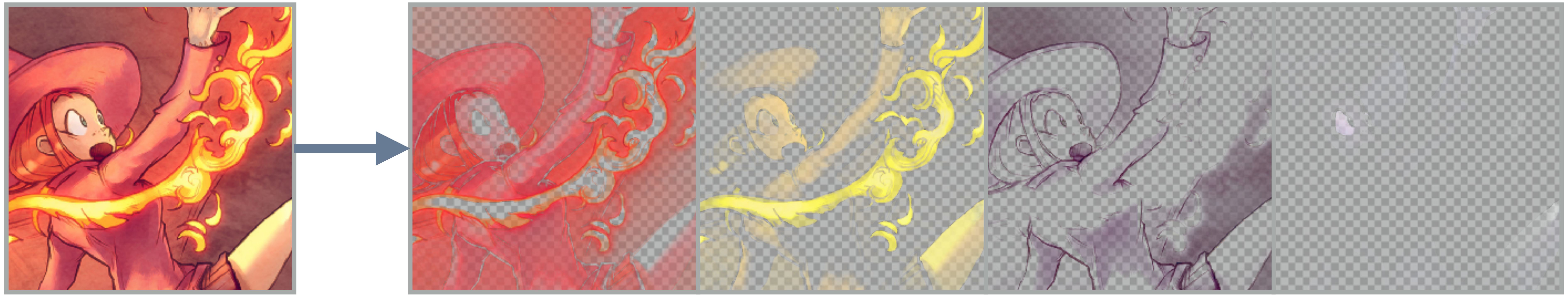
It is possible to algorithmically calculate the derivative of C(x) by recursively applying the chain rule

$$\frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}_i} = \begin{cases} \mathbf{I} & (i = k = 1) \\ \frac{\partial}{\partial \mathbf{x}_k}(\mathbf{x}_k \oplus_k \hat{\mathbf{x}}_{k-1}) & (i = k \neq 1) \\ \frac{\partial \hat{\mathbf{x}}_{k-1}}{\partial \mathbf{x}_i} \cdot \frac{\partial}{\partial \hat{\mathbf{x}}_{k-1}}(\mathbf{x}_k \oplus_k \hat{\mathbf{x}}_{k-1}), & (\text{otherwise}) \end{cases}$$

➡ Our unblending optimization problem can be efficiently solved by **gradient-based algorithms** (e.g., L-BFGS)

Unblending using linear blend mode

➡ Identical to the results by [Aksoy+17]



Unblending using various blend modes

Input image courtesy of David Revoy

Post-Processing

Refinement for Smoothness

Refinement Step for Smoothness

- **Problem:**

Resulting layers may not be smooth because the unblending optimization is "per-pixel" (no inter-pixel consideration)

Input image courtesy of David Revoy

Refinement Step for Smoothness

- **Problem:**

Resulting layers may not be smooth because the unblending optimization is "per-pixel" (no inter-pixel consideration)

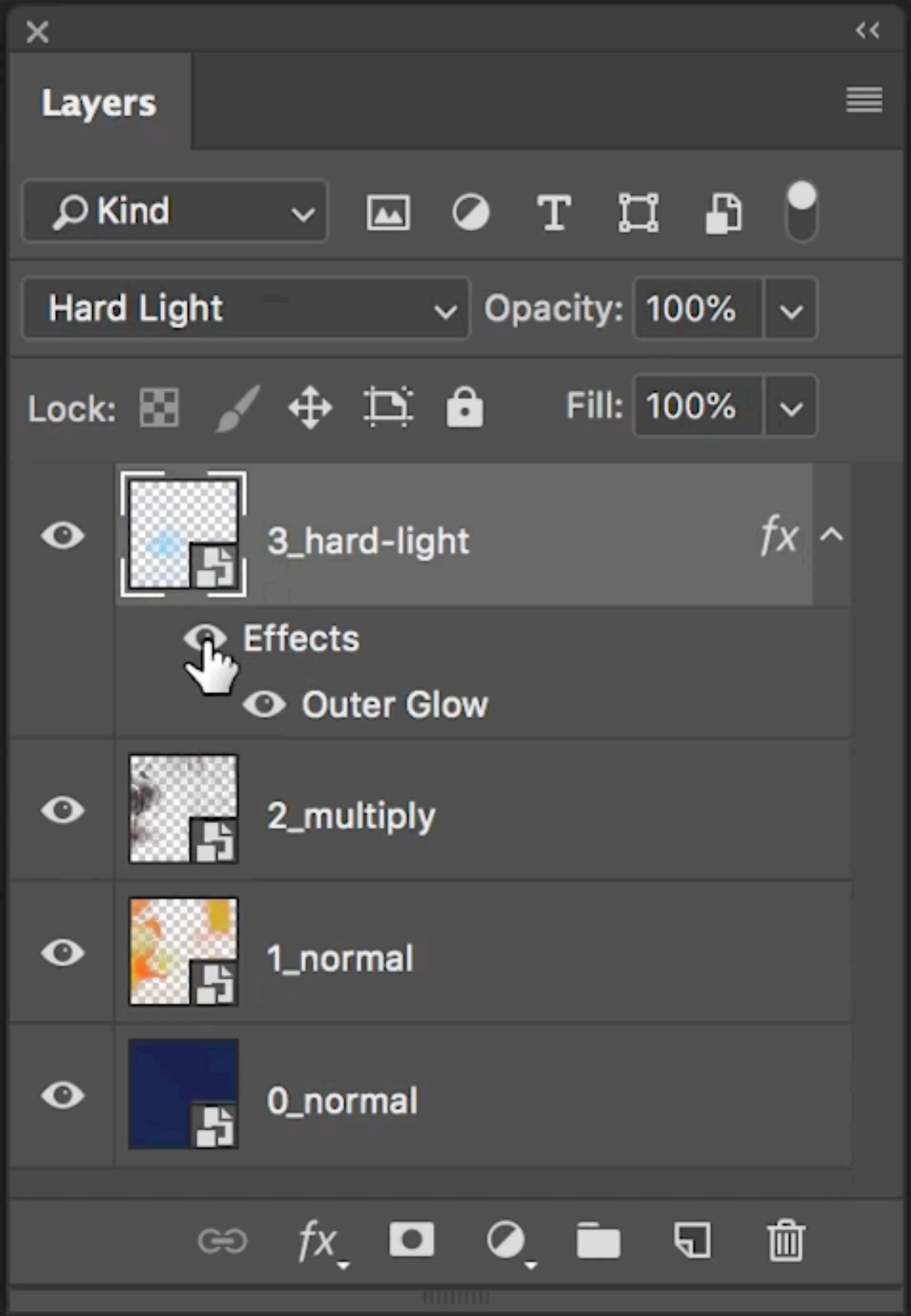
- **Solution:**

Performing a procedure called "**matte refinement**" [Aksoy+17] as a post processing to enforce smoothness

Input image courtesy of David Revoy

Usage Scenario

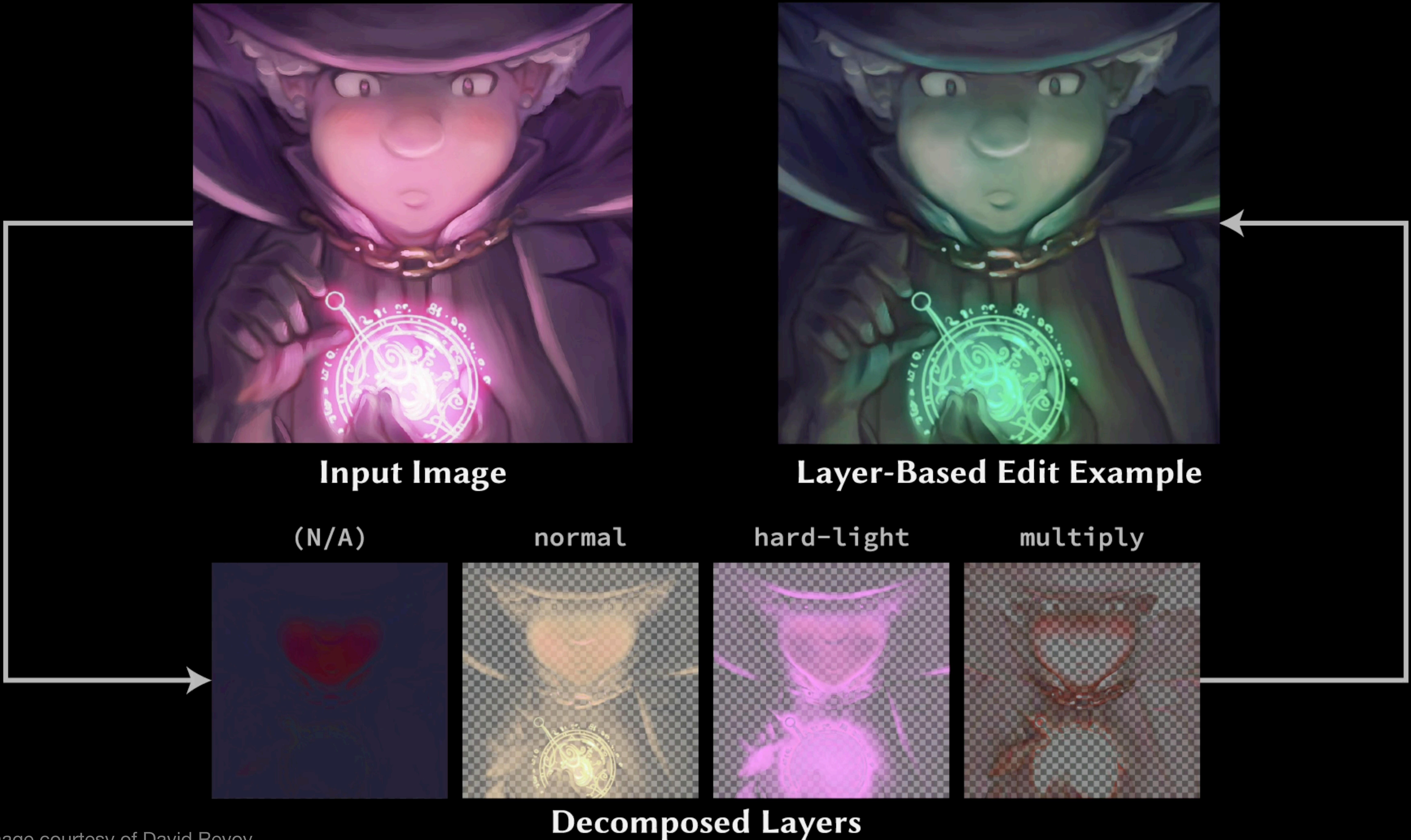
Example: Photoshop



Input image courtesy of David Revoy

Applications #1

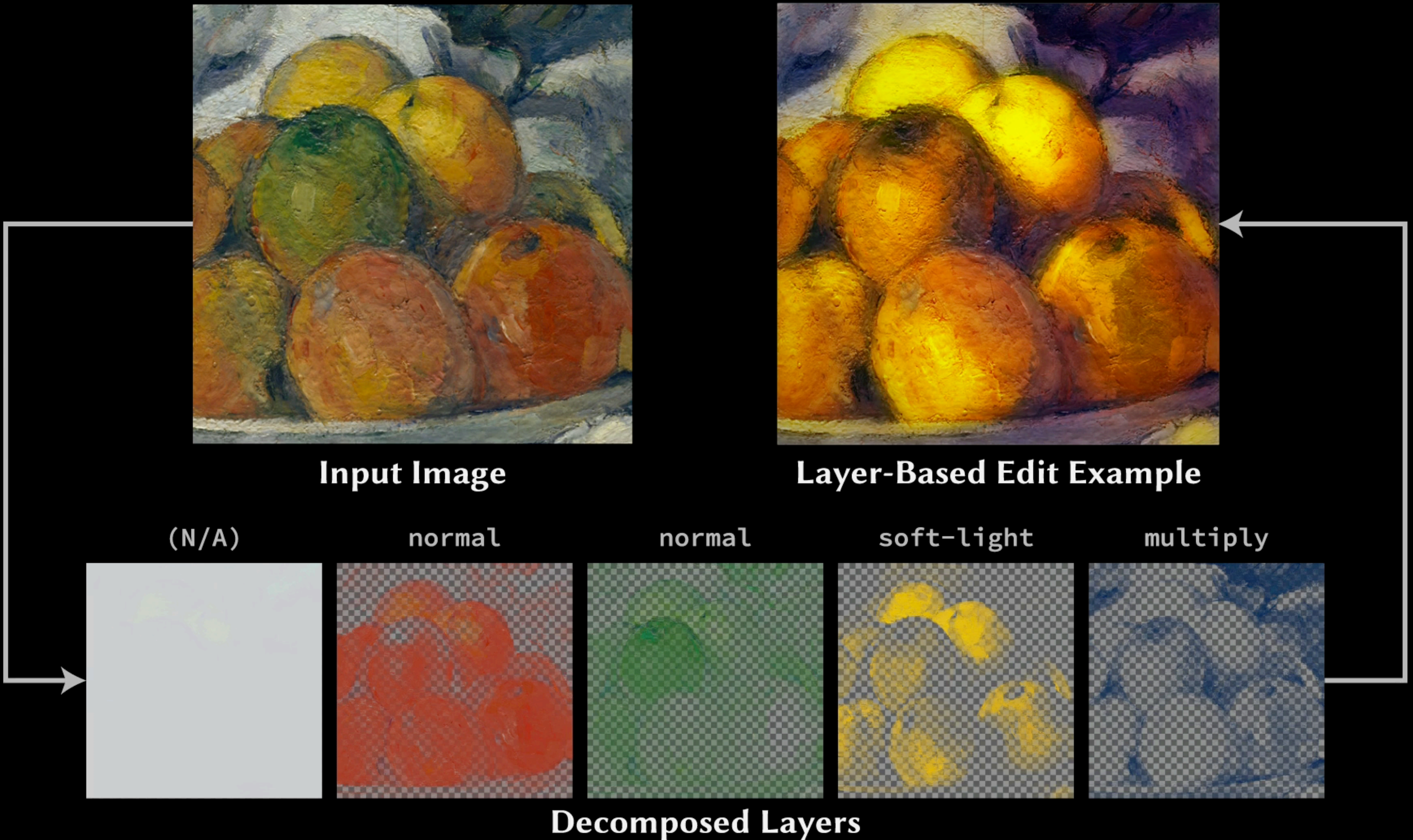
Lighting-Aware Editing of an Existing Illustration



Input image courtesy of David Revoy

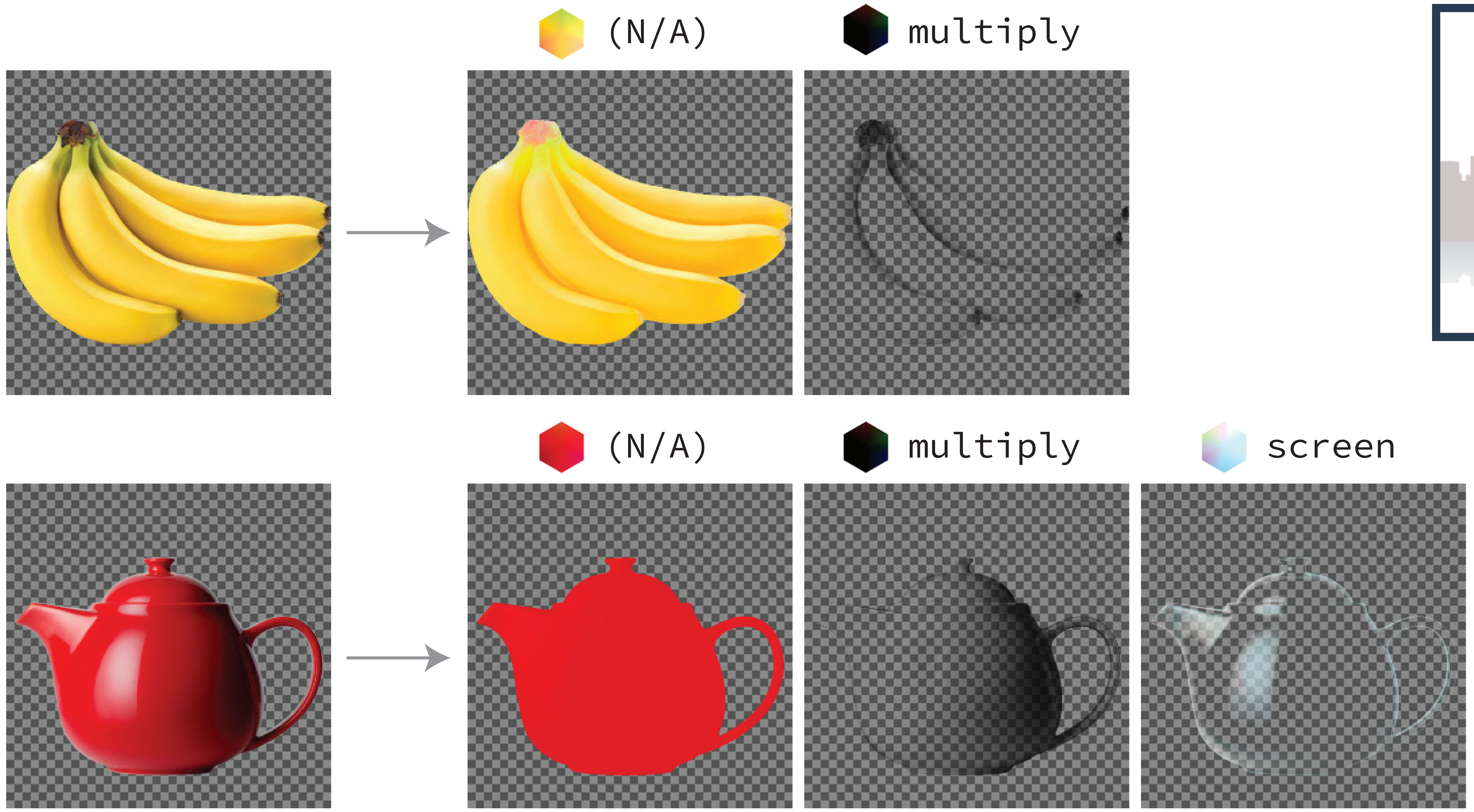
Applications #2

Bringing a Physical Painting into Digital Workflow


Input Image
Layer-Based Edit Example
(N/A)
normal
normal
soft-light
multiply
Decomposed Layers

Other Applications

Intrinsic image decomposition: decomposing an image into reflectance and shading layers

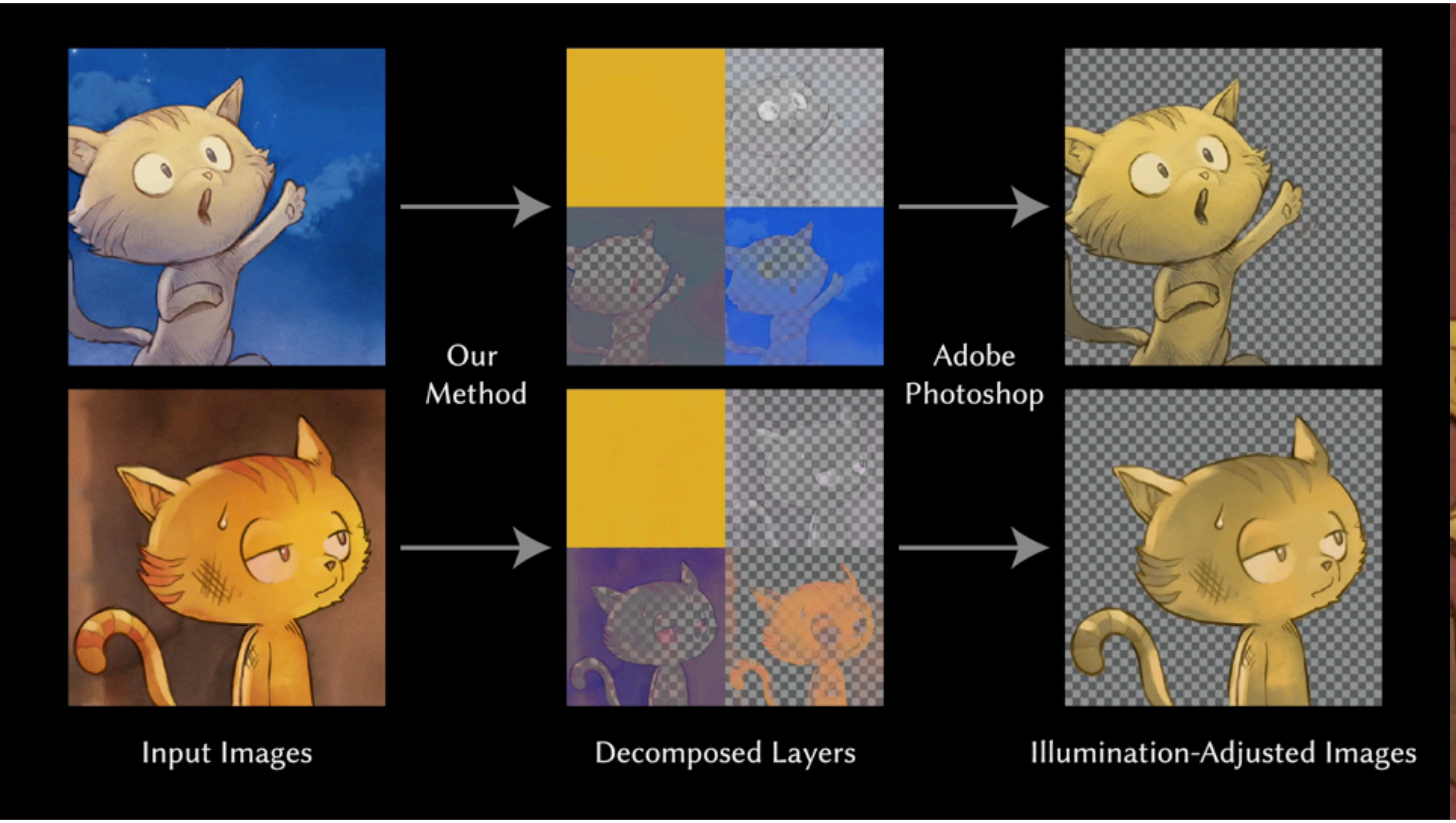


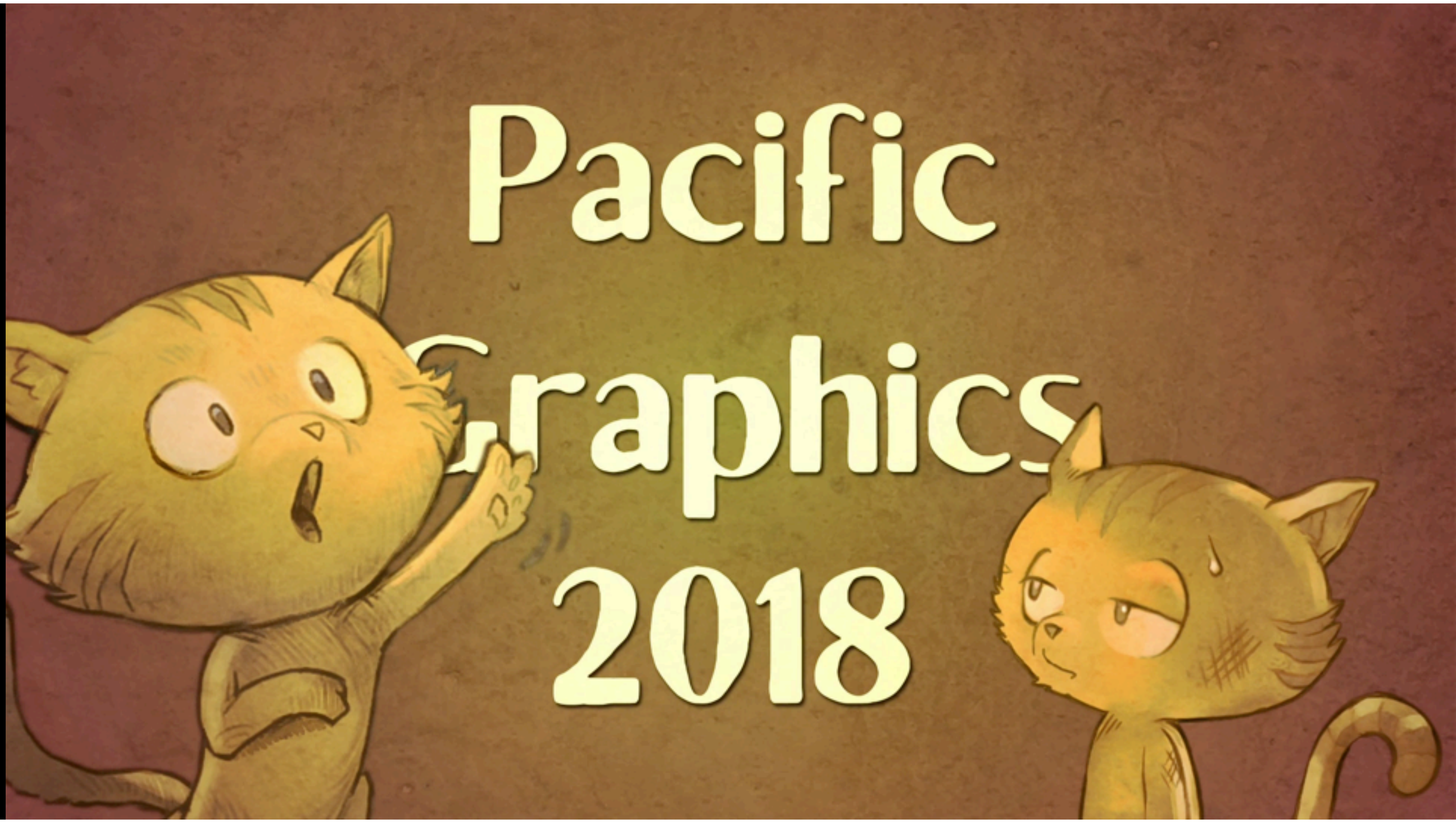
Adding gradation to an existing conference logo: extracting a "fill" layer from the logo and re-fill with gradation

Input images are provided by [Innamorati+17]

Remixing existing illustrations:

Harmonize colors of images from different scenes to put them together in a derivative work





Input image courtesy of David Revoy

Summary

Summary

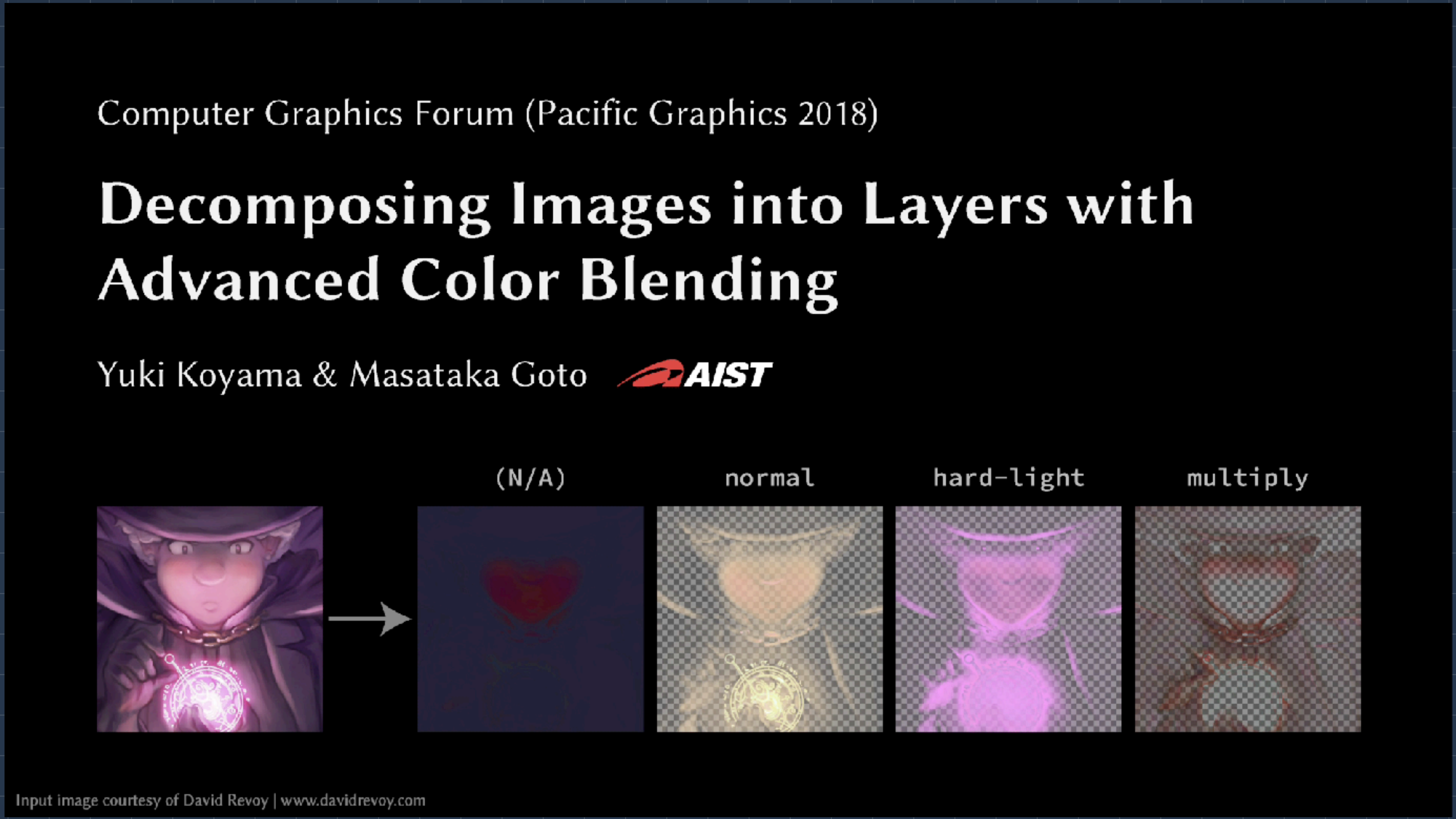
- **Solution for a new problem**
  - Decomposing an image into layers with **advanced (non-linear) color blending**
- **Techniques**
  - **Generalizing** Aksoy+'s method [16; 17] for handling advanced (non-linear) color blend modes
- **Future work**
  - Automatic determination of optimal color distributions etc.

In preparation...

We are preparing for releasing our source codes at **GitHub**
so please check our project page later

Decomposing Images into Layers with Advanced Color Blending

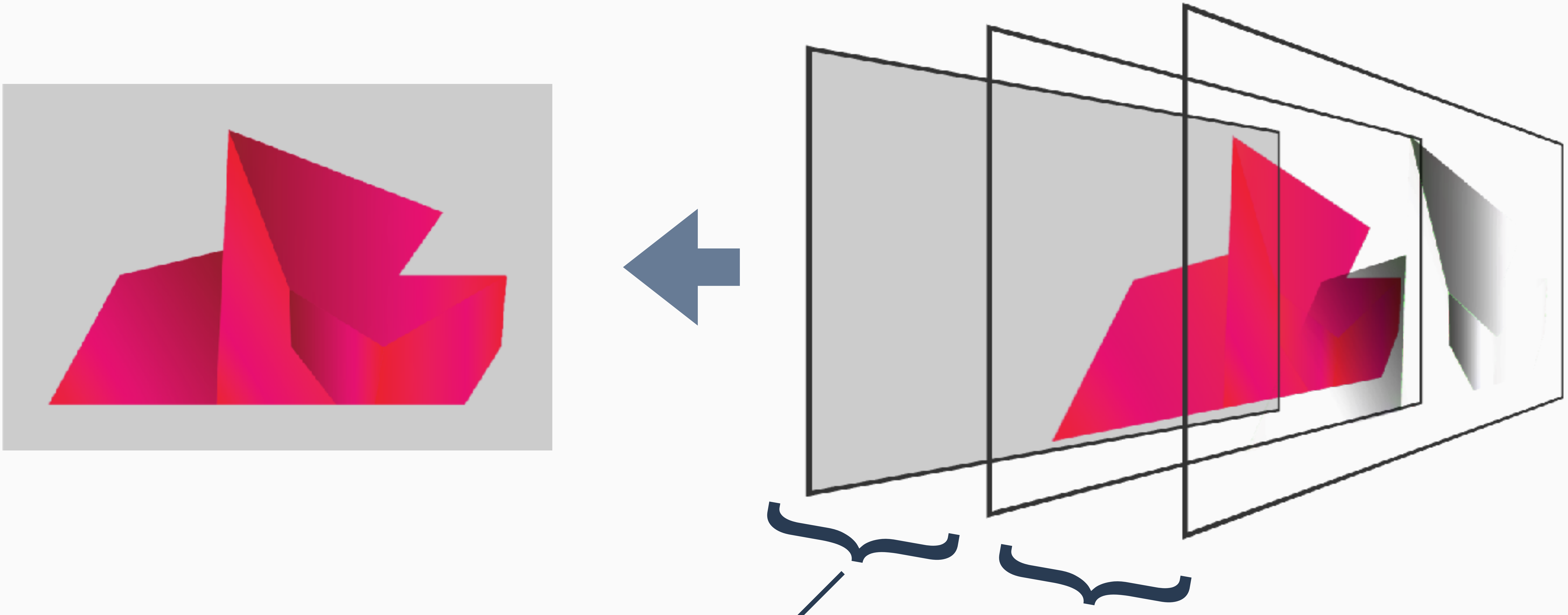
Yuki Koyama & Masataka Goto AIST



Decomposing Images into Layers with Advanced Color Blending

Yuki Koyama & Masataka Goto AIST

Two Main Usages of Layers



For segmenting regions
(not our focus)

For blending colors
(our focus)

Specialization of the Composite Operator

$$f(\mathbf{c}_s, \mathbf{c}_d) = \mathbf{c}_s,$$
$$X = Y = Z = 1$$

$$\mathbf{x}_s \oplus^{\mathrm{rgb}} \mathbf{x}_d = \frac{\alpha_s \mathbf{c}_s + (1 - \alpha_s)\alpha_d \mathbf{c}_d}{\alpha_s \oplus^{\alpha} \alpha_d},$$
$$\alpha_s \oplus^{\alpha} \alpha_d = \alpha_s + (1 - \alpha_s)\alpha_d$$

Alpha blending model

C.f., [Tan+16]

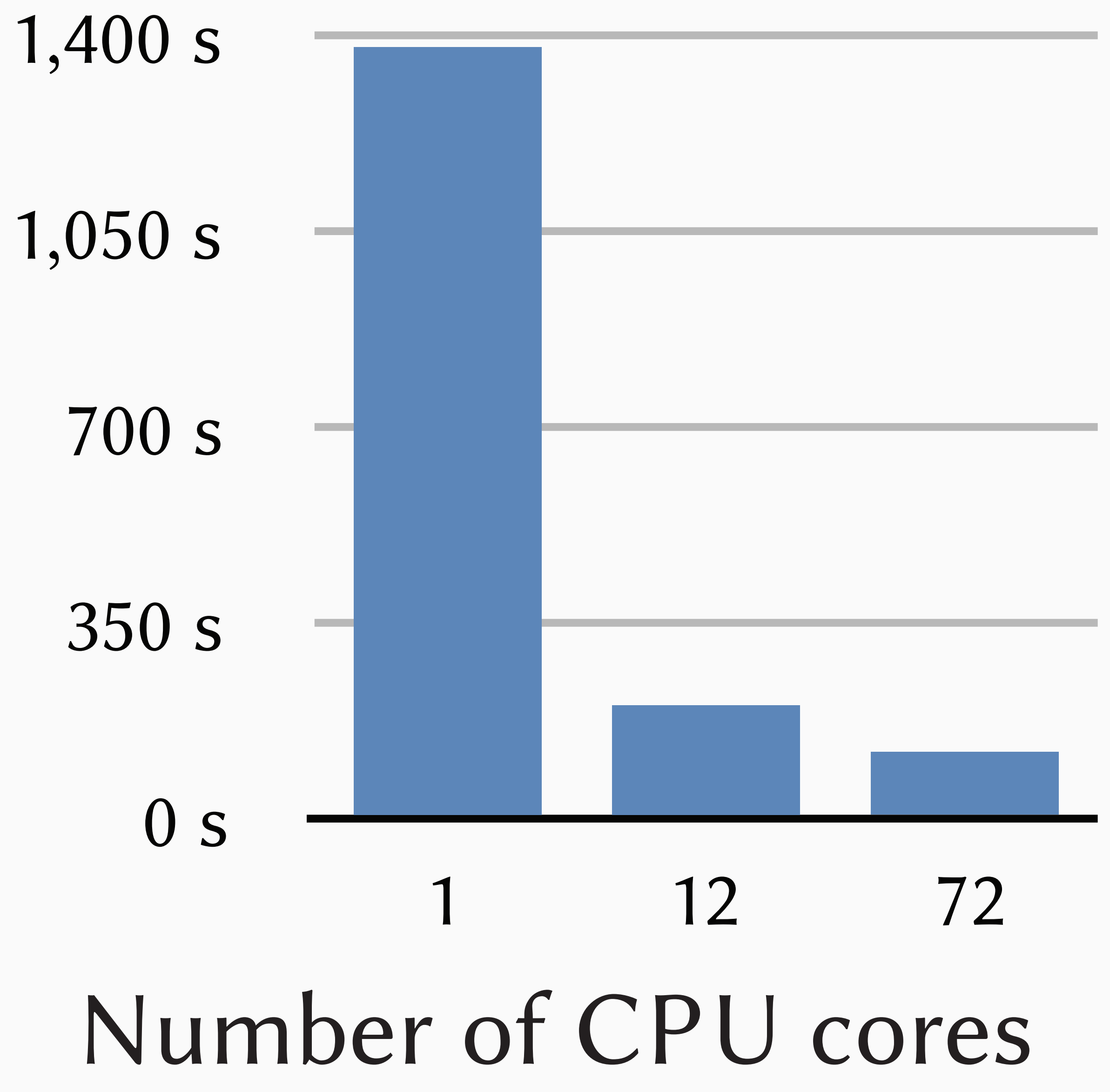
$$f(\mathbf{c}_s, \mathbf{c}_d) = \mathbf{c}_s + \mathbf{c}_d,$$
$$X = 2, Y = Z = 1$$

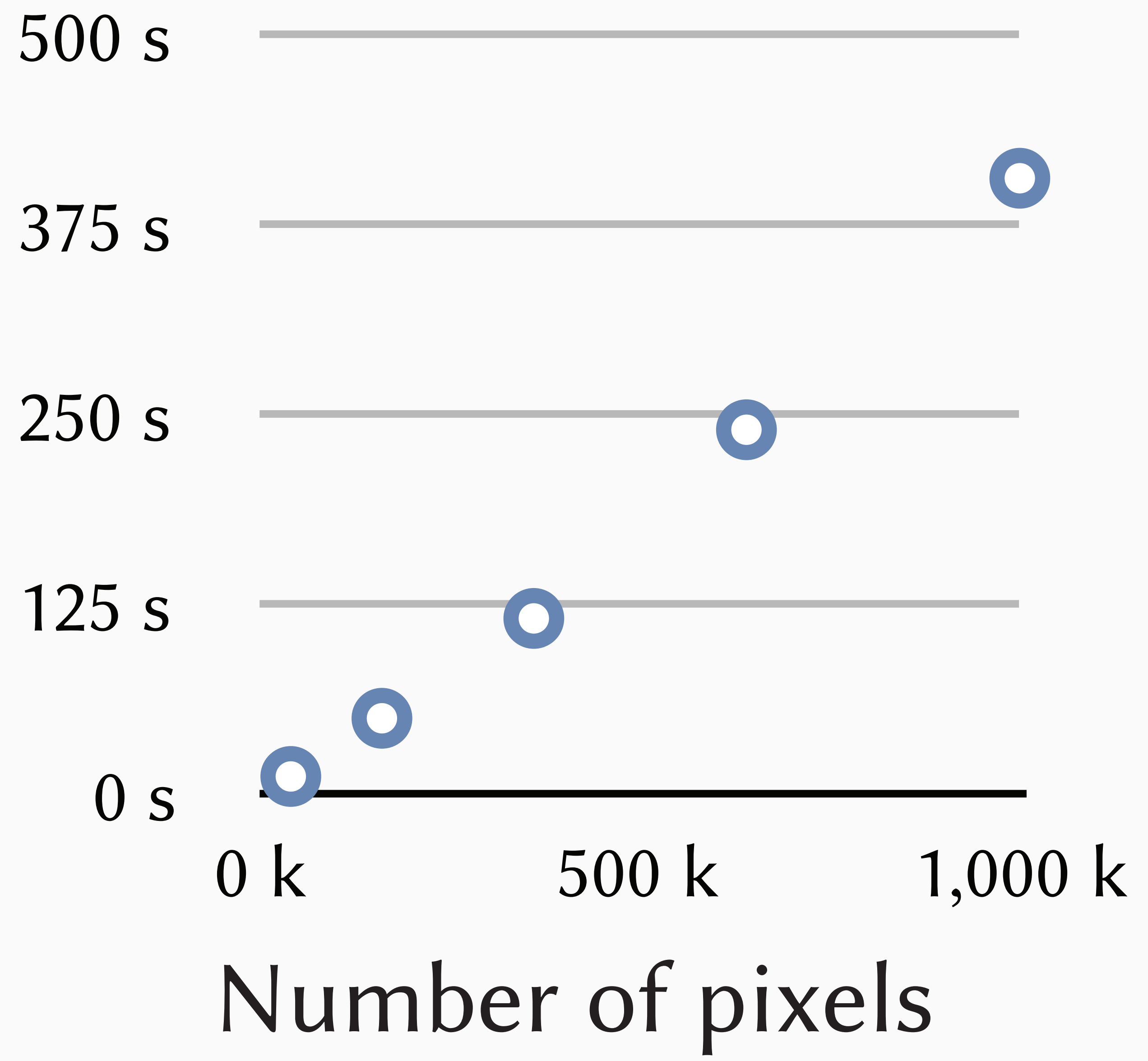
$$\mathbf{x}_s \oplus^{\mathrm{rgb}} \mathbf{x}_d = \frac{\alpha_s \mathbf{c}_s + \alpha_d \mathbf{c}_d}{\alpha_s \oplus^{\alpha} \alpha_d},$$
$$\alpha_s \oplus^{\alpha} \alpha_d = \alpha_s + \alpha_d$$

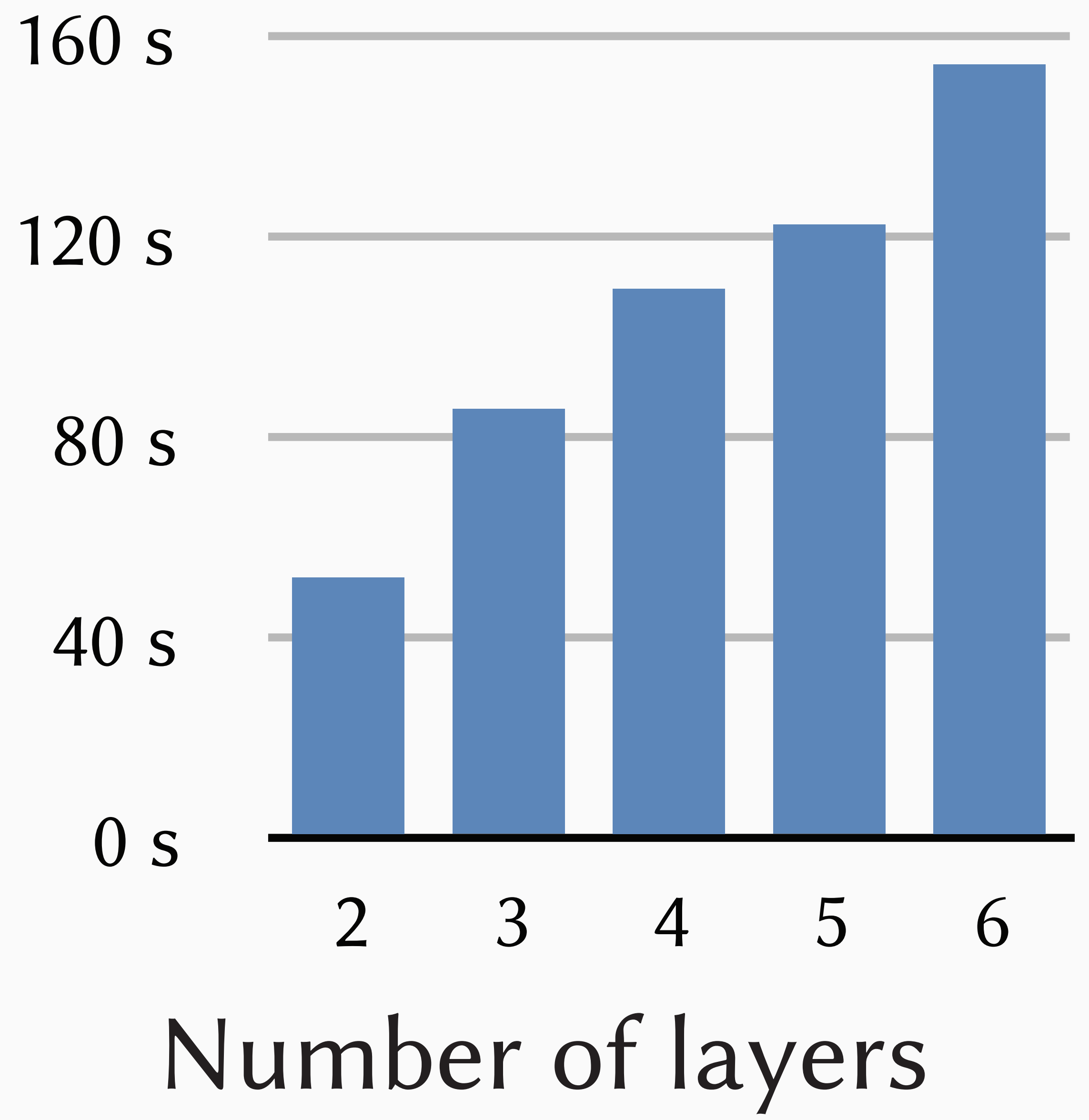
Linear additive model

C.f., [Aksoy+16; 18]

Computational Cost

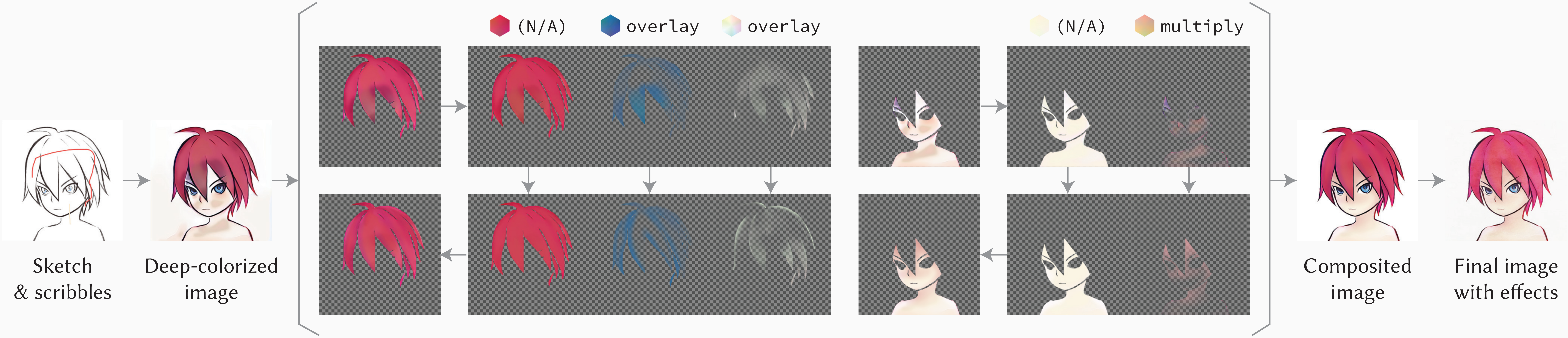






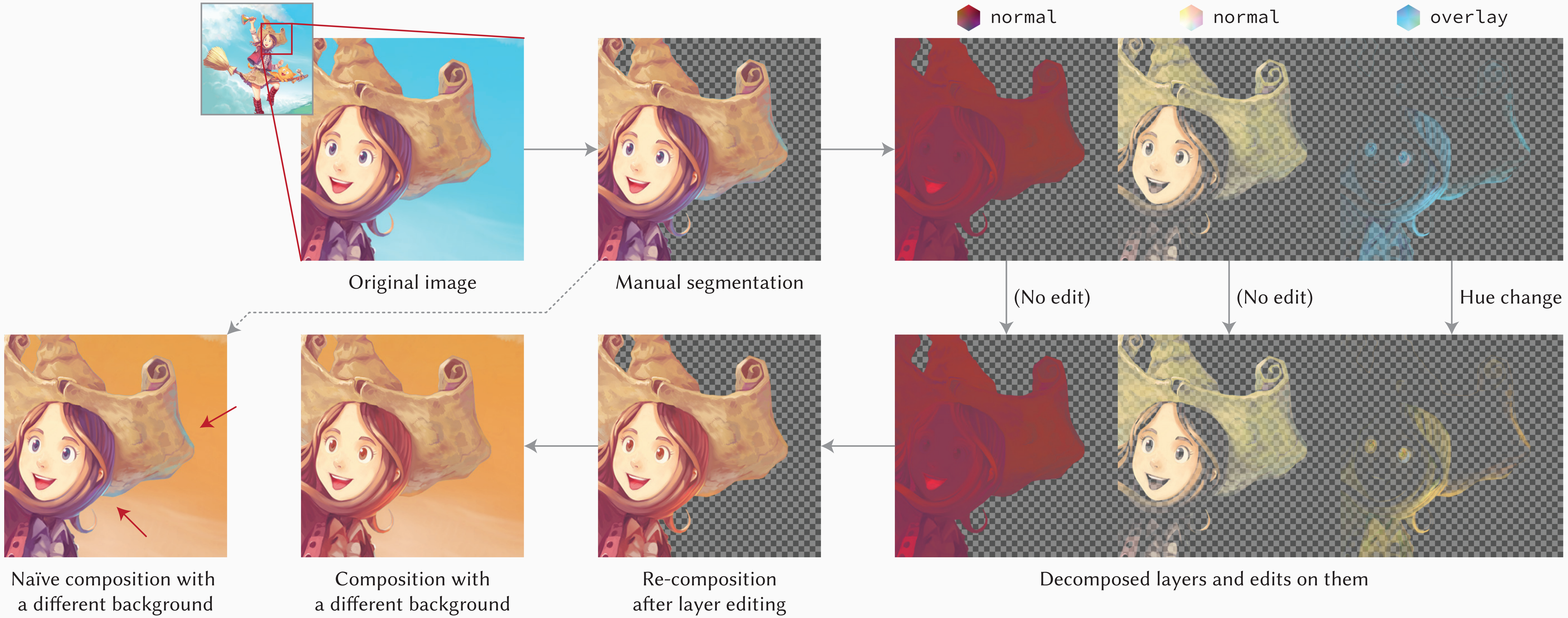
- Parallelization is effective
- Increase cost along with the number of pixels
- Increase cost along with the number of layers (due to the recursive calculation)

Fixing Deep Colorization via Layer Decomposition



Our method enables a new digital painting workflow by effectively using such a deep-colorized image as a good starting point. Instead of directly editing the generated single-layered image, the user can decompose it into layers with familiar advanced color blending then individually edit each unsatisfactory layer by adding strokes, changing hue, etc.

Lighting-Aware Replacement of Background



Example use of our method, in which the blue sky background is replaced with a sunset one. With our method, the character region is decomposed into three layers: the bottom two layers roughly correspond to albedo, and the top layer roughly corresponds to lighting by the blue sky represented as an overlay layer. To match the lighting in sunset, the top layer's hue is modified. Finally, the layers are re-composited with the sunset background. Note that naïvely compositing the character region with the sunset background (the leftmost image) does not look good because lighting is mismatched. Input image courtesy of David Revoy.