ColorfulCurves: Palette-Aware Lightness Control and Color Editing via Sparse Optimization

CHENG-KANG TED CHAO, George Mason University, USA
JASON KLEIN, Cornell University, USA
JIANCHAO TAN, Kuaishou Technology, China
JOSE ECHEVARRIA, Adobe Research, USA
YOTAM GINGOLD, George Mason University, USA

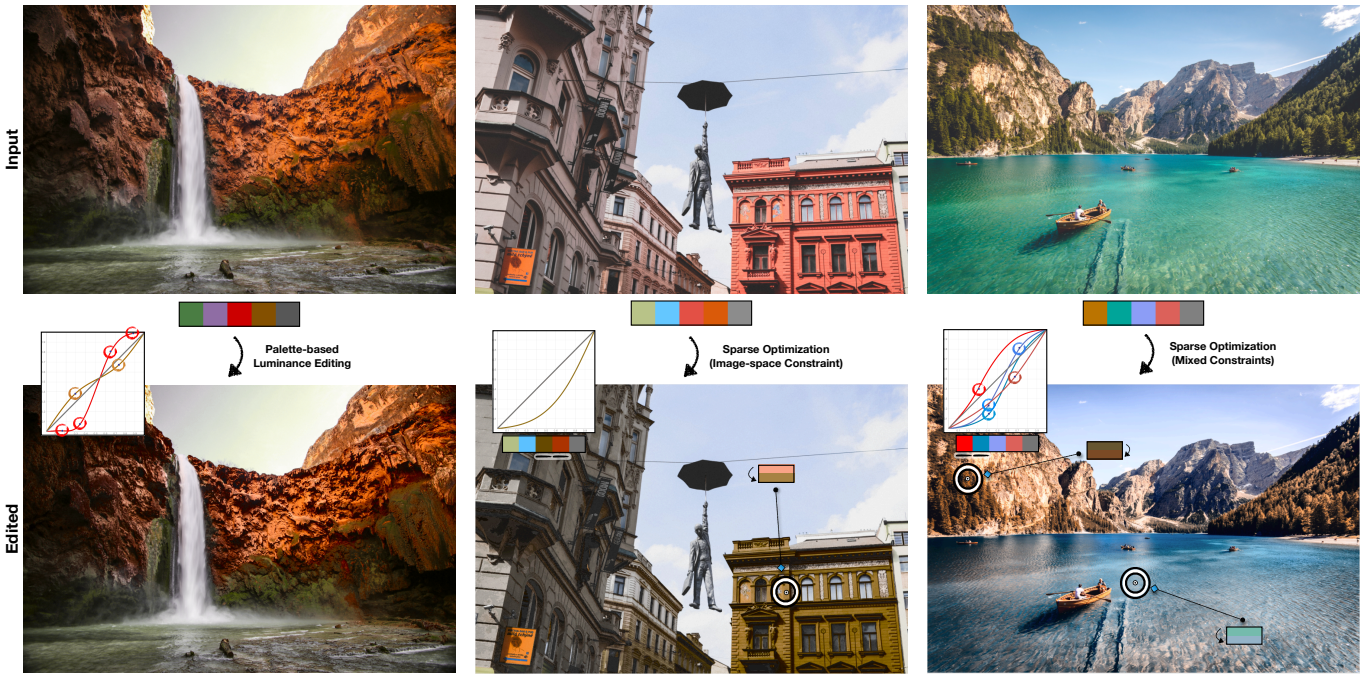

Fig. 1. *ColorfulCurves* extracts a hue-chroma palette and builds palette-based tone curves to allow sparse, per-palette-color control of lightness over the image. Users place constraints on palette colors, tone curves, or directly on image pixels. *ColorfulCurves* optimizes the palette colors and lightness curves to satisfy the user's constraints. Left: The user adds contrast to the rocks with S-shaped curve constraints on the red and brown colors. Center: The user places an image-space constraint on the building to make it dark brown. *ColorfulCurves* optimizes for the sparsest satisfying change to the palette. Right: The user places a mix of image-space, palette, and curve constraints. *Photos courtesy of (left to right) Jeremy Bishop, Nastya Dulhiier, and Pietro De Grandi.*

Color editing in images often consists of two main tasks: changing hue and saturation, and editing lightness or tone curves. State-of-the-art palette-based recoloring approaches entangle these two tasks. A user's only lightness control is changing the lightness of individual palette colors. This is inferior to state-of-the-art commercial software, where lightness editing is based on flexible tone curves that remap lightness. However, tone curves are only provided globally or per color channel (e.g., RGB). They are unrelated to the image content. Neither tone curves nor palette-based approaches support direct image-space edits—changing a *specific* pixel to a desired hue, saturation, and lightness. *ColorfulCurves* solves both of these problems by uniting palette-based and tone curve editing. In *ColorfulCurves*, users directly edit palette colors' hue and saturation, per-palette tone curves, or image pixels (hue, saturation, and lightness). *ColorfulCurves* solves an $L_{2,1}$ optimization problem in real-time to find a sparse edit that satisfies all user constraints. Our expert study found overwhelming support for *ColorfulCurves* over experts' preferred tools.

CCS Concepts: • **Computing methodologies** → **Image processing**.

Additional Key Words and Phrases: palette-based image editing, color, optimization, lightness, tone curves, usability

Authors' addresses: Cheng-Kang Ted Chao, cchao8@gmu.edu, George Mason University, USA; Jason Klein, jak532@cornell.edu, Cornell University, USA; Jianchao Tan, tanjianchaoustc@gmail.com, Kuaishou Technology, China; Jose Echevarria, echevarr@adobe.com, Adobe Research, USA; Yotam Gingold, ygingold@gmu.edu, George Mason University, USA.

0730-0301/2023/8-ARTN
https://doi.org/10.1145/3592405

ACM Reference Format:
Cheng-Kang Ted Chao, Jason Klein, Jianchao Tan, Jose Echevarria, and Yotam Gingold. 2023. *ColorfulCurves*: Palette-Aware Lightness Control and Color Editing via Sparse Optimization. *ACM Trans. Graph.* 42, 4, Article N (August 2023), 12 pages. https://doi.org/10.1145/3592405

1 INTRODUCTION

Color editing is an important task performed by photographers and photo editors. Existing color editing software allow users to alter the hue, saturation (or *chroma*), and lightness of pixels globally or more locally through different combinations of tools. In commercial applications, lightness editing is typically achieved by interactively editing tone curves [Blackmagic Design 2020; Life After Photoshop 2020]. Tone curves are global functions that remap pixels' lightness values. Tone curves are usually parametric, with a variable number of control points for flexibly editing different tonal regions (i.e., black, shadow, mid-tone, highlight, and white regions). Commercial applications allow the curves to be edited through various graphical interfaces, such as directly manipulating spline control points, overdrawing the curve, or adjusting sliders, wheels, or clicking/tapping on the image [Adobe 2022; Color Grading LLC 2022].

To address artists' desire for color editing beyond lightness, commercial tools also offer a separate tone curve for each color channel (i.e., RGB)—still applied globally to all pixels. Remapping red, green, or blue values independently affects the hue and chroma of pixel colors, enabling a wider range of color edits. While some users learn to achieve their intended global color grades through practice or simplified interfaces like wheels [Adobe 2020; Gardiner 2022], many edits are impossible through RGB curves alone. For example, the red tone curve will affect red and yellow pixels. This is due to the *non-sparsity* of RGB color space—or any 2D representation of hue and chroma. The set of pixels unaffected by, e.g., the red channel has measure zero. Users can sometimes overcome non-sparsity by creating masks. However, there are still situations like smooth color transitions where masks are not trivial or impossible to create.

Palette-based recoloring [Chang et al. 2015a] aims to provide a more efficient and intuitive approach to color editing. It does this by extracting a set of representative colors—a color palette—for the image. Edits to the palette colors directly translate to edits on the image, according to the mixing weights for each pixel. Different methods compute palettes and weights with different properties [Aksoy et al. 2017; Chang et al. 2015a; Tan et al. 2018a; Wang et al. 2019], seeking to balance between the number of colors and sparsity of the weights. However, existing palette-based methods have two main limitations: 1) Palette colors don't consider tone, so it's impossible to change a palette color's behavior in shadow, mid-tone, and highlight regions separately. 2) If an object or region's color is not present in the palette, the user must find the palette colors that affect the region and mentally invert the mixing weights.

ColorfulCurves extends palette-based recoloring with tonal controls and pixel-level color constraints. Curve-based edits become sparse and color-aware. To separate tonal controls from the other color dimensions, we extract hue-chroma color palettes and their corresponding pixel weights and assign a tone curve to each color in the palette. *ColorfulCurves* provides a *commutative* interface

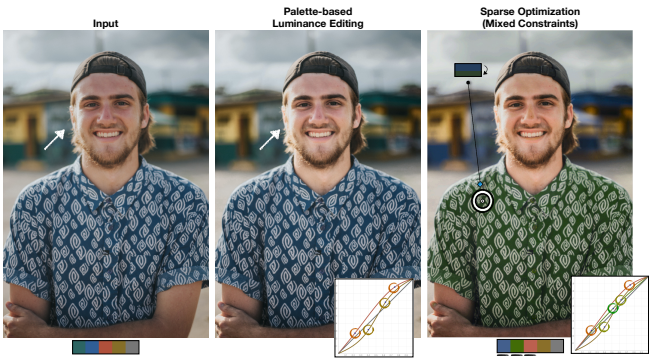

Fig. 2. *ColorfulCurves* solves for a sparse change to the palette and its tone curves that satisfied all user constraints. (Middle) The user places *curve constraints* to edit the lightness of the face without affecting the lightness of the shirt (white arrow). (Right) An additional *image-space* constraint changes the color of the shirt to green. *Photo courtesy of Vince Fleming.*

in which users can place color constraints directly on image pixels, palette colors, or tone curves. *ColorfulCurves* finds a sparse solution—the one that affects the fewest palette colors the least—in real-time by decomposing an $L_{2,1}$ optimization into an alternating sequence of extremely efficient sub-problems. Our expert study shows that *ColorfulCurves* is powerful and preferable to experts' favorite commercial software for color editing tasks (Sec. 6). We show additional applications, e.g., depth weights and greyscale conversion control, in Sec. 5. Code for this work can be found at https://github.com/CraGL/ColorfulCurves.

2 RELATED WORK

Palette-based recoloring. Palette-based recoloring was first proposed by Chang et al. [2015a], who proposed to extract a representative color palette via color clustering and recolor images via radial-basis-function-weighted color deformations. Tan et al. [2016] proposed a geometric approach to palette extraction based on simplified convex hulls in RGB-space and optimization to decompose an image into a translucent layer per color for over-compositing. Tan et al. [2018a,b] extended [Tan et al. 2016] with a simple and efficient approach in RGBXY-space to compute spatially coherent additive mixing weights for image recoloring and color harmonization. We extend the idea of this approach by building tone curves on hue-chroma palette colors, allowing users to achieve flexible, per-palette-color luminance editing. Though users are able to change the luminance of an RGB-palette color in Tan et al. [2018a], their linear formulation is severely limits user control (Fig. 3). Wang et al. [2019] optimized the colors of a geometric palette to be more compact and representative and less sensitive to outliers. Other approaches for RGB-space palette and weight computation are orthogonal to our approach. Zhang et al. [2021a] solved for palette colors and mixing weights simultaneously in a global optimization. Grogan and Smolic [2020] split RGB-space into several regions and use different geometric methods to unmix pixel colors based on their location in RGB-space. Unmixing approaches such as [Aksoy et al. 2016, 2017] minimize an energy to find a small set of sparse layers of nearly,

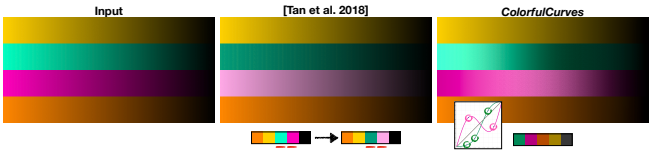

Fig. 3. Palette-based methods like [Tan et al. 2018a] treat pixel colors as mixtures of palette colors in RGB-space, making it impossible to access and edit different lightness of a palette color non-linearly (mint or pink in this example). Our approach disentangles lightness from hue-chroma, allowing users to edit independent lightness curves for the mint and pink palette colors. Additional comparisons are shown in Fig. 13.

though not entirely, homogeneous colors. Koyama and Goto [2018] generalize [Aksoy et al. 2016, 2017]'s work to support advanced color blending functions. Two recent works solve the unmixing problem using neural networks to achieve fast performance [Akimoto et al. 2020; Horita et al. 2022]. While the soft color layers generated by unmixing approaches allow users to localize edits, recoloring is more challenging due to the heterogeneous layer colors they generate.

Luminance/lightness editing and tone mapping. Luminance and lightness editing are common photo manipulation tasks to improve the visual appearance of images by, e.g., adjusting the contrast or altering the highlights and shadows. In the photo manipulation literature, luminance and lightness are often used interchangeably (and erroneously). Several approaches automatically adjust luminance and lightness for specific effects. Bailey and Grimm [2006] enhanced the apparent depth of object images via changing luminance and color temperature to achieve perceptually meaningful edits. Khan et al. [2006] and Boyadzhiev et al. [2015] utilize luminance as their basic appearance property for material editing. More recently, Ma et al. [2022] trained a two-stage network to bring up image lightness and enrich chromaticity to mimic artists' retouching. Tone mapping is an important parametric technique that modulates luminance. Early approaches such as Debevec and Gibson [2002] proposed an operator on high-contrast images that preserves image details and luminance contrast. Durand and Dorsey [2000] proposed a temporally coherent approach suitable for interactive rendering. Mantiuk et al. [2008] proposed to minimize visible contrast distortions for reproducing images on different given displays. Tone mapping algorithms can't be compared to ground truth data from the physical world, since it is ultimately an aesthetic preference. [Bychkovsky et al. 2011] created a dataset of high-quality human-retouched images for automating photo adjustment. A recent adversarial approach [Vinker et al. 2021] was proposed to train on unpaired HDR and LDR images to automate tone mapping. Unlike these automatic approaches, we provide palette-based tone curves as an interface for users to interactively manipulate lightness to achieve their desired edits. Our approach leverages mixing weights as a proxy to control lightness corresponding to hue-chroma palette colors. Our approach can be generalized to use different weighting schemes for other applications (Sec. 5).

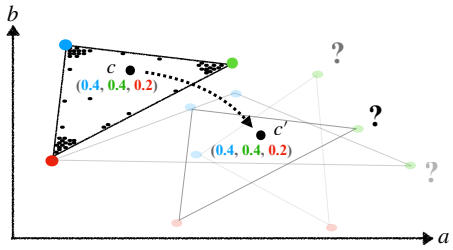

Fig. 4. An illustration of the difficulty achieving a desired image-space color with palette-based image editing. The color c is defined as a fixed mixture of the three colors. To achieve a specific target color c' in palette-based editing, users must adjust the palette colors. This is difficult as there are infinitely many solutions, many of which require adjusting multiple palette colors to stay in the gamut. Users must mentally invert the weights and iteratively adjust the palette until the target color is achieved.

3 BACKGROUND AND MOTIVATION

Background. The geometric palette-based editing formulation computes image colors $I \in \mathbb{R}^{N\times 3}$ by applying per-pixel mixing weights $W \in \mathbb{R}^{N\times p} \subseteq [0,1]$ to the palette P, i.e., $I = W \cdot P$. The rows of P are colors in a gamut, e.g., the unit cube in RGB-space, and the rows of W sum to one. Users simply change the colors in P to globally recolor the image. The mixing weights remain fixed [Tan et al. 2018a, 2016; Wang et al. 2019; Zhang et al. 2021b].

lightness Curves. While this formulation provides easy and fast color editing, it cannot support advanced curve-based lightness edits found in virtually all professional software. It is impossible to control areas with low, mid, or high tones independently because palette-based editing is limited to linear mixtures of colors (Fig. 3). Users can edit the palette, choosing brighter or darker colors, but the resulting image colors will still be linear mixtures of the palette colors (Fig. 13). An image color represented as a $\frac{1}{4} : \frac{3}{4}$ blend of a palette color and black will always remain a $\frac{1}{4} : \frac{3}{4}$ blend. This can't be solved by expanding the palette, because hue, chroma, and lightness are entangled.

Image-Space Constraints. Palette-based edits are convenient when users want to directly edit, e.g., the green color mixed into pixels. They are inconvenient when users want to directly edit a pixel's color. The indirection from the palette to a pixel's color makes precise changes at an image-space location extremely tedious. Users must repeatedly adjust multiple palette colors in order to achieve their desired color change. Formally, given a color $c \in \mathbb{R}^{1\times 3}$ with its corresponding mixing weights $w \in \mathbb{R}^{1\times p}$, we can express $c = w \cdot P$. If a user wishes to change c to a different color c', the user must mentally invert the weights and anticipate the effects of changes to the palette P. In general, there are infinitely many solutions involving multiple palette colors. This is even more difficult when the palette mixture at a given pixel is non-obvious (Fig. 4).

4 METHOD

Our goal is to allow users to edit the palette colors and lightness of a given image in real-time using constraint-driven optimization. In

support of this goal, we seek (I) a linear palette-based editing formulation for efficient optimization and (II) a color-space which disentangles lightness and hue-chroma. Approaches that non-linearly apply palette edits [Chang et al. 2015b] or extract colorful layers [Aksoy et al. 2017; Koyama and Goto 2018] do not satisfy (I). Geometric palettes (Section 3) naturally satisfy (I), though they all consider lightness to be part of the palette definition by extracting palettes in a 3D color space (typically RGB). To satisfy (II), we instead extract 2D palettes in hue-chroma space, namely, the *ab* dimensions of CIE LAB color space. We adapt Tan et al. [2018a]'s approach for palette extraction and weight computation due to its efficiency and simplicity. We extract 2D palettes in *ab*-space and compute weights in *abxy*-space for spatial smoothness. Thus, instead of computing a simplified 3D convex hull as the palette and Delaunay tessellating a 5D convex hull to compute weights, we compute a simplified 2D convex hull as the palette and Delaunay tessellate a 4D convex hull to compute weights. This has lower computational complexity, by a polynomial degree in the case of the Delaunay tessellation. We did not evaluate adapting alternative geometric palette approaches [Wang et al. 2019; Zhang et al. 2021b]; they may have led to more compact palette colors and sparser weights at the cost of an iterative optimization.

Given an image $I \in \mathbb{R}^{N\times 3}$ in Lab-space, the *ab* portion of I, i.e. $I|_{ab} \in \mathbb{R}^{N\times 2}$, can be decomposed as

$$I|_{ab} = W \cdot P \tag{1}$$

where $W \in \mathbb{R}^{N\times p} \subseteq [0,1]$ are spatially coherent weights and $P \in \mathbb{R}^{p\times 2}$ is a palette in *ab*-space with p colors. In RGB-space, extracted palettes typically include an approximately black and white color; Tan et al. [2018a] explicitly tessellate a *line of greys*. This allows direct control over grey and linear control over lightness. However, in *ab*-space, all greys are at $(0,0)$. It is unlikely that a convex hull will have a vertex at the center of the *ab* plane. Therefore, we always add grey to our *ab*-space palette P (inset right) and tessellate a fan around it.

We further denote $L_0 \in \mathbb{R}^{N\times 1}$ as the original lightness of I. We define a set of lightness curve functions $F = \{f_1, f_2, \cdots, f_p\}$, where each $f_i \colon [0,1] \to [0,1]$ corresponds to each palette color P_i, for $i = 1, \cdots, p$. For simplicity, we denote $f_i(L_0)$ as a vector, where each element in $f_i(L_0)$ is the result of mapping each element in L_0 using f_i. The lightness portion I_L of I can then be computed as

$$I|_L = \sum_{i=1}^{p} \widehat{W}_i \odot f_i(L_0) \tag{2}$$

where $\widehat{W}_i$ is the i^{th} column of $\widehat{W}$ and $\odot$ is the Hadamard product. $\widehat{W}_i$ are re-normalized weights, where we scale down the grey weight by 100 and re-normalize W so that pixel lightness will be dominated by the colorful members of the palette. Our formulation generalizes the traditional RGB tone curves to more color-aware *palette-based tone curves* that provide more flexibility and control. This formulation perfectly reconstructs L_0 in the initial image decomposition, since each row of W sums to 1 and f_i are initially the identity map. Users edit the image by changing the *ab*-space palette P and curve functions F. The mixing weights W propagate each palette's lightness function to the pixels.

Directly changing palette hue-chroma in Eq. 1 or palette-based tone curves in Eq. 2 provides an easy and computationally efficient approach for image editing. However, the indirection from a palette and a set of f_i's to a pixel color is non-trivial and presents a usability problem. Users can't directly edit pixel colors. Because the mapping from a desired pixel color to palette and lightness curves is ambiguous, *ColorfulCurves* optimizes for a palette and a set of lightness functions that satisfy the following user constraints placed on:

(1) The given image, i.e. *image-space constraints*, to specify desired colors directly on selected pixels.
(2) The palette, i.e. *palette constraints*, to directly change a palette color or keep it from changing.
(3) Curve functions, i.e. *curve constraints*, to directly manipulate curve functions for lightness edits.

4.1 Curve Desiderata

Users edit lightness by changing the curve functions F in Eq. 2. However, users constrain specific lightness values, not entire curve functions. We have three desirable properties for the f_i's [Margulis 2022]:

(1) Unless explicitly specified by the user, $f_i(0) = 0$ and $f_i(1) = 1$ to maintain the original white and black points in the image.
(2) f_i should be smooth to avoid sharp changes between tones.
(3) f_i should satisfy the user's curve constraints.

Constrained lightness editing, i.e., placing control points directly on curves, is more general than using gamma correction functions since an exponential function can only interpolate three points (e.g., black, white, and a single arbitrary curve constraint). In practice, photo-editing software like Lightroom or DaVinci Resolve create interpolating splines for users' curve constraints. Spline interpolation is straightforward when users are directly manipulating a single curve function, but our curves need to be variational to account for the indirection from a constraint on a pixel's lightness to the lightness curves. This is because pixels are weighted averages of palette colors. An image-space constraint has infinitely many solutions that blend to the desired appearance. Therefore, we wish to choose among them in a principled way by evaluating the loss computed over the solution curve—in our case the sparsest solution (Sec. 4.2).

4.2 Optimizing for Sparse Edits

Our goal is to find a minimum change from the original palette P and identity mapping lightness curves that satisfy *image-space constraints*, *palette constraints*, and *curve constraints*. Since there are infinitely many solutions, we wish to find the solution that changes the fewest number of palette colors the least. Given color palette $P \in \mathbb{R}^{p\times 2}$ and weights $W \in \mathbb{R}^{N\times p}$, let $\widetilde{W} \in \mathbb{R}^{p\times c}$ be a collection of mixing weights from the pixels at *image-space constraints*, where c is the number of constraints. Since we seek global sparsity among all lightness curves and palette colors, we evaluate the lightness curves on their entire domain discretized at s equally spaced sample points for each f_i. We denote $L_i \in \mathbb{R}^s$ as a vector of function

values at sample points for f_i and $B \in \mathbb{R}^{s\times s}$ as a Laplace operator (with mirroring at endpoints for natural boundary conditions). We formulate the sparsity loss E_{sp} on both palette colors and lightness curves as the following $L_{2,1}$ norm:

$$E_{sp} = \sum_{i=1}^{p} \sqrt{L_i^T B^T B L_i + w_{sp} \cdot \|q_i \cdot \Delta P_{i,*}\|_2^2} \tag{3}$$

where w_{sp} is relative weight and $q \in \mathbb{R}^p$ is a penalty vector $(1, ..., 1, 6)$ that increases the cost (by a factor of 6) of moving the grey palette color. [1] To satisfy the desired lightness from the *image-space constraints* (considering only the luminance values) and direct *curve constraints* from direct curve manipulation, we can now define the loss E_l for lightness as

$$E_l = \sum_{j=1}^{c} \|\widetilde{S}_j \odot \left(\sum_{i=1}^{p} \widetilde{W}_{ij} L_i\right) - \widetilde{C}_j\|_2^2 + \sum_{i=1}^{p} \|\overline{S}_i \odot L_i - \overline{C}_i\|_2^2 \tag{4}$$

The first term measures lightness error for image-space constraints. The second term measures lightness curve constraint satisfaction. Here, $\widetilde{S}, \widetilde{C} \in \mathbb{R}^{s\times c}$ are a selection matrix and lightness constraint matrix, respectively, where the j^{th} column, i.e., $\widetilde{S}_j$, is a one-hot vector indicating the location of the lightness constraint, and $\widetilde{C}_j$ is a vector of all zeros except the desired lightness value at the location of the lightness constraint. The same applies *mutatis mutandis* to $\overline{S}, \overline{C} \in \mathbb{R}^{s\times p}$ for placing constraints directly on lightness curves.

For hue-chroma, we allow users to place *palette constraints* on palette colors to perform global edits or to keep them from changing. We define an index set P^c to store which palette colors have *palette constraints*. We compute the loss E_p for the hue-chrome palette as

$$E_p = \sum_{j=1}^{c} \left\| \left(\sum_{i=1}^{p} \widetilde{W}_{ij} (P_{i,*} + \Delta P_{i,*}) \right) - \widehat{C}_{*,j} \right\|_2^2 + \sum_{i \in P^c} \left\| (P_{i,*} + \Delta P_{i,*}) - \widehat{P}_{i,*} \right\|_2^2 \tag{5}$$

where $\widehat{C} \in \mathbb{R}^{2\times c}$ is a matrix of *ab*-space color constraints and $\widehat{P} \in \mathbb{R}^{|P^c|\times 2}$ stores target palette colors. The first term measures hue-chroma error for image-space constraints. The second term measures hue-chroma palette constraint satisfaction. We can solve for the smallest changes to both lightness curves and palette hue-chroma (Eq. 3) satisfying all constraint types by combining Eqs. 4 and 5:

$$\begin{aligned} \{L_i, \Delta P_i\} = \quad & \arg\min_{L_i, \Delta P_i} E_{sp} + w_{eq}(E_l + E_p) \\ \text{subject to} \quad & -128 \le P_i + \Delta P_i \le 127 \\ \text{and} \quad & L_{i,1} = 0,\ L_{i,s} = 1 \end{aligned} \tag{6}$$

where we use $w_{eq} = 1000$ to force the optimizer to satisfy constraints on both lightness and hue-chroma. This constrained optimization problem can be solved using SciPy's SLSQP solver (due to the inequalities). Although this formulation achieves good editing results by considering the sparsity of palette change and lightness together, the performance is far from real-time, even with analytical gradients. For example, if there is a *image-space* and a *palette* constraint, Eq. 6 needs ~26 seconds to converge with $s = 30$ samples per curve. Our real-time approach (Sec. 4.3) takes ~0.08 seconds.

[1] Note that we append grey (the origin in *ab*-space) to our palette. Virtually all colors have non-zero weight with respect to grey, but this solution is much less undesirable. Our goal is to find a new palette that uses as little grey as possible to make the sparsest edits.

4.3 Real-Time Optimization

The above approach, while correct, is too slow for a real-time response when manipulating colors or editing curves. To address this, we describe an approximately ~400× to ~800× faster optimization approach. When optimizing Eq. 6, most of the time is spent solving for lightness since it has $(s \cdot p)$ degrees of freedom.[2] Our acceleration scheme is based on a sequence of alternating optimization steps (block coordinate descent) separating the small number of *ab*-space variables from the lightness curves. We solve each set of variables independently and iteratively, updating the $L_{2,1}$ square root scale factors (Eq. 10) until convergence.

4.3.1 Lightness Optimization. Let the total sparsity loss in Eq. 6 be $E = E_{sp} + w_{eq}(E_l + E_p)$. To solve for luminance, we can take the gradient of E with respect to L_i and set it to zero:

$$\frac{\partial E}{\partial L_i} = \frac{\partial E_{sp}}{\partial L_i} + w_{eq} \cdot \frac{\partial E_l}{\partial L_i} = 0 \tag{7}$$

where

$$\frac{\partial E_{sp}}{\partial L_i} = \frac{B^T B L_i}{\sqrt{L_i^T B^T B L_i + w_{sp} \cdot \|q_i \cdot \Delta P_{i,*}\|_2^2}} \tag{8}$$

and

$$\begin{aligned} \frac{\partial E_l}{\partial L_i} = \sum_{j=1}^{c} 2 \cdot \widetilde{W}_{ij} \cdot (\widetilde{S}_j \odot \left(\sum_{i=1}^{p} \widetilde{W}_{ij} L_i\right) - \widetilde{C}_j) \odot \widetilde{S}_j \\ +2 \cdot (\overline{S}_i \odot L_i - \overline{C}_i) \odot \overline{S}_i \end{aligned} \tag{9}$$

We observe that Eq. 7 would be linear if the denominator in Eq. 8 were treated as a constant. We define scaling factors d_i as

$$d_i = \sqrt{L_i^T B^T B L_i + w_{sp} \cdot \|q_i \cdot \Delta P_{i,*}\|_2^2} \tag{10}$$

and the scaling matrix D as

$$D = \text{diag}\left([1/d_1, \cdots, 1/d_p]^T\right) \in \mathbb{R}^{p\times p} \tag{11}$$

We remind readers that q_i here is the same q_i defined in Eq. 3. To express our entire system in matrix form, we define the lightness matrix $L \in \mathbb{R}^{s\times p}$ whose columns are the L_i vectors of function values at sample points for f_i. Algebraic manipulation of Eq. 7 (see appendix) results in a linear system:

$$Ax_l = b \tag{12}$$

where

$$A = (D \otimes (B^T B)) + w_{eq} \cdot (\widetilde{W} \otimes I_n) \cdot \text{diag}(\text{vec}(2\widetilde{S} \odot \widetilde{S})) \cdot (\widetilde{W}^T \otimes I_n)$$

$$x_l = \text{vec}(L)$$

and

$$b = w_{eq} \cdot (\widetilde{W} \otimes I_n) \cdot \text{vec}(2\widetilde{C} \odot \widetilde{S})$$

Note that 'vec' refers to column-wise vectorization and 'diag' makes a diagonal matrix from a vector. To enforce direct *curve constraints* (i.e., the second gradient term in Eq. 9) in this formulation, we replace rows of A with identity rows at corresponding curve constraints' locations and fill b at the same locations with the corresponding

[2] We use $s = 100$ and $p = 5$ for all our examples.

desired curve constraints. Eq. 12 is a small and sparse linear system easily built and solved in real-time.

4.3.2 Palette Optimization. Consider an *image-space constraint* c_x placed at image location x with desired color $c \in \mathbb{R}^{1\times 2}$. Let $w_x \in \mathbb{R}^{1\times p}$ be the pixel weights obtained from W at x. We wish to find the sparsest change ΔP to the image palette P:

$$\begin{aligned} \min_{\Delta P} \quad & \sum_{i=1}^{p} \sqrt{k_i + w_{sp} \cdot \|q_i \cdot \Delta P_{i,*}\|_2^2} \\ \text{subject to} \quad & w_x \cdot (P + \Delta P) = c_x, \\ & -128 \le P + \Delta P \le 127, \\ \text{and} \quad & (P_{j,*} + \Delta P_{j,*}) = c_P \end{aligned} \tag{13}$$

where k_i denotes $L_i^T B^T B L_i$. Here, the inequality constraint enforces in-gamut colors and the final equality constraint enforces our *palette constraints* (the j^{th} palette color of palette P should equal c_P).

4.3.3 Alternating Optimization. We know from Eq. 12 that D is the bi-Laplacian contribution of different curves. We start the algorithm by initializing D to an identity matrix. Then, we solve Eq. 12 to obtain updated k_i. We solve Eq. 13 with these k_i to find the best sparse palette change. This gives us a more accurate scaling matrix D. We alternate solving these two sub-problems until both k_i and ΔP converge. Our alternating optimization approach is two–three orders of magnitude faster and, in our experience, always converges to an identical, high-quality result. Pseudocode can be found in Algorithm 1.

Algorithm 1: Alternating Optimization for Palette and lightness Sparsity

```
Initialization: D^(0) = I_p, j = 0;
k_i^(1) ← Solve Eq. 12 with D^(0);
ΔP^(1) = Solve Eq. 13 with k_i^(1);
D^(1) ← Update (k_i^(1), ΔP^(1));
while True do
    if ‖ΔP^(j) − ΔP^(j+1)‖_F ≤ j.n.d. and ‖k_i^(j) − k_i^(j+1)‖_F ≤ ϵ
     then
        break;
    else
        j ← j + 1;
        k_i^(j+1) ← Solve Eq. 12 with D^(j);
        ΔP^(j+1) = Solve Eq. 13 with k_i^(j+1);
        D^(j+1) ← Update (k_i^(j+1), ΔP^(j+1));
    end
end
```

5 RESULTS AND APPLICATIONS

A variety of edited results using *ColorfulCurves* can be seen in Figs. 1, 2, 15, 16, as well as results created as part of our expert study's open task (Fig. 14, Section 6). *ColorfulCurves* makes the convenience of palette-based editing available to lightness curves by propagating changes to similar hue-chroma regions. We compare to state-of-the-art palette-based editing [Tan et al. 2018a] in Figs. 3 and 13. Because these approaches combine palette colors linearly, they do not provide enough flexibility to adjust contrast without undesirable side effects. Fig. 6 shows a comparison with Photoshop's Color Range selection, which is a generalization of key-based approaches. With these approaches, the selection is either too small and speckled (even with feathering) or too large. Our evaluation (Sec. 6) describes additional experiments with professionals using commercial tools. *ColorfulCurves* is compatible with using a depth map for lightness editing on image depth (Fig. 5) and is also able to perform controllable interactive grayscale conversion (Fig. 8).

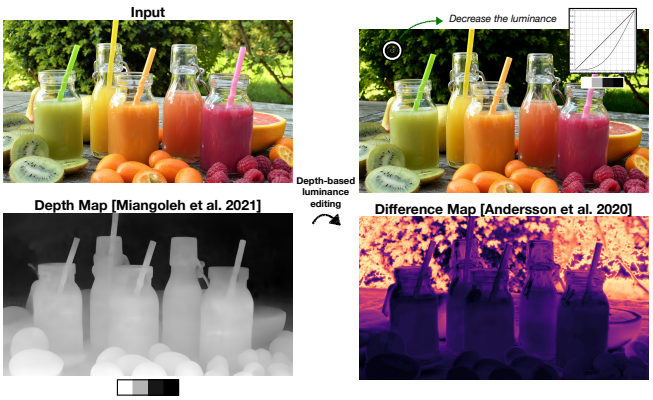

Fig. 5. We create a *depth palette* (farthest, far, near, nearest) from a depth map M [Miangoleh et al. 2021] and compute corresponding spatially coherent weights in MXY-space. The depth palette is shown in grayscale (bottom left). Our approach can create tone curves for each palette depth. *ColorfulCurves* then computes the sparse change in tone curves (Eq. 12) with respect to a constraint placed on the top-left of the input image. The difference map [Andersson et al. 2020] (bottom right) demonstrates the sparsity of the edits with respect to the depth values, mainly affecting regions of similar depth to the user constraint.

Implementation and Performance. We implemented *ColorfulCurves* in Python using NumPy for the linear algebra and SciPy's sequential least squares programming (SLSQP) solver for the optimization, scikit-image for the color conversion, and Qt for the GUI. We also experimented with an interior-point solver, but its performance was worse than SLSQP. We have an optional dependency on PyTorch or OpenCL for parallelizing scikit-image's Lab-to-RGB conversion with a lookup table on the GPU. After conversion to RGB-space, *ColorfulCurves* clips out-of-gamut values to lie within $[0, 1]$ (Fig. 7).

Note that our optimization is independent of image resolution. Our alternating optimization (Sec. 4.3) is real-time and scales linearly with respect to palette size on different numbers and kinds of constraints placed. We compare the performance of directly optimizing Eq. 6 with SLSQP versus our optimization approach (Sec. 4.2) for several scenarios: (1) $s = 40$, two *image-space* constraints and one *palette* constraint: ~89.8 seconds versus ~0.14 seconds (641× faster). (2) $s = 80$, one *image-space* constraints, one *palette* constraint and one *curve* constraint: ~100.12 seconds versus ~0.12 seconds (834× faster).

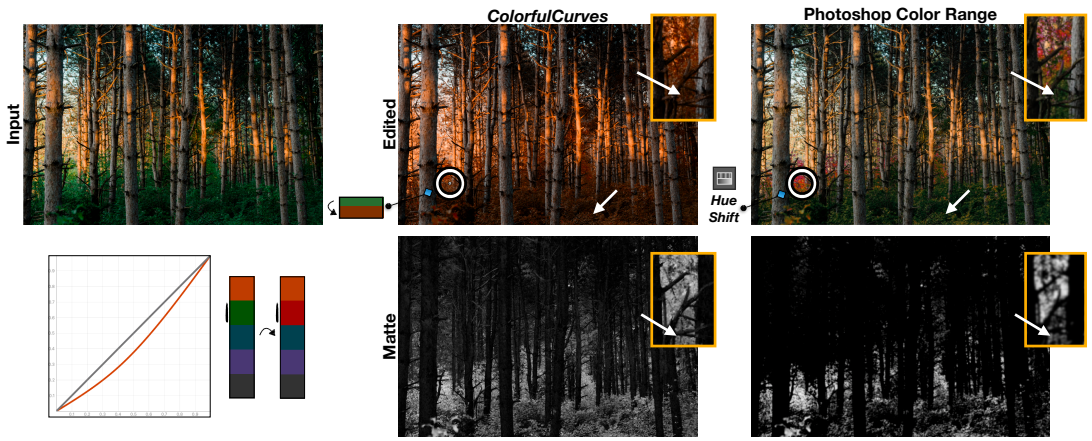

Fig. 6. A comparison with Color Range selection in Photoshop. The user wishes to change the green leaves to fall colors. In *ColorfulCurves*, the user places an image-space color constraint (inside the white circle). The palettes and curves are automatically optimized. The optimized palette and curves can be seen in the lower-left. The weights for the palette color chosen by *ColorfulCurves*'s sparse optimization are shown as its matte. Photoshop's Color Range functionality is applied at the same pixel location, followed by a manual hue/saturation shift to obtain a similar pixel color. Color Range obtains its mask with a fuzziness threshold on the color distance to the selected pixel color. It is difficult to obtain an appropriate matte that is not speckled or too large. Feathering (1-pixel shown) does not overcome this problem. *Photo courtesy of weston m.*

Fig. 7. *ColorfulCurves* operates in Lab-space, whose gamut is larger than RGB. In this example, a single image-space constraint is placed changing the brown shoe to an extreme red. *ColorfulCurves* clips the resulting pixel colors lying outside the RGB gamut. Right: Two views of the clipped colors in RGB space.

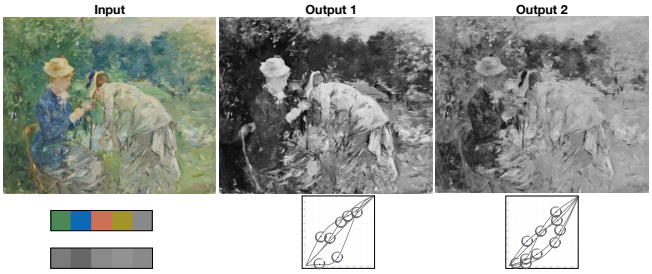

Fig. 8. Using *ColorfulCurves* for converting color images to grayscale. After extracting the hue-chroma palette, we discard the *ab* components of the palette colors and the final image. Since our tone curves are built upon palette colors, users can still interactively control the contrast and exposure separately. *Photo courtesy of Europeana.*

Applications. Our approach is not limited to using a color palette and weights. Depth is also a popular option for edits, e.g., masking the background and editing color and lightness separately from the foreground. For this application, we extract a 1D "depth" palette (farthest, far, near, nearest) from an inferred depth map [Miangoleh et al. 2021]. The depth palette is a 1D convex hull with two more internal vertices computed from K-means. Our approach is easily compatible with this application (Fig. 5). Another application is interactive control for grayscale conversion (Fig. 8). Inspired by [Gooch et al. 2005] and [Margulis 2022], *ColorfulCurves* provides user with control over the resulting contrast in luminosity based on the colors in the input image.

6 EVALUATION

We performed an expert study with professional photographers and photo editors to evaluate *ColorfulCurves*. We recruited ten participants, P1-P10, (ages 23-53, two women, eight men). Two of them were contacted via social networks and the rest were paid $50 via the UpWork freelancer platform. The participants had an average of 10 years of photo editing experience, ranging from three to thirty years. The study (roughly an hour) was performed remotely and asynchronously through our written instructions (see files in the supplemental).

At the beginning of the study, experts followed a tutorial on *ColorfulCurves*. Each expert was asked to complete three tasks. In the first two tasks, experts were given two images, each with an editing intent. Experts completed the same tasks with (1) *ColorfulCurves* **and** (2) their most comfortable editing tool. (Six experts chose Photoshop and four chose Lightroom). Each expert was asked to record the time spent and save all the edited results for each task. The last task asked experts to edit an image of their own using *ColorfulCurves*. At the end of the study, all participants filled out a questionnaire. This included eight unpaired Likert-scale questions (Q1-Q8).

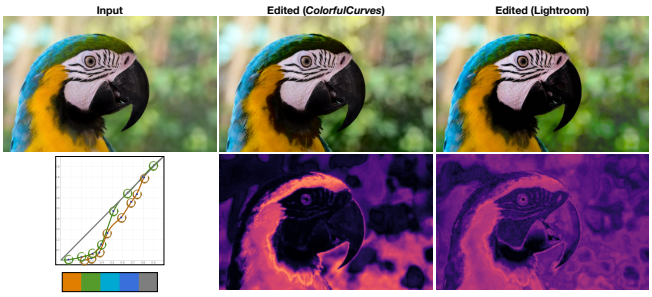

Fig. 9. Visual comparison of the sparsity of the edits performed by P8 using Lightroom and *ColorfulCurves*. The task required to make the parrot's forehead and the yellow feathers darker using just tone curves. Due to the apparent subtlety of the edits, difference images were computed using FLIP [Andersson et al. 2020]. With our method, P8 was able to focus more clearly on the regions of interest, affecting slightly some areas in the background with colors very similar to the parrot's forehead, but leaving the rest mostly unchanged. With Lightroom, the user was not able to push the edits as much as with *ColorfulCurves*, given edits were already affecting most of the image (background, parrot's beak). The bottom left inset shows the image palette and curve edits by P8. *Photo courtesy of David Clode.*

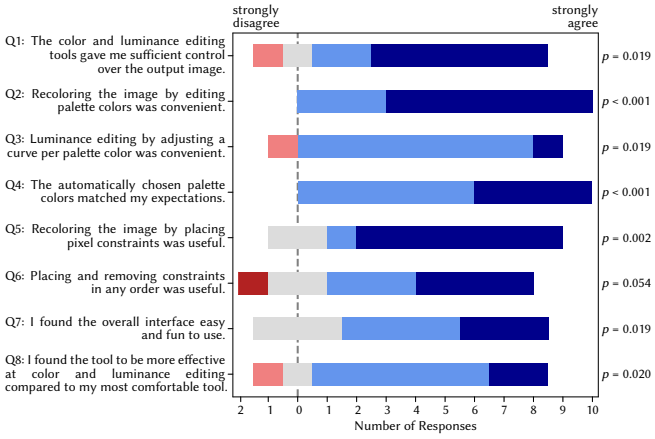

Fig. 10. Survey results from our expert study. p-values shown have been adjusted with a Holm-Bonferroni family-wise error rate correction. Legend: Light and dark blue refer to agree and strongly agree, respectively. Light and dark red refer to disagree and strongly disagree, respectively. Grey refers to neutral.

A comparison between results obtained with *ColorfulCurves* and commercial software can be seen in Fig. 9. A gallery of edited results from experts' own images can be found in Fig. 14. We performed statistical analysis on Q1-Q8 (Fig. 10). We compute satistical significance via a one-sample non-parametric permutation t-test against the theoretical median response of 'neutral'. We corrected for the family-wise error rate among the eight questions with a Holm-Bonferroni correction. All questions obtained statistical significance with (corrected) $p < 0.05$ except for Q6.

To investigate the effect of tool preference, we also examined answers for experts of Adobe Lightroom (N=4) and Adobe Photoshop

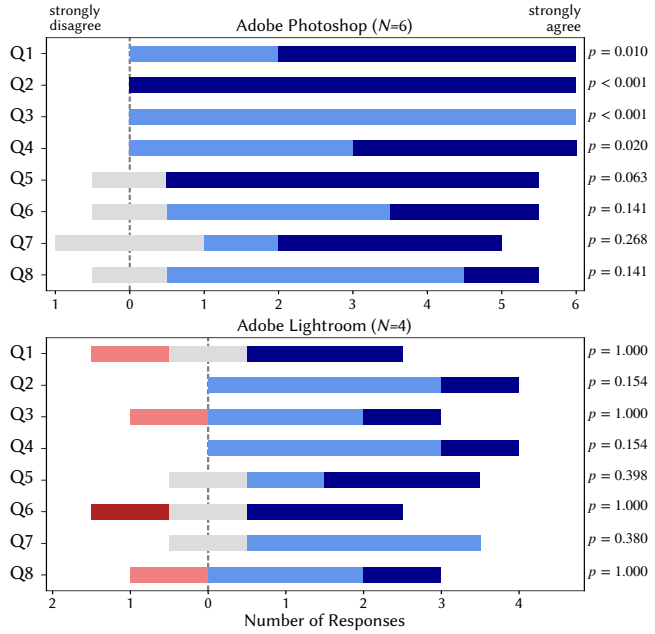

Fig. 11. Survey results from experts using Adobe Photoshop (N=6) and Adobe Lightroom (N=4). p-values shown have been adjusted with a Holm-Bonferroni family-wise error rate correction. The scarcity of experts in the respective sub-groups renders the sample size inadequate to obtain statistical significance for most answers. Legend: Light and dark blue refer to agree and strongly agree, respectively. Light and dark red refer to disagree and strongly disagree, respectively. Grey refers to neutral.

(N=6) separately (Fig. 11). Note that our initial experimental protocol did not set out to study these groups separately, as we did not know which tools experts would choose. We corrected for the family-wise error rate among all 24 questions with a Holm-Bonferroni correction. Since the study populations of these sub-groups are quite small, the statistical power is low. Nevertheless, responses to Q1–Q4 among Photoshop users obtained statistical significance with (corrected) $p < 0.05$. We also note that all of the negative sentiment in our experiment came from the same expert (P4), who was one of the four Lightroom users.

Seven of the ten experts strongly agreed that (Q5) recoloring the image by placing pixel constraints was useful. They noted that, e.g., "[T]he color changing tool is amazing. I was blown away with the accuracy of the selection tool and how clean it is. I also couldn't go to the blue tones with Lightroom like *ColorfulCurves*.", "[I] have a very good user experience for color adjustments. Compared with only using curves to adjust color tones, visual adjustments allow me to find the direction I want to adjust more quickly." and "[T]he color shifting was super easy and flawlessly happened. The final outcome was as expected. Really love it." One expert commented on *ColorfulCurves* for portrait editing, "[T]o push the challenge further, I choose a portrait to do a skin tone editing and the results are really good."

Eight of the ten experts agreed or strongly agreed that (Q8) "I found the tool to be more effective at color and luminance editing

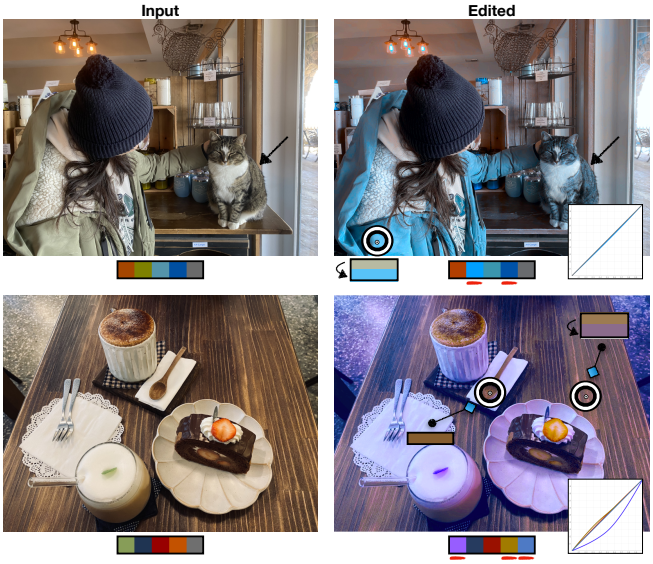

Fig. 12. Limitations of *ColorfulCurves*. First row: When different objects have similar colors, they may share a palette color. Changing the color on one object will apply similar color changes to the other objects (e.g. the woman's jacket and cat). Second row: If two different image-space constraints are placed on pixels with similar input colors, *ColorfulCurves* is unable to find a palette satisfying both constraints simultaneously without drastic, undesirable changes elsewhere (e.g. the spoon and table).

compared to my most comfortable tool." The experts appreciated the speed and sparsity of color and luminance editing, commenting, e.g., "[I]t was impossible for me to achieve same results using curves adjustments in photoshop, had to use several curves and selective masking. Using *ColorfulCurves* was easy and quick.", "[I]f we talk about *ColorfulCurves* and especially curves, it helps quickly change the color. In Photoshop we can't do so clear color in curves because it worked only on shadows and highlights. So, we need to use others tools in PS, such as hue to change the color. So, curves in *ColorfulCurves* work better with colors than curves in PS.", "*ColorfulCurves* is intuitive and easy to adjust the color of a specific area, which cannot be achieved with lightroom.", and "[V]ery easy to adjust for regional color, brightness."

Overall, *ColorfulCurves* was favored by our experts. Nine of the ten experts agreed or strongly agreed that (Q3) our palette-based tone curves were convenient for luminance editing. Eight of the ten users found *ColorfulCurves* more effective than their most comfortable tool. The only question for which we did not achieve statistical significance regarded the commutative nature of our interface (Q6: "Placing and removing constraints in any order was useful."). We speculate that this aspect was not meaningfully different than non-destructive editing experts are already familiar with such as adjustment layers.

7 CONCLUSION AND FUTURE WORK

We presented a new approach for color and lightness editing, unifying tone curves with palette-based editing. Our approach gives photo editors a more flexible (tone curves) and usable (by solving the indirection from an image to the palette and curves) approach to hue, chroma, and lightness manipulation in images. Our approach creates sparser edits compared to state-of-the-art palette-based approaches and commercial software. Most experts agreed that *ColorfulCurves* is more effective at color and lightness editing compared to Photoshop and Lightroom.

Limitations and Future Work. ColorfulCurves may be unable to recolor specific image regions when there is a semantic but not color difference (Fig. 12). Extending *ColorfulCurves* to use multiple palettes on different semantic regions or semantic weights may mitigate this problem. In the future, we would also like to explore text-guided editing ([Avrahami et al. 2022; Bar-Tal et al. 2022]). We would also like to extend our work to video domain. Our 2D palettes are simpler to track than previous work on palette-based video editing [Du et al. 2021].

ACKNOWLEDGEMENTS

We thank Yu-Lin Hsu for statistical advice and the anonymous reviewers for their constructive feedback and suggestions. Yotam Gingold and Cheng-Kang Ted Chao were supported in part by a gift from Adobe Inc.

REFERENCES

Adobe. 2020. What is Color Grading in Lightroom? https://blog.adobe.com/en/publish/2020/10/20/introducing-color-grading. [Online; accessed 24-January-2023].
Adobe. 2022. Adjust the colors in your photos on your phone or tablet. https://helpx.adobe.com/lightroom-cc/how-to/color-adjustment.html. [Online; accessed 24-January-2023].
Naofumi Akimoto, Huachun Zhu, Yanghua Jin, and Yoshimitsu Aoki. 2020. Fast Soft Color Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8277–8286.
Yağiz Aksoy, Tunç Ozan Aydin, Marc Pollefeys, and Aljoša Smolić. 2016. Interactive high-quality green-screen keying via color unmixing. *ACM Transactions on Graphics (TOG)* 35, 5 (2016), 152.
Yağiz Aksoy, Tunç Ozan Aydin, Aljoša Smolić, and Marc Pollefeys. 2017. Unmixing-based soft color segmentation for image manipulation. *ACM Transactions on Graphics (TOG)* 36, 2 (2017), 19.
Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D Fairchild. 2020. FLIP: A Difference Evaluator for Alternating Images. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 2 (2020), 15–1.
Omri Avrahami, Dani Lischinski, and Ohad Fried. 2022. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18208–18218.
Reynold J Bailey and Cindy Grimm. 2006. Perceptually meaningful image editing: Depth. (2006).
Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. 2022. Text2live: Text-driven layered image and video editing. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XV*. Springer, 707–723.
Blackmagic Design. 2020. DaVinci Resolve 18. https://www.blackmagicdesign.com/products/davinciresolve/color. [Online; accessed 24-January-2023].
Ivaylo Boyadzhiev, Kavita Bala, Sylvain Paris, and Edward Adelson. 2015. Band-sifting decomposition for image-based material editing. *ACM Transactions on Graphics (TOG)* 34, 5 (2015), 1–16.
Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. 2011. Learning photographic global tonal adjustment with a database of input/output image pairs. In *CVPR 2011*. IEEE, 97–104.
Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. 2015a. Palette-based Photo Recoloring. *ACM Trans. Graph.* 34, 4 (July 2015).
Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. 2015b. Palette-based photo recoloring. *ACM Trans. Graph.* 34, 4 (2015), 139–1.
Color Grading LLC. 2022. Cinema Grade. https://www.cinemagrade.com/. [Online; accessed 24-January-2023].
Paul Debevec and Simon Gibson. 2002. A tone mapping algorithm for high contrast images. In *13th eurographics workshop on rendering: Pisa, Italy. Citeseer.*

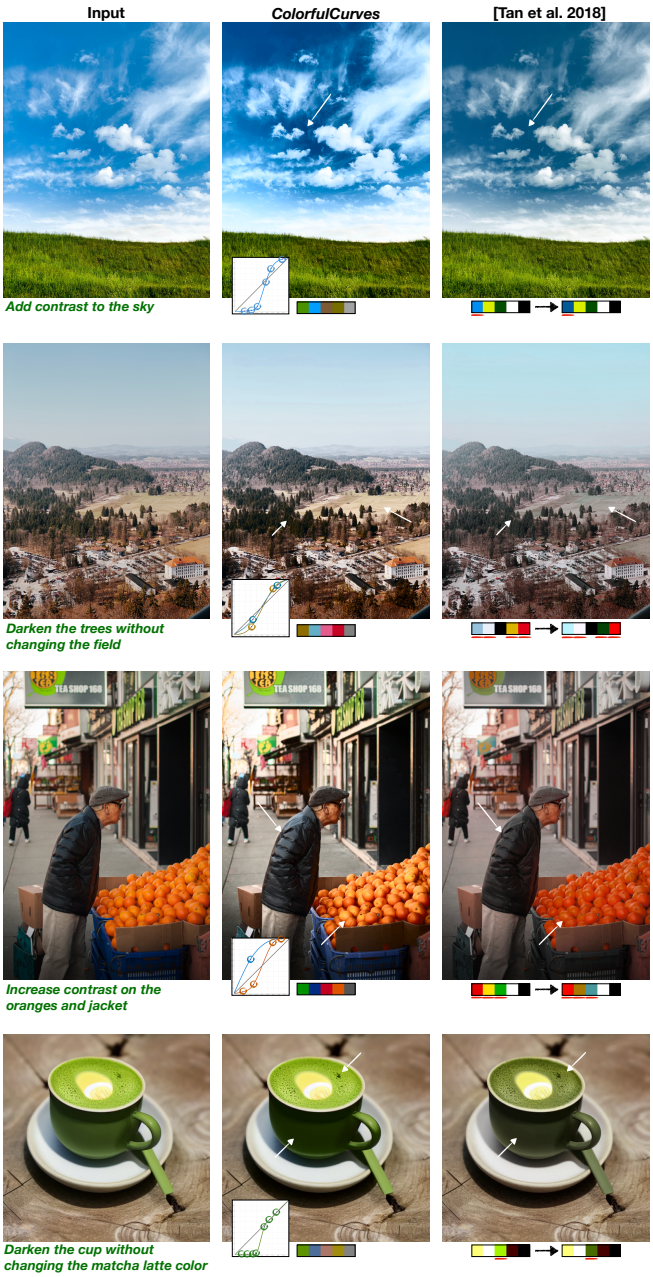

Fig. 13. Comparison to Tan et al. [2018a]. Editing intents are shown in green above each input image. Our approach enables flexible lightness editing via tone curves, allowing photographers to easily add contrast. In Tan et al. [2018a], contrast is difficult to achieve, since images are linear combinations of palette colors. Multiple colors must be edited, with undesirable side effects. *Photos courtesy of Omar Ram, Open Photo, and Bannon Morrissy.*

Fig. 14. Experts (P1, P3, P6, P8, P10 from top to bottom) from our expert study used *ColorfulCurves* to edit images in an open task. Image and palette constraints are not shown. *Photos courtesy (top to bottom) of Mariano Garcia, Cheng-Ju Ko, Eric Wang, Mina Nabil, and Jaan AlBalushi.*

Zheng-Jun Du, Kai-Xiang Lei, Kun Xu, Jianchao Tan, and Yotam Gingold. 2021. Video Recoloring via Spatial-Temporal Geometric Palettes. *ACM Transactions on Graphics (TOG)* 40, 4 (Aug. 2021).

Fredo Durand and Julie Dorsey. 2000. Interactive tone mapping. In *Eurographics Workshop on Rendering Techniques*. Springer, 219–230.

Marie Gardiner. 2022. How to Use the Primaries Color Wheels in DaVinci Resolve. https://photography.tutsplus.com/tutorials/primaries-color-wheels-resolve--cms-41775. [Online; accessed 24-January-2023].

Amy A Gooch, Sven C Olsen, Jack Tumblin, and Bruce Gooch. 2005. Color2gray: salience-preserving color removal. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 634–639.

Mairéad Grogan and Aljosa Smolic. 2020. Image Decomposition Using Geometric Region Colour Unmixing. In *European Conference on Visual Media Production* (Virtual Event, United Kingdom) *(CVMP '20)*. Association for Computing Machinery, New York, NY, USA, Article 2, 10 pages. https://doi.org/10.1145/3429341.3429354

Daichi Horita, Kiyoharu Aizawa, Ryohei Suzuki, Taizan Yonetsuji, and Huachun Zhu. 2022. Fast Nonlinear Image Unblending. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2051–2059.

Erum Arif Khan, Erik Reinhard, Roland W Fleming, and Heinrich H Bülthoff. 2006. Image-based material editing. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 654–663.

Yuki Koyama and Masataka Goto. 2018. Decomposing images into layers with advanced color blending. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 397–407.

Life After Photoshop. 2020. Lightroom tone curves explained: Tone Curve vs Point Curve vs Target Curve adjustments. https://www.lifeafterphotoshop.com/lightroom-tone-curves-explained-tone-curve-vs-point-curve-vs-target-curve/. [Online; accessed 24-January-2023].

Hailong Ma, Sibo Feng, Xi Xiao, Chenyu Dong, and Xingyue Cheng. 2022. Cascade Luminance and Chrominance for Image Retouching: More Like Artist. *arXiv preprint arXiv:2205.15999* (2022).

Rafał Mantiuk, Scott Daly, and Louis Kerofsky. 2008. Display adaptive tone mapping. In *ACM SIGGRAPH 2008 papers*. 1–10.

Dan Margulis. 2022. Wiki (Dan Margulis). (2022). https://en.wikipedia.org/wiki/Dan_Margulis#cite_note-1

S Mahdi H Miangoleh, Sebastian Dille, Long Mai, Sylvain Paris, and Yagiz Aksoy. 2021. Boosting monocular depth estimation models to high-resolution via content-adaptive multi-resolution merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9685–9694.

Jianchao Tan, Jose Echevarria, and Yotam Gingold. 2018a. Efficient palette-based decomposition and recoloring of images via RGBXY-space geometry. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–10.

Jianchao Tan, Jose Echevarria, and Yotam Gingold. 2018b. Palette-based image decomposition, harmonization, and color transfer. *arXiv preprint arXiv:1804.01225* (2018).

Jianchao Tan, Jyh Ming Lien, and Yotam Gingold. 2016. Decomposing Images into Layers via RGB-Space Geometry. *ACM Transactions on Graphics* 36, 1 (2016), 1–14.

Yael Vinker, Inbar Huberman-Spiegelglas, and Raanan Fattal. 2021. Unpaired learning for high dynamic range image tone mapping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14657–14666.

Yili Wang, Yifan Liu, and Kun Xu. 2019. An Improved Geometric Approach for Palette-based Image Decomposition and Recoloring. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 11–22.

Qing Zhang, Yongwei Nie, Lei Zhu, Chunxia Xiao, and Wei-Shi Zheng. 2021a. A Blind Color Separation Model for Faithful Palette-based Image Recoloring. *IEEE Transactions on Multimedia* (2021), 1–1. https://doi.org/10.1109/TMM.2021.3067463

Q. Zhang, Y. Nie, L. Zhu, C. Xiao, and W.-S. Zheng. 2021b. A Blind Color Separation Model for Faithful Palette-based Image Recoloring. *IEEE Transactions on Multimedia* (2021), 1–1. https://doi.org/10.1109/TMM.2021.3067463 Conference Name: IEEE Transactions on Multimedia.

A ALGEBRAIC DERIVATION

We define $L \in \mathbb{R}^{s\times p}$, where each column of L is $L_i \in \mathbb{R}^s$, i.e. function values at sample points for f_i. Note that the second term of Eq. 9 is the direct lightness curve constraints. In other words, the second term forces specific entries of L_i to be specific values. Therefore, we do not need to consider it when deriving the linear system since we can replace the rows of the left-hand side in the final linear system with identity rows at corresponding lightness curve constraints' locations. Therefore, to express Eq. 9 in matrix form, we can write

$$\frac{\partial E_l}{\partial L} = (2H \odot \widetilde{S})\widetilde{W}^T \tag{14}$$

where $H = \widetilde{S} \odot (L\widetilde{W}) - \widetilde{C}$. In addition, if we treat the denominator in Eq. 8 as a constant, we obtain

$$\frac{\partial E_{sp}}{\partial L} = (B^T B)LD \tag{15}$$

where D is a diagonal scaling matrix (Eq. 11). By substituting these terms into Eq. 7 and solving for zero gradient, we end up with the linear system:

$$(B^T B)LD = -w_{eq} \cdot (2H \odot \widetilde{S})\widetilde{W}^T \tag{16}$$

Note that we need to simplify Eq. 16 since the right-hand side involves L. We start by substituting for H and factoring out L:

$$\begin{aligned}
(B^T B)LD &= -w_{eq} \cdot (2(\widetilde{S} \odot (L\widetilde{W}) - \widetilde{C}) \odot \widetilde{S})\widetilde{W}^T \\
&= -w_{eq} \cdot ((2\widetilde{S} \odot (L\widetilde{W}) - 2\widetilde{C}) \odot \widetilde{S})\widetilde{W}^T \\
&= -w_{eq} \cdot (2\widetilde{S} \odot (L\widetilde{W}) \odot \widetilde{S} - 2\widetilde{C} \odot \widetilde{S})\widetilde{W}^T \\
&= -w_{eq} \cdot (2\widetilde{S} \odot \widetilde{S} \odot (L\widetilde{W}) - 2\widetilde{C} \odot \widetilde{S})\widetilde{W}^T \\
&= -w_{eq} \cdot (2\widetilde{S} \odot \widetilde{S} \odot (L\widetilde{W}))\widetilde{W}^T + w_{eq} \cdot (2\widetilde{C} \odot \widetilde{S})\widetilde{W}^T
\end{aligned}$$

We move all terms involving L to the left-hand side:

$$(B^T B)LD + w_{eq} \cdot (2\widetilde{S} \odot \widetilde{S} \odot (L\widetilde{W}))\widetilde{W}^T = w_{eq} \cdot (2\widetilde{C} \odot \widetilde{S})\widetilde{W}^T$$

We vectorize both sides of the above linear equation:

$$\begin{aligned}
\mathrm{vec}((B^T B)LD) + w_{eq} \cdot \mathrm{vec}((2\widetilde{S} \odot \widetilde{S} \odot (L\widetilde{W}))\widetilde{W}^T) \\
= w_{eq} \cdot \mathrm{vec}((2\widetilde{C} \odot \widetilde{S})\widetilde{W}^T)
\end{aligned} \tag{17}$$

Notice that D is a diagonal matrix. The first term of the left-hand side in Eq. 17 is

$$\mathrm{vec}((B^T B)LD) = (D \otimes (B^T B)) \cdot \mathrm{vec}(L)$$

The right-hand side of Eq. 17 is

$$w_{eq} \cdot \mathrm{vec}((2\widetilde{C} \odot \widetilde{S})\widetilde{W}^T) = w_{eq} \cdot (\widetilde{W} \otimes I_n) \cdot \mathrm{vec}(2\widetilde{C} \odot \widetilde{S})$$

The second term of the left-hand side in Eq. 17 is trickier. We need to expand the Hadamard products to factor out L. Dropping the w_{eq} for simplicity,

$$\begin{aligned}
&\mathrm{vec}((2\widetilde{S} \odot \widetilde{S} \odot (L\widetilde{W}))\widetilde{W}^T) \\
&= (\widetilde{W} \otimes I_n) \cdot \mathrm{vec}(2\widetilde{S} \odot \widetilde{S} \odot (L\widetilde{W})) \\
&= (\widetilde{W} \otimes I_n) \cdot [\mathrm{vec}(2\widetilde{S} \odot \widetilde{S}) \odot \mathrm{vec}(L\widetilde{W})] \\
&= (\widetilde{W} \otimes I_n) \cdot [\mathrm{Diag}(\mathrm{vec}(2\widetilde{S} \odot \widetilde{S})) \cdot \mathrm{vec}(L\widetilde{W})] \\
&= (\widetilde{W} \otimes I_n) \cdot \mathrm{Diag}(\mathrm{vec}(2\widetilde{S} \odot \widetilde{S})) \cdot (\widetilde{W}^T \otimes I_n) \cdot \mathrm{vec}(L)
\end{aligned}$$

Finally, we combine all the equations above. Our solution is obtained by solving the linear system $Ax_l = b$ where

$$A = (D \otimes (B^T B)) + w_{eq} \cdot (\widetilde{W} \otimes I_n) \cdot \mathrm{Diag}(\mathrm{vec}(2\widetilde{S} \odot \widetilde{S})) \cdot (\widetilde{W}^T \otimes I_n)$$

$$x_l = \mathrm{vec}(L)$$

and

$$b = w_{eq} \cdot (\widetilde{W} \otimes I_n) \cdot \mathrm{vec}(2\widetilde{C} \odot \widetilde{S})$$

□

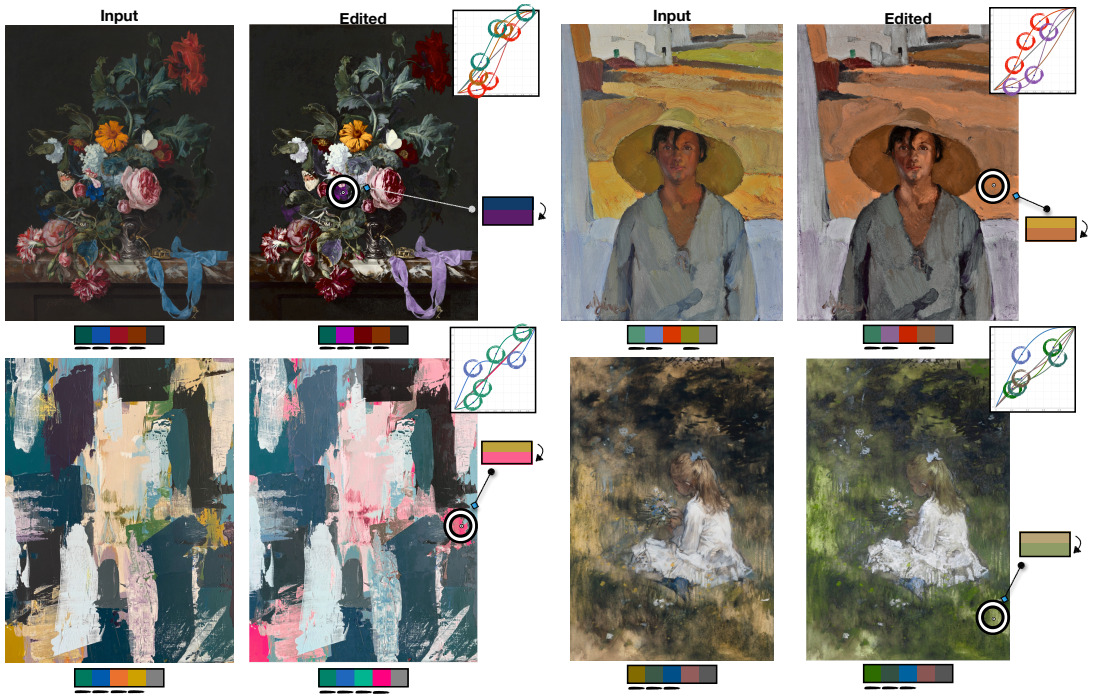

Fig. 15. *ColorfulCurves* can also be used to edit different art styles, including realism, impressionism, and oil painting. *Photo courtesy (from top to bottom) of Europeana, Europeana, Vojtech Bruzek, and Europeana.*

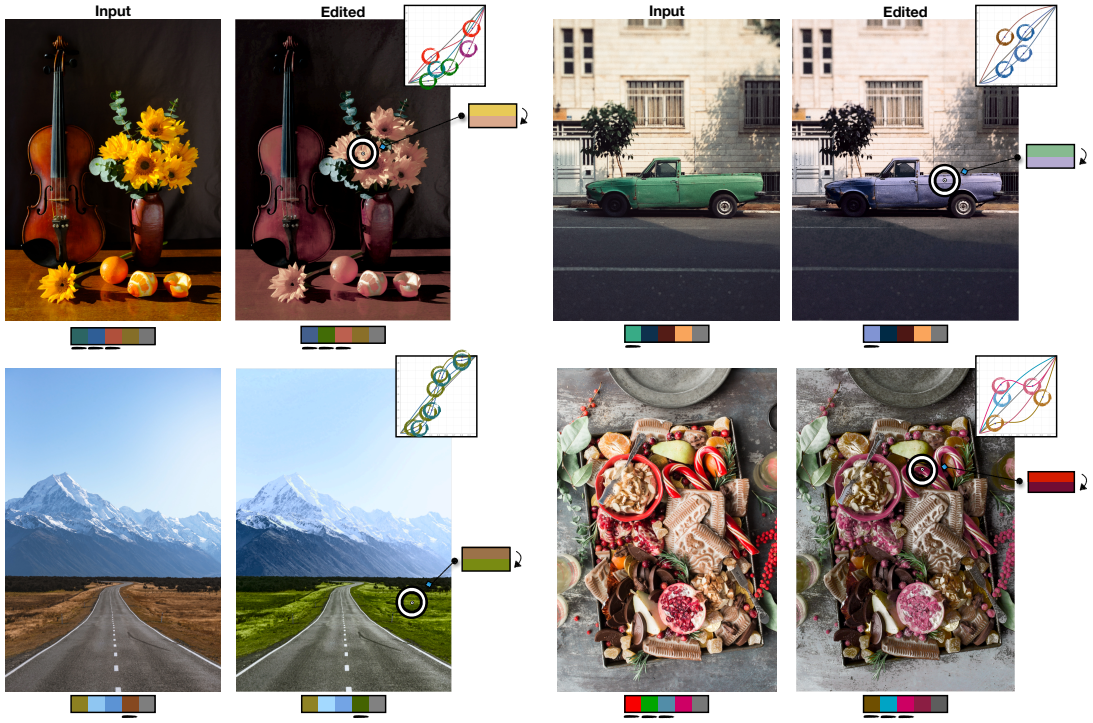

Fig. 16. *ColorfulCurves* can edit a variety of different landscapes, still life, objects, and food. *Photo courtesy (from top to bottom) of Garreth Paul, Casey Horner, Morash, and Brooke Lark.*