Eurographics Symposium on Rendering 2021
A. Bousseau and M. McGuire
(Guest Editors)

Volume 40 (2021), Number 4

PosterChild: Blend-Aware Artistic Posterization

Cheng-Kang Ted Chao[1], Karan Singh[2], Yotam Gingold[1]

[1]George Mason University, USA
[2]University of Toronto, Canada

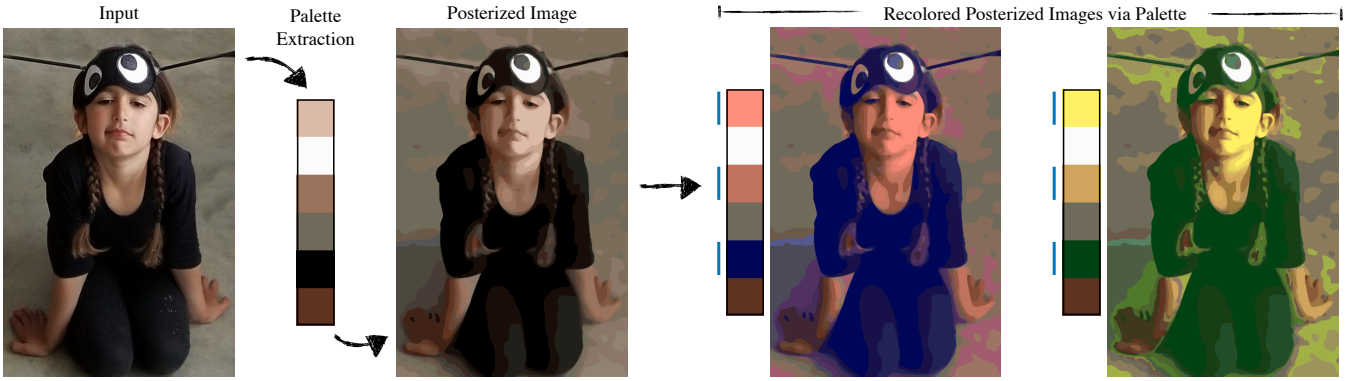

Figure 1: *Artistic Posterizations generated by our algorithm. Our algorithm captures accurate colors and preserves high-frequency details in the original scene, while creating smooth layer boundaries elsewhere. Our posterized layers are defined as blends of an automatically extracted representative palette. This allows artists to create characteristic recoloring effects in real-time by manipulating palette colors after pre-processing by our posterization pipeline.*

Abstract

Posterization is an artistic effect which converts continuous images into regions of constant color with smooth boundaries, often with an artistically recolored palette. Artistic posterization is extremely time-consuming and tedious. We introduce a blend-aware algorithm for generating posterized images with palette-based control for artistic recoloring. Our algorithm automatically extracts a palette and then uses multi-label optimization to find blended-color regions in terms of that palette. We smooth boundaries away from image details with frequency-guided median filtering. We evaluate our algorithm with a comparative user study and showcase its ability to produce compelling posterizations of a variety of inputs. Our parameters provide artistic control and enable cohesive, real-time recoloring after posterization pre-processing.

CCS Concepts

• ***Computing methodologies*** → ***Non-photorealistic rendering; Image processing;*** • ***Applied computing*** → *Fine arts;*

1. Introduction

Posterization is an effect in which an image with continuous colors such as a photograph is converted into an image consisting of regions of constant color. Posterization is needed when preparing a photograph for printing with a fabrication process capable of reproducing a fixed number of colors in contiguous regions, such as screen printing or Batik, or for vectorization. In some variants of posterization, colors may blend together—mimicking stippled paint deposition—but the regions stay discrete—mimicking an opaque covering that must be manually removed. Posterization has become an intrinsically desirable aesthetic style (e.g. A Scanner Darkly). Tutorials created by graphic artists show significant effort is spent tracing large numbers of color layers [Eva,Rat17,Gra10] or manipulating automatically filtered regions [Hea20], to create posterized images. These tutorials highlight the importance of color blending in producing pleasing posterized gradients, motivating this unique feature of our algorithm. Posterization is also used in commercial [Meh16] and educational [Gri20] settings. High-quality artistic posterization is time consuming and tedious. Ex-

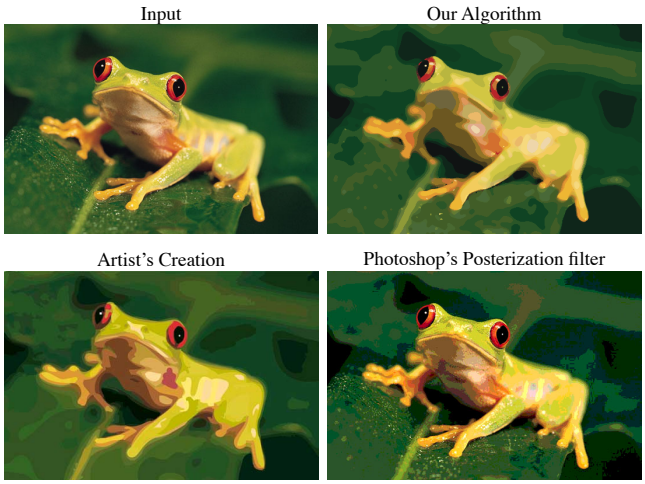

Figure 2: *Our algorithm takes the input (top left) and generates a detailed-preserving posterization (top right) with smooth boundaries and colors matching with the input image. Our posterized output is similar to the manual posterization created by an artist (bottom left, Copyright ©2009 Cascadia Graphics & Publishing). The artist's posterization manually traced region boundaries and applied different Saturation techniques in Adobe Photoshop and Illustrator [Gra10]. Our algorithm takes a minute to generate the posterization. Photoshop's Posterization filter (bottom right) generates undesirable artifacts like noise along color boundaries; its colors do not resemble the input's when a small number of colors are desired.*

isting automatic posterization algorithms produce output quite different from artists, e.g. posterization in Photoshop (see Figure 2).

Our primary **contribution** is a blend-aware algorithm for automatic posterization (Figure 1). Our algorithm is based on principled steps carefully designed to capture artistic design requirements. For example, the smoothness of bending regions is a desirable control for artists over the overall aesthetics of the posters. When posterizing an image, artists often change the colors for aesthetic purposes. The chosen colors may deviate substantially from any in the input image. Our algorithm explicitly takes these considerations into account. Our posterized output is defined in terms of blends of a small color palette (Figure 3). Our algorithm extracts a small yet representative color palette, groups pixels into rough regions whose colors are blends of palette colors, then refines blends and region boundaries in a manner that preserves high frequency detail from the input image while generating smooth boundaries. A secondary **contribution** is a simple technique to improve the robustness and relevance of simplified convex hull palettes. Rather than optimizing the output palette, we remove outliers from the input by K-means clustering with large K (Figure 7). We evaluate our tool with a user study of professional artists.

Figure 3: *We create posterized images (top right) using blended colors from an automatically chosen palette, computing per-pixel additive mixing weights for the palette. Users can edit the palette (bottom row), to re-color the posterized image by re-mixing the per-pixel weighted palette in real-time.*

2. Related Work

2.1. Image Abstraction

Non-photorealistic rendering (NPR) is a longstanding topic in computer graphics concerned with artistic depictions or transformations. See Kyprianidis et al. [KCWI13] for a survey. Our work is most closely related to work on image abstraction. Gerstner et al. [GDA*13] introduced an iterative process for pixelated image abstraction which mimics the "pixel art" artistic style. As part of this work, they show one posterization result via an intermediate stage of their algorithm. We compare to this output in Figure 4. Xu and Kaplan [XK08] proposed a graph-cut technique to achieve black-and-white-only discrete-tone images. While not proposed as techniques for image abstraction, clustering methods [M*67, VS08] and filtering [Afi18] can be used to create a *posterized* look. Photoshop's Posterization filter and Illustrator's Live Trace are interactive tools that generate color effects and vectorized images resembling artistic posterization (Figure 5). Photoshop's Posterization filter produces noisy region boundaries (Figure 2). Live Trace outputs a large number of opaque regions that are difficult to recolor (see Figure 5). Iseringhausen et al. [IWHH20] introduced an algorithm for approximating an image with a grid of edge-aligned, laser-cut wood patches. Machine learning techniques, such as neural networks for style transfer [JYF*20], have received a lot of attention in recent years. One could adopt these network architectures as a posterized image translator, but we know of no large dataset for training. None of these methods simultaneously consider the large, smooth boundaries and representative, discrete colors needed for artistic posterization. We compare our technique to many of these methods in Section 4.1.

2.2. Palette Extraction

Color palettes play an important role in image abstraction and recoloring. One of the primary challenges is extracting a small but useful set of colors, often corresponding to the most prominent

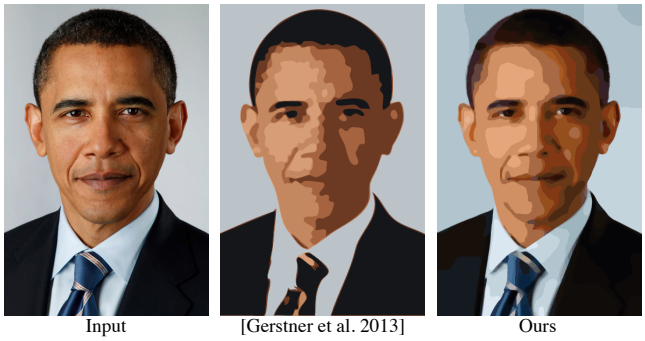

Figure 4: *Our algorithm generates posterized images with better matching colors (the tie) and smoother region boundaries than Gerstner et al. [GDA*13]. Our algorithm better preserves details for the eyes, nostrils, and collar.*

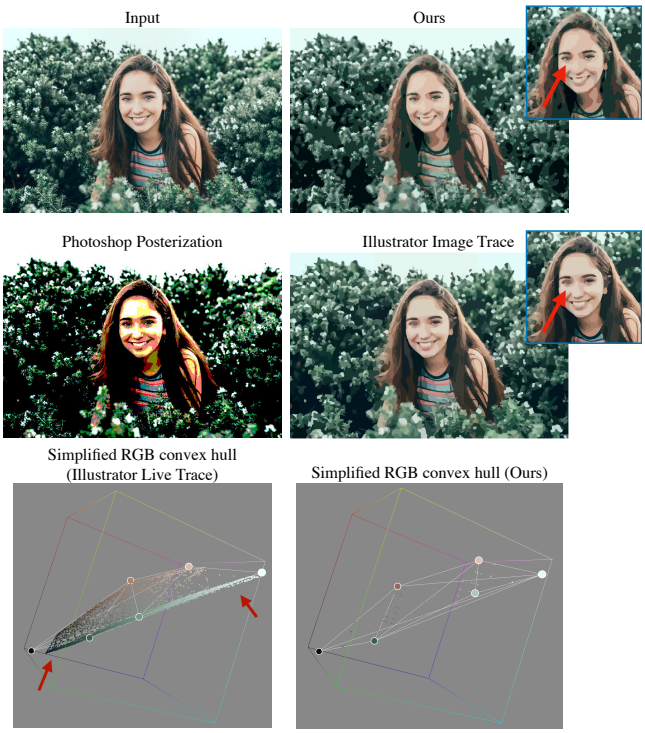

Figure 5: *Posterizations with approximately 20* apparent *colors created with our algorithm, Adobe Photoshop's Posterize filter, and Adobe Illustrator Live Trace. Photoshop's posterization creates a very speckly 20-color image with rough region boundaries and unrepresentative colors. Illustrator Live Trace fails to preserve details in the eyes and is tedious to recolor. In color space we see that thousands of colors were created, which would be difficult to edit by hand. Palette-based recoloring approaches struggle on the Live Trace output. The palette extracted by Tan et al.'s [TLG16] method with the same number of colors as ours (6) does not cover the image's colors. Our algorithm preserves details in the eyes and represents its 21 colors in terms of a representative palette of only 6 colors, facilitating recoloring. Image ©Christian Ferrer*

or relevant colors according to human visual perception. Chang et al. [CFL*15a] and Phan et al. [PFC18] proposed clustering-based methods to find palettes in LAB-space. Rang et al. [RPCB17] developed a modified 2D clustering algorithm based on the *ab* channels of LAB-space for extracting themes and recoloring images. Shugrina et al. [SKFS20] presented a minimal nonlinear primitive to model images with 2-manifold color distributions. These approaches create palettes with custom (i.e., non-linear) blending or editing approaches. We integrate linear blending (additive mixing) directly into our algorithm pipeline. As a result, we prefer palettes designed for linear blending, such as those based on convex geometry in color-space. Tan et al. [TLG16, TEG18] proposed to simplify the convex hull of all color samples in RGB-space to obtain a polyhedron whose vertices better represent the image's color gamut. Wang et al. [WLX19] introduced an optimization procedure to improve the compactness and relevance of the polyhedron's vertices. We, too, adopt simplified convex hull palettes, but propose to improve the vertex quality by preprocessing the input. These color palettes trivially support additive mixing, which our algorithm makes use of. Aksoy et al. [AASP17] proposed an approach based on additive mixing colorful layers. This approach has perfect reconstruction accuracy, but color editing is complex since each layer contains a distribution of colors as opposed to a single palette color. Crowd-sourced data has been mined to develop models of color theme preference [OAH11, LH13]. Such an approach could be used to generate suggested recolorings for our approach.

3. Method

Our algorithm takes an arbitrary color image I as input. The output of our algorithm is a *posterized* image P partitioned into regions with smooth boundaries where possible while preserving important details. Region colors are a close visual match to original image. Region colors are represented as linear blends between (at most two) colors in a compact, representative palette. Recoloring is easily accomplished by adjusting the palette colors themselves and updating the apparent region colors by remixing with the same blending weights, as in palette-based image recoloring approaches [CFL*15b, TLG16, WLX19].

3.1. Overview

We decompose algorithmic posterization into palette extraction, rough region and color assignment, and refinement (improving the accuracy of color blends and smoothness of region boundaries). This pipeline is illustrated in Figure 6. We extract palette colors using a simplified convex hull. To make palette extraction robust to outliers, we apply K-means clustering on the input image before extracting its palette. Given a set of palette colors and their pairwise blends, we define and solve a multi-label optimization problem on the input image. The pairwise blends ensure that the many shades of output regions can be edited via a small palette. Multi-label optimization is limited to a discrete set of color blends. We refine each output region's blending weights to better match the input image. We reconstruct a color image using the blending weights and the possibly-recolored palette. To produce smooth region boundaries in the output while preserving details, we post-process the image with a high-pass filter. Low frequency regions in the input image

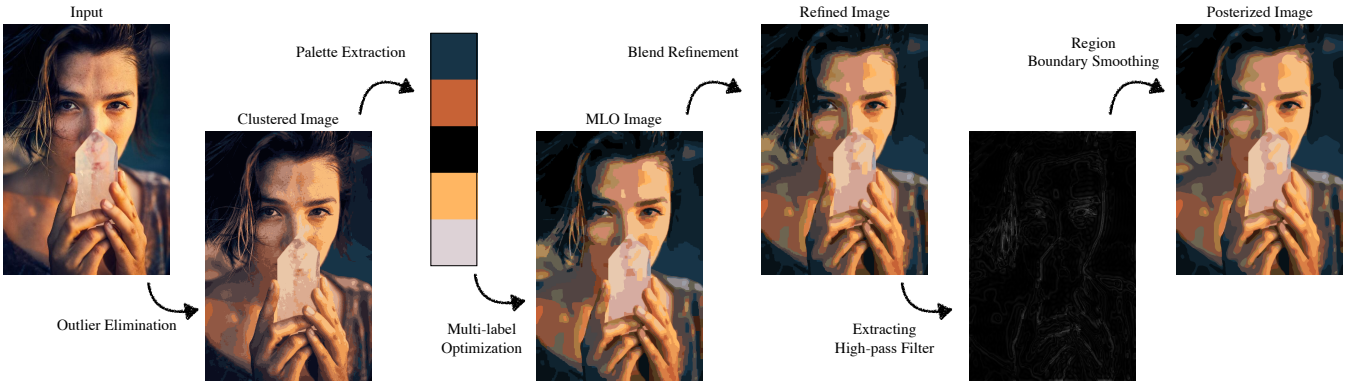

Figure 6: *The pipeline for our automatic posterization algorithm. ©Darius Bashar*

are smoothed with a median filter, which preserves the piecewise constant nature of region colors.

The user-facing parameters in our algorithm are: rare color suppression (the number of clusters K in K-means clustering to remove outlier colors), palette size $p > 3$, the recolored palette, palette blends (pair-wise discrete blending steps d), detail abstraction (threshold t to truncate the low-frequency areas when smoothing region boundaries), and, finally, region clumpiness (λ which balances between the data and pairwise terms in our multi-label optimization). We provide a GUI which exposes these settings and sliders for recoloring (Figure 16).

3.2. Palette Extraction

Discrete tone regions are an essential property of posterization. Our goal is to create a posterized image using a discrete set of colors that represent the input image and provide convenient handles for recoloring. By considering blends of palette colors, we can achieve a larger number of discrete colors with a very compact, editable palette. This is important for creating gradations of tones from continuous regions such as skin, the sky, and sunsets. Convex-hull-based palettes also consider color blends, so they are a logical fit for our problem. Tan et al. [TLG16] proposed to compute the convex hull of all pixel colors in RGB-space to extract a palette capable of reproducing all colors in the image. They proposed to simplify the convex hull until a user-selected palette size p is reached. Wang et al. [WLX19] observed that convex-hull-based palettes are sensitive to outliers. To reduce outlier sensitivity, they formulated an optimization problem to refine the simplified convex hull palette to make it more compact.

Our algorithm to extract palette is built on Tan et al.'s [TLG16] convex hull simplification and the motivation from Wang et al. [WLX19]. Rather than optimizing the output of Tan et al.'s [TLG16] method, we remove outliers from the input. We found that performing K-means clustering on the input RGB colors eliminates outliers and results in a more representative palette and region colors which better match the input in subsequent steps (see Figure 7). We use the user-selected K. The resulting clustered image is then passed as input to Tan et al.'s [TLG16] simplified convex hull palette extraction algorithm. The clustered image is not used for any other subsequent steps.

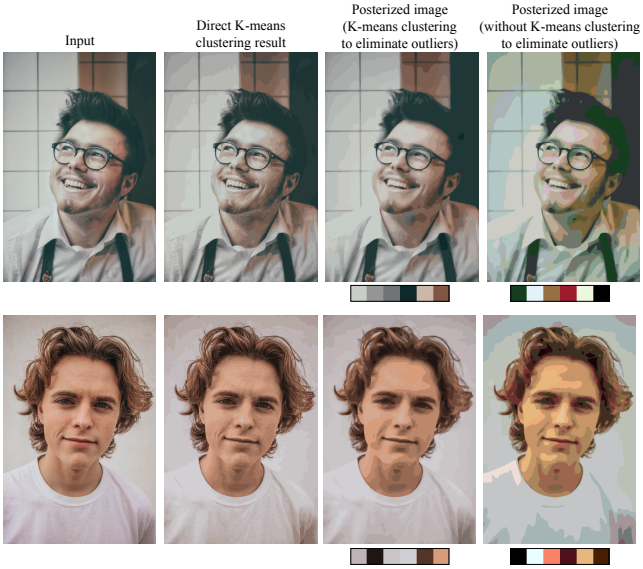

Figure 7: *K-means clustering an image's colors with large K has a subtle effect on the image itself (second column), but a large* outlier removal *effect on the simplified convex hull palettes used by our algorithm. This results in posterizations with more accurate colors (third vs. fourth columns), such as the green tint in the top row and the grey tint in the bottom row. Images ©Petr Sevcovic (top). ©Austin Wade (bottom).*

3.3. Rough Region and Color Assignment

Posterized images are partitioned into contiguous, constant-color regions. The goal of this stage is to partition the image into colored regions such that the pixels inside do not deviate too much from the underlying image yet have spatial consistency. This is a challenging (NP-Hard) optimization problem to solve in general. To solve it, we formulate the problem as multi-label optimization and make

Figure 8: *Visualizing the additive mixing weights for colors in the palette. These weights are computed before region boundary smoothing. Image ©Omid Armin*

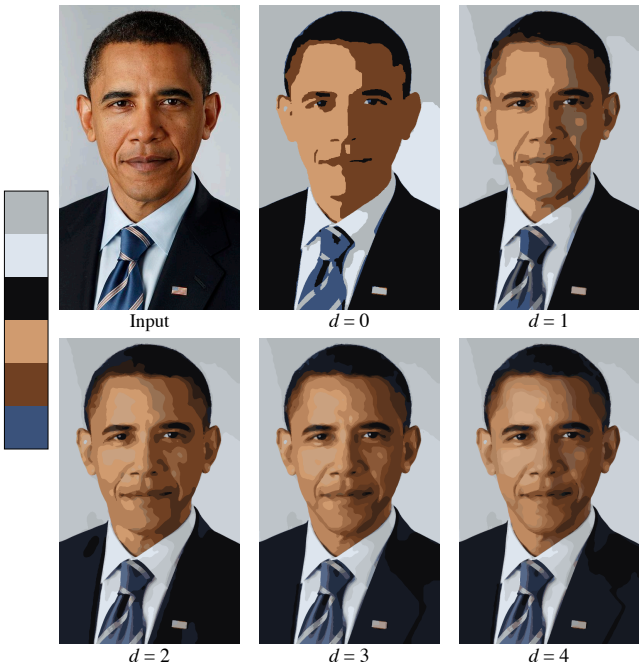

Figure 9: *Users can choose the number of discrete blending steps d between pairs of palette colors considered by our posterization algorithm. Regions with continuous gradations of tones, e.g. skin, are posterized into bands with different density. Once these approximate regions are determined, a later pipeline stage refines the piecewise-constant blend ratio for each region with a continuous optimization (Section 3.4).*

use of the solver proposed by Boykov and Kolmogorov [BVZ01]. In multi-label optimization, elements (pixels) are assigned labels (colors) that minimize a unary function (the cost of assigning a given pixel a color) and a binary function (the cost of assigning neighboring pixels different labels).

In our problem, labels are the possible colors: pairwise two-way blends between the palette colors. Users can adjust the number of blending steps d, since it directly affects the number of discrete tones visible in the output image (see Figure 9). Specifically, we create labels corresponding to the color created by

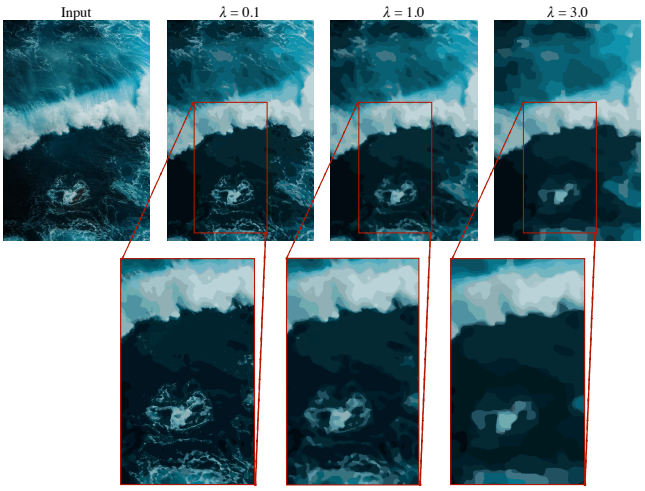

Figure 10: λ *in Equation (1) controls the clumpiness of regions in the posterized output. ©Bechir Kaddech*

blending each pair of palette colors with weights $\left(\frac{1}{d+1}, 1-\frac{1}{d+1}\right)$, $\left(\frac{2}{d+1}, 1-\frac{2}{d+1}\right)$, ..., $\left(\frac{d}{d+1}, 1-\frac{d}{d+1}\right)$.

Formally, for each pixel p in the given clustered image C, we seek a label for each pixel $f_p \in L$, where f is the labeling function and L is the finite set of labels. Our goal is to find a labeling function f that minimizes the energy

$$E(f) = E_{data}(f) + \lambda E_{pairwise}(f) \quad (1)$$

The data-term E_{data} measures, for each pixel, the difference between its label's color and its color in the original image I using the Euclidean distance in RGB-space:

$$E_{data}(f) = \sum_{p \in I} ||f_p - I_p||_2 \quad (2)$$

where f_p is the assigned label (color) for pixel p and I_p is the color of pixel p in the original image.

The strength λ of the pairwise term controls the clumpiness of regions in the output from multi-label optimization (see Figure 10). The pairwise-term $E_{pairwise}$ measures the color difference between

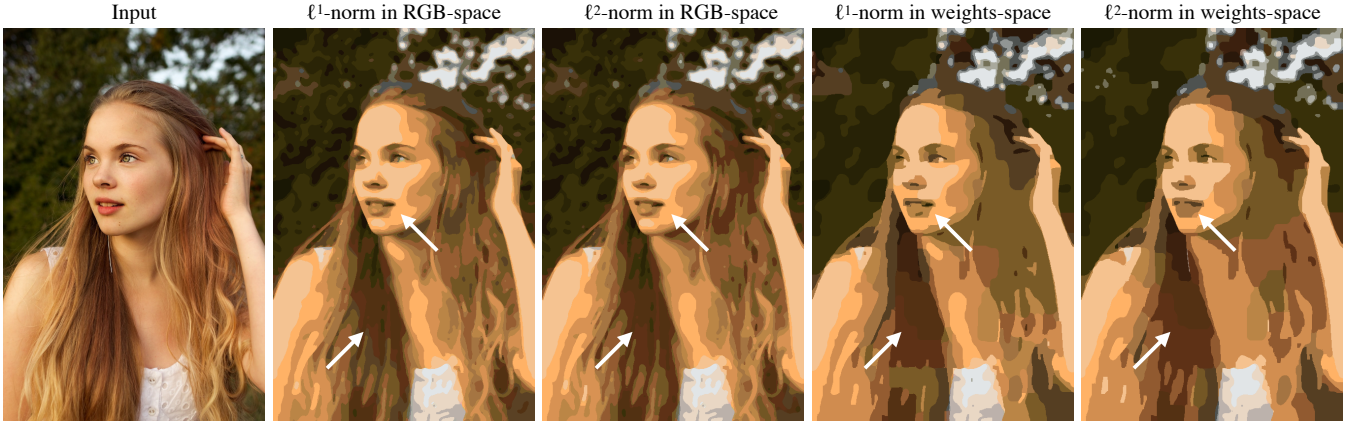

Figure 11: *The energy in Equation (1) can be expressed under different norms and spaces. The results when measuring in weights-space are clumpier since weight-space distances don't necessarily correspond to salient color differences (white arrows). ©Eleanor*

adjacent pixels, also using the Euclidean distance in RGB-space:

$$E_{pairwise}(f) = \sum_{p,q \in N} ||f_p - f_q||_2 \tag{3}$$

where N is the set of all pairs of adjacent pixels and f_p and f_q are the assigned labels (colors) for pixels p and q, respectively. We penalize differing output colors rather than differing input image colors (i.e., $I_p - I_q$), because the input colors are constant and would not affect label assignment.

We experimented with measuring the pairwise-term $E_{pairwise}$ in LAB-space, but found only minor differences compared to our results in RGB-space. We chose to use RGB-space consistently throughout our algorithm, because our convex hull is extracted in RGB-space. Extracting convex hulls in LAB-space raises challenges due to the non-convex nature of LAB-space. We experimented with the l^1-norm instead of the l^2-norm and found the differences to be minor (Figure 11). We also experimented with measuring differences directly in weight-space, but the results show significant loss of detail. This is because the norm of a difference vector in weight-space has no relationship to the perceptual differene in color-space. For example, if $w_1 = (1,0,0)$ and $w_2 = (0,1,0)$, $\|w_1 - w_2\|$ is as large as possible, but the corresponding colors may be quite similar.

While this optimization is capable of considering the cohesion between neighboring pixels (clumpiness), it does not address the region boundary smoothness necessary for high-quality posterization. Region boundary smoothing and refinement of region color blend ratios is done in subsequent steps.

3.4. Blend Refinement

The pixels in each region output from multi-label optimization are all assigned the same label. This label corresponds to a pure or blend of two palette colors. The blending weights can be written as a vector $w = \{w_1, w_2, ..., w_P\} \in \mathbb{R}^{\#P}$, where each $0 \leq w_i \leq 1$, $\sum_{i=1}^{\#P} w_i = 1$, and $\#P$ is the palette size. Each element w_i is the additive mixing weight for the i-th palette color P_i. Since we only consider pairwise blends, at most two weights will be non-zero per pixel. Since our multi-label optimization requires finite labels, it can only consider the same discrete blending ratios for all regions. We improve the color accuracy of each region, by refining the region's blending weights, to decrease its deviation from the input image (E_{data}).

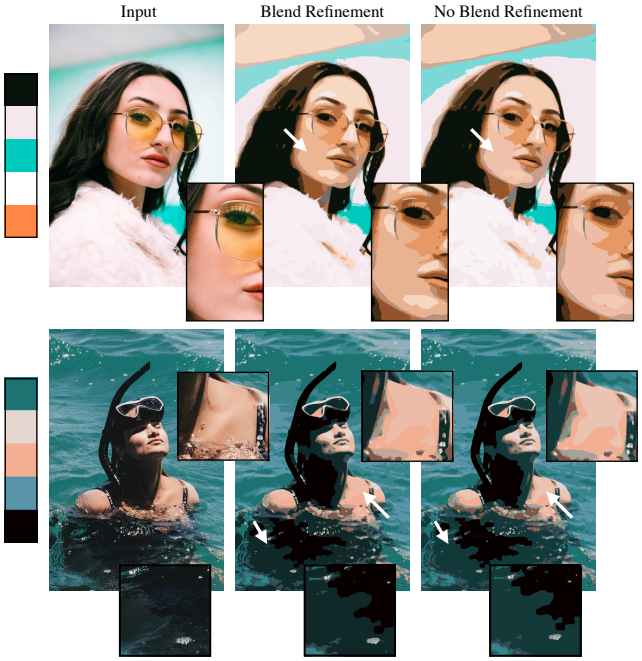

Figure 12: *Our local blend refinement improves the accuracy of region colors without modifying the color palette, shown here using a blending step $d = 1$ between the 5 palette colors. ©Jordan (top) ©Jonathan Borba (bottom).*

For a region R with label f_R, our goal is to minimize the color difference between f_R and the original image at every pixel in the re-

gion. We can express this as minimizing a squared Frobenius norm in terms of the two non-zero mixing additive weights w_i and w_j:

$$w_i', w_j' = \arg\min_{w_i, w_j} \|(w_i P_i + w_j P_j) - I_R\|_F^2 \tag{4}$$

where I_R is the matrix whose rows are the RGB colors for each pixel in region R, and P_i and P_j are the i-th and j-th palette RGB colors broadcasted to match R. Since the weights must sum to one, we can substitute $w_j = 1 - w_i$ and expand Equation (4) to obtain a constrained quadratic programming problem in one variable:

$$w_i' = \arg\min_{w_i} a w_i^2 + 2bw_j + c \tag{5}$$
$$\text{s.t. } 0 \le w_i' \le 1$$

where

$$a = ||\bar{P}_i - \bar{P}_j||_F^2,$$
$$b = < \bar{P}_i, \bar{P}_j >_F - < I_R, \bar{P}_i >_F + < \bar{P}_j, I_R >_F - ||\bar{P}_j||_F^2,$$
$$c = ||\bar{P}_j - I_R||_F^2,$$

$\bar{P}_i$ and $\bar{P}_j$ are matrices with the same dimensions as I_R whose rows are repetitions of the i-th and j-th palette colors P_i and P_j, and $< \cdot, \cdot >_F$ is the Frobenius inner product. The solution is obtained by differentiating Equation (5):

$$w_i' = \max\{0, \min\left\{1, \frac{-b}{2a}\right\}\} \tag{6}$$

Finally, the optimized color c' for region R is the straightforward additive blend $c' = w_i' P_i + (1 - w_i') P_j$. The optimized color still lies on the RGB-space line segment between the palette colors. The effects of this stage can be seen in Figure 12. If the user is recoloring (Section 4), the recolored palette colors P_i' and P_j' are used instead. The additive mixing weights are shown in Figure 8.

3.5. Region Boundary Smoothing

We wish to smooth all region boundaries simultaneously while maintaining the property that the boundaries partition the image. This is difficult to do when considering the boundary curves explicitly (in a Lagrangian manner), as we would need to consider intersections between a large number of curves. Filtering the entire image (e.g. by blurring) allows us to process all regions simultaneously. However, this poses several problems. Some approaches to filtering, such as a Gaussian blur, would introduce new, continuous colors. The median filter does not [HYT79].† However, any isotropic filter may blur away important detailed areas, such as the eyes or mouth. Anisotropic or edge-aware filters can preserve sharp edges [PM90] while smoothing elsewhere. Curvature filters can reduce the Gaussian curvature by smoothing the entire image at once as a heightfield [GS17]. However, in our posterized setting, images are piece-wise constant, so all pixels have zero Gaussian curvature and region interiors (away from edges) are already as smooth as possible.

† The per-channel median filter we use can create new discrete colors. We experimented with the spatial or geometric median, which does not. The result was similar but slower and occasionally specklier.

Table 1: *Typical parameter values.*

Parameter	Typical Range
palette size	4–15
palette blends	0–15
rare color suppression	15–50
region clumpiness	0.01–5.0
detail abstraction	0.0–1.0
boundary smoothness	3–9

The high-pass filter has been widely used in signal processing for extracting high frequencies via a cutoff frequency. The multi-label optimization we use for our initial region assignment generates sharp and accurate boundaries where there are abrupt changes in the input image, e.g. eyes, but non-smooth boundaries where the input image is smooth, e.g. the bands on a person's forehead. This phenomenon corresponds to different frequencies. We wish to preserve the boundary details where the image has high frequencies and smooth in low-frequency areas.

We convert the output color image M resulting from Blend Refinement (Section 3.4) into a grayscale image M_{gray}. Then, we compute its frequency domain Q via a $2D$ Discrete Fourier Transform, mask a small cutoff-frequency window on Q, and compute the inverse Discrete Fourier Transform to obtain our high-pass filter H.

We median-filter the color image M using H as a mask so as to only smooth the low-frequency areas and preserve detailed, high-frequency areas (see Figure 6). We perform three iterations of median filtering with the users's chosen window size and blurring threshold. As an alternative, we also allow artists to supply their own smoothing masks H for precise control (Figure 14). The result is our final, posterized image.

4. Results and Evaluation

We show a variety of posterized results throughout the paper and in galleries (Figures 20, 21, 22, and 23). Other than Figure 14, all results use automatic region boundary smoothing. Please see the supplemental materials for additional posterized imagery created by our algorithm for which we may not have the appropriate rights. In portraits (Figures 9, 12, 11, 13, 14, 18, 20, 22, 23), PosterChild preserves details near eyes while smoothing the bands on smooth regions like foreheads. PosterChild is also able to handle landscape and object photographs, creating smooth, discrete regions for clouds (Figure 15, 20, 21, and 23); waves (Figures 10 and 20); and preserving sharp structure boundaries such as rivers (Figure 20), mountains (Figure 21), roads (Figure 21), and other objects (Figures 3 and 20). We posterize an impressionistic painting in Figure 23. Typical values for our parameters can be seen in Table 1.

Our blend-aware posterization approach naturally supports recoloring effects common to artistic posterization (Figures 1, 3, and 20). Artists edit the poster's compact, representative palette; PosterChild re-uses the per-pixel mixing weights for real-time recoloring (Figure 8). Note that our region boundary smoothing is applied after recoloring. We also allow users to provide a binary saliency

Figure 13: *A comparison between our approach, Afifi's [Afi18] posterization algorithm, Xu and Kaplan's [XK08] artistic thresholding algorithm, and clustering. Afifi's [Afi18] bilateral filtering approach chooses oversaturated colors. Xu and Kaplan's [XK08] approach is limited to two tones. K-means in RGB-space produces spatially incoherent regions with noisy boundaries. With small K, color vibrancy is lost. With $K = 44$, equal to the number of blended colors in our output, K-means produces an image too close to the input. K-means in RGBXY-space can product more spatially coherent regions, but does not produce smooth region boundaries. Our posterization has smooth boundaries and accurate colors.*

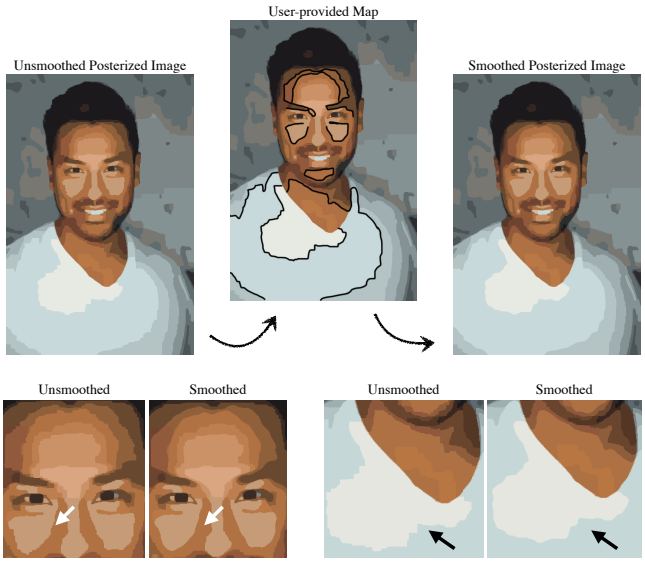

Figure 14: *Our algorithm can incorporate a user-provided saliency map to smooth only the region boundaries desired. In this case, the user wishes to capture more details around the eyes. The saliency map smoothes strong region boundaries elsewhere, on the shirt and skin, but leaves the eyes unaffected. ©Joseph Gonzalez.*

map to specify where smoothing should be applied (Figure 14). The saliency map can be created easily by painting over rough region boundaries in, e.g., Photoshop.

Performance. We implemented our technique in Python using GCO [BVZ01, BK04] and OpenCV [Bra00]. We ran our algorithm on a 2017 15" MacBook Pro with a 3.1 GHz Intel Core i7-7920HQ CPU and 16 GB of RAM. The performance of our algorithm is summarized in Table 2. The vast majority of our algorithm's processing time is spent on the multi-label optimization. This processing time can be greatly reduced by choosing a small palette size or blending steps. Alternatively, downsampling the input images produces similar results with a 4.55× geometric average speedup (Figure 19). See our Expert Study for details (Section 4.2).

Table 2: *The performance of each stage in our algorithm. The multi-label optimization (MLO) in our rough region and color assignment step (Section 3.3) is the main performance bottleneck of our algorithm. We report resolution (Res.), number of palette colors (#P), time taken for palette Extraction (Palette Extr.), multi-label optimization (MLO), blend refinement (Blend Ref.), and region boundary smoothing (Smoothing). The* downsampled *variant of our algorithm on the same input drastically reduces the time spent on MLO while producing images of similar quality (Figure 19). This reduces the total processing time by 4.55× on average.*

Input	Res.	#P	# Blend steps	Palette Extr.	MLO	Blend Ref.	Smoothing	Total	MLO fraction
Frog (Fig. 2)	540 × 360	6	3	8.86s	30.35s	0.30s	0.06s	40.18s	**75.54%**
Downsampled				1.80s	**6.47s**	0.37s	0.07s	**9.09s**	**71.18%**
Darius (Fig. 6)	533 × 800	6	2	15.61s	28.89s	0.38s	0.15s	46.47s	**62.19%**
Downsampled				4.57s	**4.97s**	0.27s	0.14s	**10.75s**	**46.23%**
Austin (Fig. 7)	951 × 634	6	2	18.61s	25.98s	0.35s	0.27s	47.20s	**55.04%**
Downsampled				4.23s	**8.14s**	0.31s	0.28s	**13.60s**	**59.85%**
Obama (Fig. 9)	795 × 469	6	2	11.31s	31.76s	0.33s	0.15s	44.73s	**71.00%**
Downsampled				2.61s	**4.42s**	0.29s	0.14s	**7.80s**	**56.67%**
Eleanor (Fig. 10)	800 × 532	6	2	15.78s	34.50s	0.34s	0.15s	52.26s	**66.02%**
Downsampled				3.84s	**5.45s**	0.29s	0.15s	**10.15s**	**53.69%**

Figure 15: *Our algorithm generates posterizations with smooth boundaries and accurate colors (and can be recolored). The bilateral filtering technique creates overexposed colors with noisier boundaries. ©Omar Abdelaal.*

4.1. Comparisons to Related Techniques

Clustering. Color quantization is an important aspect of high-quality posterization. The main drawback of clustering-based methods is that the colors deviate substantially from the input when the number of clusters is small. To avoid dissimilar colors, then, users tend to increase the number of regions. But a large number of regions tends towards the un-posterized, original image (Figure 13). Clustering-based approaches also create noisy region boundaries. Finally, clustering-based methods do not identify color blends, so the result cannot be recolored in terms of a small, representative palette.

Bilateral Filtering [Afi18]. The posterizations produced by Afifi's [Afi18] bilateral filtering-based approach have nicely contrasting discrete colors. However, the colors themselves look similar to an overexposed photo and are not expressed in a recolor-friendly format (Figure 15). Bilateral filtering also does not produce output ready for recoloring.

4.2. Expert Study

We asked 6 art and design professionals to evaluate PosterChild. 5 artists were contacted via social networks and 1 was paid $20 via the UpWork freelancer platform. Due to COVID restrictions, the evaluation was conducted remotely. We asked artists to spend approximately 10 minutes getting familiar with our program via a directed tutorial. They then posterized two different images and filled out a post-study questionnaire. All artists posterized the same two images in the same order. For each image, artists were asked

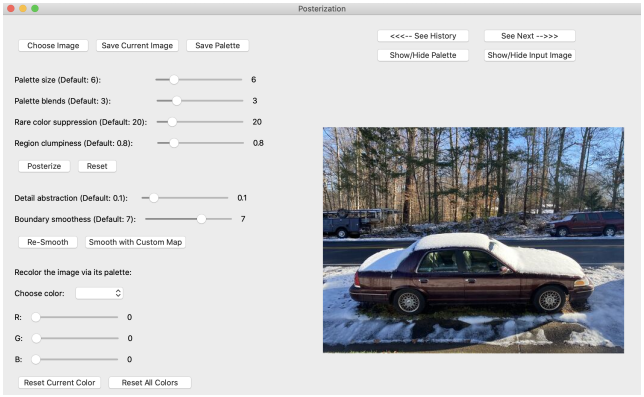

Figure 16: *PosterChild's GUI allows users to adjust posterization parameters and perform real-time recoloring.*

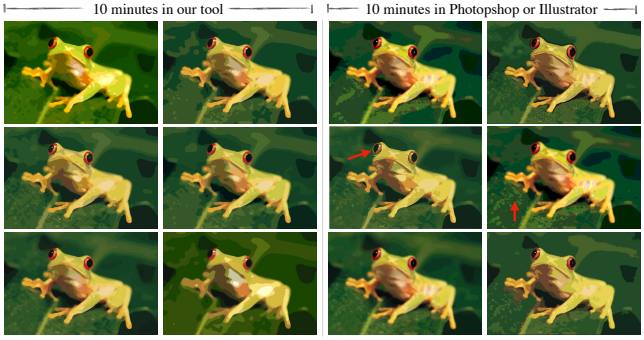

Figure 17: *Six expert users preferred the output they made using our tool compared to Photoshop or Illustrator. The input image can be seen in Figure 2. We asked the experts to create a poster similar to the bottom-right image in Figure 2.*

Figure 18: *Our expert study asked six expert users to posterize this input image using our tool and using Photoshop or Illustrator. We asked the experts to create a poster similar to the target shown on the left. The experts appreciated our tool's ability to easily adjust colors using by editing the palette blends. Posterizations created using Photoshop and Illustrator have sharp or noisy region boundaries as well. Input ©PiXimperfect*

to spend at most 10 minutes creating a posterization using PosterChild, followed by at most 10 minutes creating a posterization in Adobe Photoshop or Illustrator. Artists' output in PosterChild and Photoshop or Illustrator can be seen in Figures 17 and 18. The Tutorial, User Study, PosterChild itself, and full survey responses can be found in the supplemental materials.

Five of the six artists agreed or strongly agreed that "PosterChild is more effective at efficiently posterizing images compared to posterization support in Photoshop or Illustrator." The artists appreciated PosterChild's ability to remove small details and produce smooth region boundaries, e.g., "[T]he tool did a great job removing the water speckles while maintaining the details in the frog's eyes. This was harder to do in Photoshop and I had to resort to layer masking combined with a Gaussian blur." and "My Photoshopped frog is a bit more grainy, and I touched it up manually in parts... I prefer the overall color blending in the frog image produced by the tool."

Five of the six artists also agreed or strongly agreed that PosterChild's palette-based recoloring was useful. They noted that, e.g., "changing the palette (recolouring) was much easier here as compared to Select Colour Range -> Recolour iterations in Photoshop" and 'It was hard to get Photoshop to use the "correct" colors when I used the Posterize feature, and ended up correcting this by manually selecting a colour in the image with the magic wand and replacing it with something more suitable." and "I ended up spending additional manual effort recolouring images in PS."

All artists found the smoothing controls useful. Two artists highlighted our custom smoothing mask (Figure 14), e.g., "a great little feature to help me iron out some of the wonky details that can happen with posterization." Artists also agreed that they could use our output for downstream applications like animation or laser-cutting.

Many artists commented on the lengthy processing time needed when adjusting parameters that affect the multi-label optimization. They noted that this negatively affected their ability to experiment in the allotted time. In response to this feedback, we created a faster *downsampling* variant of our algorithm. We downsample the input image by a factor of two and then upsample prior to region boundary smoothing. To compensate for the downsampling, we decrease the penalization term (λ) in Equation (1) by a factor of two and perform a median filter with a radius of two immediately after upsampling. This decreased processing time by a factor of 4.55× on average (Table 2). Output quality is similar (Figure 19), though not identical due to the aforementioned interactions between parameters, resolution, and resampling.

Figure 19: *The* downsampling *variant of our algorithm drastically reduces processing time (Table 2) without compromising quality.*

5. Conclusion

We have shown that it is possible to automatically create posterized images qualitatively similar to those created by artists in a time-consuming manner. Our approach is based on a decomposition of the problem into palette extraction, approximate region creation, refinement, and recoloring. Our algorithm provides useful artistic controls for directing the output. By explicitly considering blended colors, our output is conducive to real-time palette-based recoloring. Our approach to color outlier removal may benefit all palette

extraction algorithms based on simplified convex hulls. PosterChild's output more faithfully captures the style of artistic posterization than previous algorithms. Most experts found PosterChild to be more effective than Illustrator or Photoshop-based posterization tools.

5.1. Limitations and Future Work

One major limitation of PosterChild is that it only allows real-time recoloring. All other parameters require a lengthy recomputation. This is true for our initial and downsampled algorithms. The main bottlenecks are the multi-label optimization for region and color assignment and, to a lesser degree, the palette extraction. One possible direction to speed up the region and color assignment is to replace the multi-label optimization with a neural network that performs semantic segmentation given an input image, a palette, and its decomposition into weight layers. In essence, the network would sparsify the weights. Our extracted palettes are more compact and representative as a result of the K-means relaxation we proposed. However, K-means is notably slow and does not scale linearly with image size. Although real-time palette extraction is still an open problem, one possible faster solution is to replace our K-means relaxation with the iterative algorithm [WLX19]. Another possible solution to obtain a more compact palette would be to find an affine transformation for the palette that moves its vertices inwards based on color sparsity. This could remove outliers while maintaining convexity. Finally, the two-stage palette extraction approach from Tan et al. [TEG18] allows for real-time layer extraction after an initial pre-processing step. This could be used to provide real-time artist control after the initial pre-processing.

Our approach does not recognize the semantics of input images. Object recognition techniques could be used to generate saliency maps for region smoothing or clutter removal. We would also like to explore different approaches for palette extraction and different output types, such as simultaneously solving for the palette and regions, and generating layered output rather than additive mixing. This could benefit physical fabrication techniques like Batik or crepe paper. Finally, we would like to improve the usability of palette-based recoloring in general, by allowing e.g., direct manipulation of pixel colors.

References

[AASP17] AKSOY Y., AYDIN T. O., SMOLIC A., POLLEFEYS M.: Unmixing-based soft color segmentation for image manipulation. *ACM Transactions on Graphics (TOG) 36* (2017), 1 – 19. 3

[Afi18] AFIFI M.: Image posterization using fuzzy logic and bilateral filter. *ArXiv abs/1802.01009* (2018). 2, 8, 9

[BK04] BOYKOV Y., KOLMOGOROV V.: An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 9 (2004), 1124–1137. doi:10.1109/TPAMI.2004.60. 8

[Bra00] BRADSKI G.: The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000). 8

[BVZ01] BOYKOV Y., VEKSLER O., ZABIH R.: Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 11 (Nov. 2001). URL: https://doi.org/10.1109/34.969114, doi:10.1109/34.969114. 5, 8

[CFL*15a] CHANG H., FRIED O., LIU Y., DIVERDI S., FINKELSTEIN A.: Palette-based photo recoloring. *ACM Trans. Graph. 34*, 4 (July 2015). URL: https://doi.org/10.1145/2766978, doi:10.1145/2766978. 3

[CFL*15b] CHANG H., FRIED O., LIU Y., DIVERDI S., FINKELSTEIN A.: Palette-based photo recoloring. *ACM Trans. Graph. 34*, 4 (July 2015). 3

[Eva] EVANS M.: Vector art with photoshop. URL: http://www.melissaevans.com/tutorials/vector-art-with-photoshop. 1

[GDA*13] GERSTNER T., DECARLO D., ALEXA M., FINKELSTEIN A., GINGOLD Y., NEALEN A.: Pixelated image abstraction with integrated user constraints. *Computers & Graphics 37*, 5 (2013), 333–347. 2, 3

[Gra10] GRAPHICS C.: Using photoshop and illustrator to achieve a posterized style. URL: http://www.atelier.cascadiagraphics.com/http:/www.atelier.cascadiagraphics.com/2010/12/posteriz. 1, 2

[Gri20] GRIESHOP A.: Posterized painting. URL: https://sites.google.com/a/minsterschools.org/minster-art-is-awesome/posterized-painting. 1

[GS17] GONG Y., SBALZARINI I. F.: Curvature filters efficiently reduce certain variational energies. *IEEE Transactions on Image Processing 26*, 4 (2017), 1786–1798. doi:10.1109/TIP.2017.2658954. 7

[Hea20] HEARTBEATS C.: How to posterize in adobe photoshop. URL: https://chasingheartbeats.com/how-to-posterize-in-adobe-photoshop/. 1

[HYT79] HUANG T., YANG G., TANG G.: A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing 27*, 1 (1979), 13–18. doi:10.1109/TASSP.1979.1163188. 7

[IWHH20] ISERINGHAUSEN J., WEINMANN M., HUANG W., HULLIN M. B.: Computational parquetry: Fabricated style transfer with wood pixels. *ACM Trans. Graph. 39*, 2 (Feb. 2020). URL: https://doi.org/10.1145/3378541, doi:10.1145/3378541. 2

[JYF*20] JING Y., YANG Y., FENG Z., YE J., YU Y., SONG M.: Neural style transfer: A review. *IEEE Transactions on Visualization and Computer Graphics 26*, 11 (2020), 3365–3385. doi:10.1109/TVCG.2019.2921336. 2

[KCWI13] KYPRIANIDIS J. E., COLLOMOSSE J., WANG T., ISENBERG T.: State of the "art": A taxonomy of artistic stylization techniques for images and video. *IEEE Transactions on Visualization and Computer Graphics 19*, 5 (2013), 866–885. doi:10.1109/TVCG.2012.160. 2

[LH13] LIN S., HANRAHAN P.: Modeling how people extract color themes from images. CHI '13, Association for Computing Machinery, p. 3101–3110. URL: https://doi.org/10.1145/2470654.2466424, doi:10.1145/2470654.2466424. 3

[M*67] MACQUEEN J., ET AL.: Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), vol. 1, Oakland, CA, USA, pp. 281–297. 2

[Meh16] MEHLOCE: I will create a black and white posterized portrait. URL: https://www.fiverr.com/mehloce/create-a-posterized-portrait. 1

[OAH11] O'DONOVAN P., AGARWALA A., HERTZMANN A.: Color compatibility from large datasets. *ACM Trans. Graph. 30*, 4 (July 2011). URL: https://doi.org/10.1145/2010324.1964958, doi:10.1145/2010324.1964958. 3

[PFC18] PHAN H. Q., FU H., CHAN A. B.: Color orchestra: Ordering color palettes for interpolation and prediction. *IEEE Transactions on Visualization and Computer Graphics 24*, 6 (2018), 1942–1955. doi:10.1109/TVCG.2017.2697948. 3

[PM90] PERONA P., MALIK J.: Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence (PAMI) 12*, 7 (1990), 629–639. 7

[Rat17] RATERMANIS A.: How to create a stylized, high contrast portrait in Adobe Illustrator. URL: https://www.ratermanis.com/blog/2017/10/27/create-a-high-contrast-portrait. 1

[RPCB17] RANG N. H. M., PRICE B. L., COHEN S., BROWN M. S.: Group-theme recoloring for multi-image color consistency. *Computer Graphics Forum 36* (2017). 3

[SKFS20] SHUGRINA M., KAR A., FIDLER S., SINGH K.: Nonlinear color triads for approximation, learning and direct manipulation of color distributions. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 97–1. 3

[TEG18] TAN J., ECHEVARRIA J., GINGOLD Y.: Efficient palette-based decomposition and recoloring of images via rgbxy-space geometry. *ACM Transactions on Graphics (TOG) 37*, 6 (2018), 1–10. 3, 11

[TLG16] TAN J., LIEN J.-M., GINGOLD Y.: Decomposing images into layers via RGB-space geometry. *ACM Trans. Graph. 36*, 1 (Nov. 2016), 7:1–7:14. URL: http://doi.acm.org/10.1145/2988229, doi:10.1145/2988229. 3, 4

[VS08] VEDALDI A., SOATTO S.: Quick shift and kernel methods for mode seeking. In *ECCV* (2008), pp. 705–718. 2

[WLX19] WANG Y., LIU Y., XU K.: An improved geometric approach for palette-based image decomposition and recoloring. *Computer Graphics Forum 38*, 7 (Nov. 2019). URL: https://doi.org/10.1111/cgf.13812, doi:10.1111/cgf.13812. 3, 4, 11

[XK08] XU J., KAPLAN C. S.: Artistic thresholding. In *Proceedings of the 6th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2008), NPAR '08, Association for Computing Machinery, p. 39–47. URL: https://doi.org/10.1145/1377980.1377990, doi:10.1145/1377980.1377990. 2, 8

Figure 20: *Our algorithm can posterize landscapes, portraits, and objects, and recolor them with palette edits. Images (top-to-bottom) ©Ekmeds Photos ©Tyler Nix ©Angelo Abear ©Matheo JBT ©Jakayla Toney*

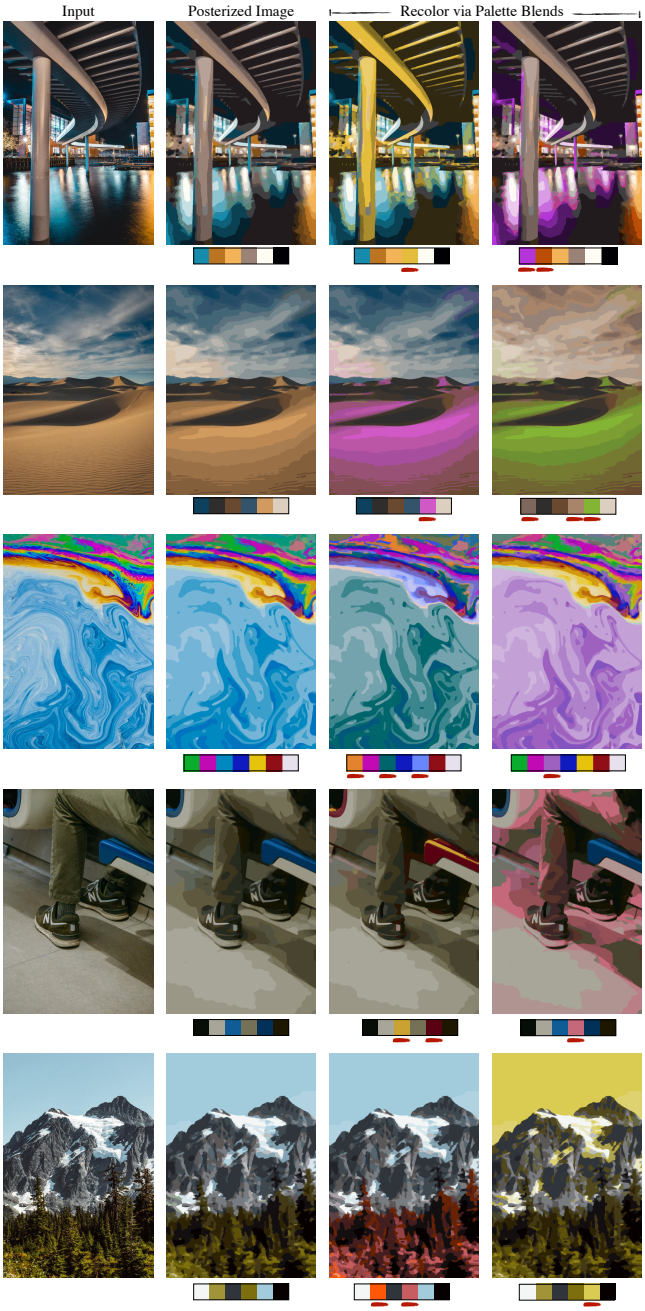

Figure 21: *Additional landscapes and objects posterized and recolored with our algorithm. Images (top-to-bottom) ©Andreas Dress ©Matteo Di Iorio ©Daniele Levis Pelusi ©Charles Deluvio ©Nathan Dumlao*

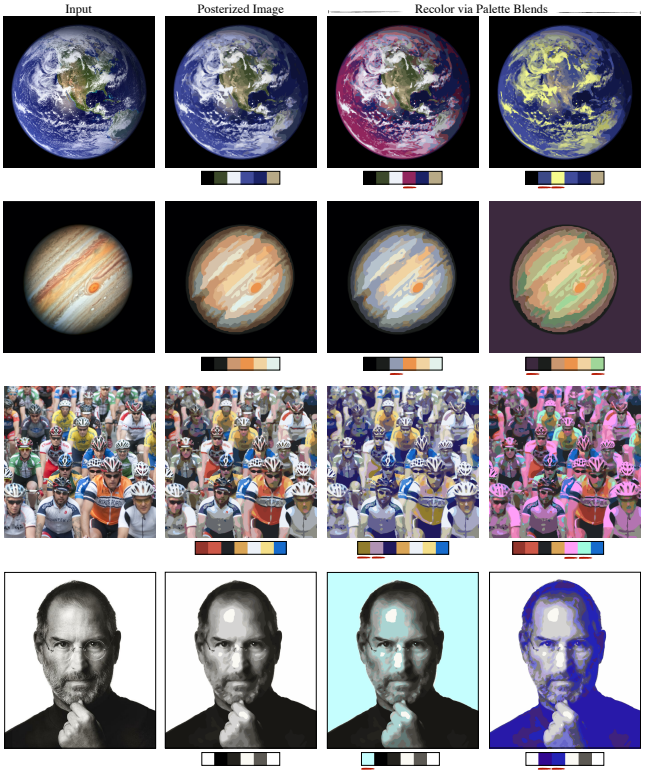

Figure 22: *Additional types of imagery (planets and people) posterized and recolored with our algorithm.*

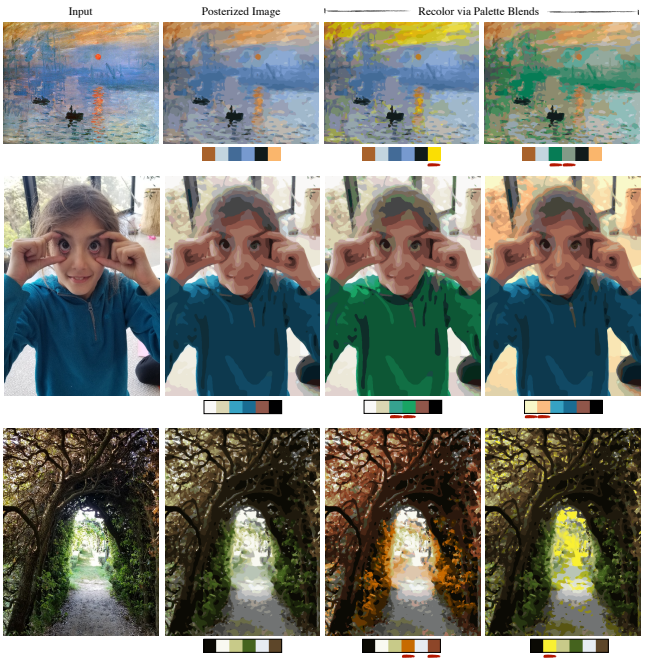

Figure 23: *Posterizing and recoloring an impressionist painting, portrait, and landscape.*