Nonlinear Color Triads for Approximation, Learning and Direct Manipulation of Color Distributions

MARIA SHUGRINA, University of Toronto, Vector Institute, and NVIDIA
AMLAN KAR, University of Toronto, Vector Institute, and NVIDIA
SANJA FIDLER, University of Toronto, Vector Institute, and NVIDIA
KARAN SINGH, University of Toronto

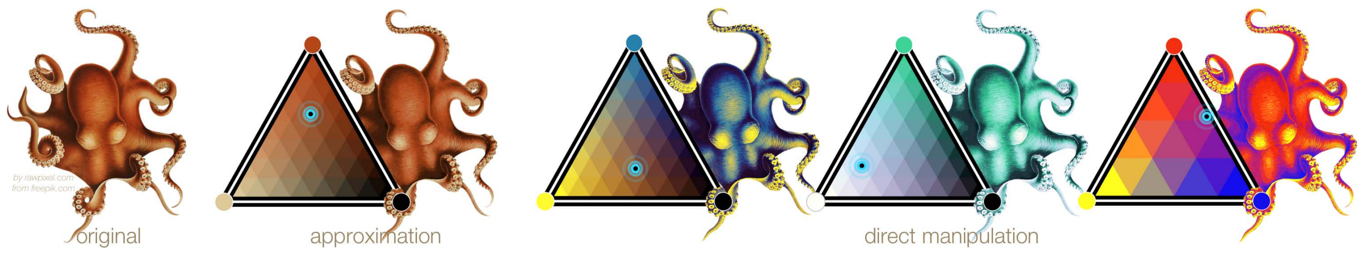


Fig. 1. Color triad is designed to approximate color distributions of shaded objects, enabling direct manipulation of colors in an image and other applications.

We present nonlinear color triads, an extension of color gradients able to approximate a variety of natural color distributions that have no standard interactive representation. We derive a method to fit this compact parametric representation to existing images and show its power for tasks such as image editing and compression. Our color triad formulation can also be included in standard deep learning architectures, facilitating further research.

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

Additional Key Words and Phrases: color palettes, recoloring, neural networks, deep learning, interactive techniques.

ACM Reference Format:
Maria Shugrina, Amlan Kar, Sanja Fidler, and Karan Singh. 2020. Nonlinear Color Triads for Approximation, Learning and Direct Manipulation of Color Distributions. *ACM Trans. Graph.* 39, 4, Article 1 (July 2020), 13 pages. https://doi.org/10.1145/3386569.3392461

1 INTRODUCTION

Existing color theme representations are either restricted to a few color swatches or are fully unconstrained. In this work, we show the power of a color representation that strikes a careful balance between expressiveness and structure. We propose non-linear color triads, which can both approximate a wide array of color distributions and naturally lend themselves to many applications.

Authors' addresses: Maria Shugrina, University of Toronto, Vector Institute, NVIDIA, shumash@cs.toronto.edu; Amlan Kar, University of Toronto, Vector Institute, NVIDIA, amlan@cs.toronto.edu; Sanja Fidler, University of Toronto, Vector Institute, NVIDIA, fidler@cs.toronto.edu; Karan Singh, University of Toronto, karan@dgp.toronto.edu.

0730-0301/2020/7-ART1 $15.00
https://doi.org/10.1145/3386569.3392461

The simplicity of our representation is inspired by discrete color palettes, typically comprised of 5-10 independent colors. Despite their limited representative power, simplicity and the ease of construction have made discrete palettes a popular tool for artists and researchers alike, and they have found use in recoloring, color theme extraction and machine learning. While discrete palettes may be suitable for some domains, they are a poor fit for larger, continuous color distributions found in most art and design. Palettes modeling more complex color distributions [Meier et al. 2004; Nguyen et al. 2015; Shugrina et al. 2017, 2019] focus on supporting the creation of novel artwork and have not been applied to representing, analyzing and editing existing images, perhaps due to their freeform unstructured nature. Color triads combine versatility and representational power with simplicity, enabling new applications.

A color triad is simply a triangular patch of RGB space governed by five parameters: three colors defining the linear interpolation space, and the fourth parameter setting a constrained amount of non-linearity. This non-linearity is critical for modeling an array of color distributions and blending behaviors, while constraints prevent degeneracies during user editing. An additional fifth parameter defines the level of discretization, making it possible to model both coarsely and densely sampled distributions. We explain the intuition behind this simple interactive representation, and demonstrate a number of useful applications. Our contributions are:

- analysis and intuition for color distribution modeling (§3)
- nonlinear color triad formulation (§4)
- algorithm for fitting color triads to images (§5)
- application of color triads to interactive editing of images (§6)
- demonstration of color triad formulation incorporated into a fully differentiable deep learning architecture (§7)

We also sketch out potential applications of our model to image compression and paint pigment modeling in the Supplemental Material. We evaluate our model quantitatively to show its representative power and qualitatively with a user study (§8).

2 RELATED WORK

Discrete color palettes: Discrete color palettes, consisting of a small (typically 5) number of solid color swatches, have become a ubiquitous representation of color combinations in design applications. Their simple representation makes discrete palettes, or themes, easy to construct and share, e.g. using a platform such as Adobe Color CC [Adobe 2017], or [COLOURlovers 2017]. Perhaps due to this simplicity, discrete color palettes have also received substantial attention from the research community. A number of approaches have been proposed for extracting discrete color palettes from images [Chang et al. 2015; Lin and Hanrahan 2013; O'Donovan et al. 2011], and also for using these palettes for image recoloring [Chang et al. 2015; Chen et al. 2014; Wang et al. 2010] or colorization [Lin et al. 2013]. In addition, crowd sourcing user preferences for this canonical color theme representation inspired work on computational aesthetics of palettes [O'Donovan et al. 2011, 2014], and machine learning approaches to palette extraction and generation [Colormind.io 2018]. Other methods for discrete color palette exploration have been proposed, e.g. a constraint-based exploration method of [Mellado et al. 2017]. However, color distributions of most man-made images are not well-modeled by small discrete sets of colors, limiting the applicability of discrete color palettes.

Other color palettes: Gamuts with a large number of colors have no canonical representation. Perhaps for this reason they are harder to construct, analyze and edit, and have received less attention from the research community. [Nguyen et al. 2015] fit color manifolds to collections of images, but their color manifold representation is not editable once constructed and requires a large amount of data to approximate. Conversely, the compact continuous palette of [Shugrina et al. 2017] is tuned for interaction during digital painting and does not naturally support tasks outside of this scenario. Another recent work [Shugrina et al. 2019] focuses on direct manipulation of color blocks to allow unstructured construction of multi-color blends. The underlying representation is limited to RGB blending, which has limited representative power, and its application outside of the proposed interface has not been explored.

Color blending: Digital color interpolation depends on its digital representation. Linear interpolation in CIELAB [CIE 2001], HSV [Smith 1978], HWB [Smith and Lyons 1996] and RGB results in vastly different gradients (Fig.2). Typically, graphic design software exposes only RGB blending due to its simplicity. However, RGB blending can result in grayish tones when transitioning from one vibrant color to another. The team behind Paper53 App [2017] hand-tuned gradients of color pairs to look more pleasing [Dannen 2012]. Another possibility is modeling physical paint blending with the Kubelka-Munk equation [Hecht 1983], e.g. to simulate watercolor [Curtis et al. 1997]. It is also possible to take a data-driven approach and learn blending and compositing behavior from examples [Lu et al. 2014], or to optimize a set of multispectral pigments to model appearance of a particular artwork [Tan et al. 2018a]. Rather than settle on a particular alternative to the standard RGB interpolation we model a variety of blending behaviors with the color triad's bending parameter. Our system exposes the control of this parameter to the artist, but it would also be possible to restrict it to better match a particular medium in a more targeted application.

Recoloring and Colorization: Chang et al. [2015] extract a *discrete* color palette from an image and use it for recoloring. Their method precomputes color transfer functions for each pixel and allows only indirect control over the resulting colors. In contrast, we use color triads to model the entire color distribution, allowing direct editing and additional creative affordances. In addition, we allow finer control by fitting multiple palettes to regions of the image. Shugrina at el. [2017] allow recoloring a painting by changing a *continuous* palette, but their method applies only to artwork created with their interface. In contrast, our method works with any design. Related to recoloring are user-guided systems for grayscale image colorization [Preferred Networks Inc. 2017; Sangkloy et al. 2017; Zhang et al. 2017]. While these systems effectively learn image semantics for colorization, they are not designed to edit an image that already contains color. We address a common use case of exploring color choices in a design where color relationships have already been defined by the creator.

Photo Segmentation and Editing: Image segmentation has been addressed for a variety of contexts [Aksoy et al. 2018; Chen et al. 2016; Qi et al. 2017; Shen et al. 2016], and desirable segmentation is, in general, application specific. We formulate segmentation to allow direct manipulation of color distributions in the segmented regions using color triads. While [Aksoy et al. 2018] also use color cues to guide segmentation, the result is not tailored to direct color distribution editing. Like most prior work, we leverage data to train a segmentation model, but our method is fully unsupervised and applicable to visual domains without labeled data. Related *color* segmentation methods separate an image into solid-colored alpha layers, e.g. [Aksoy et al. 2017; Tan et al. 2018b, 2017]. While such layers are useful for image adjustments, color relationships are baked into them and cannot be edited. Our model is more flexible, and allows novel creative affordances.

Machine Learning for Image Manipulation: Recently, a number of deep learning methods for image manipulation have been proposed. Methods such as [Gatys et al. 2015] allow users to change the style of an image, change its season, or swap out horses with zebras [Zhu et al. 2017]. [Zhu et al. 2016] allow users to generate images on the fly in an interactive manner. In our work, we show how to use the proposed triad formulation with deep learning, to produce a set of user controls (alpha masks and color triads) that give users *direct* control to easily recolor their images.

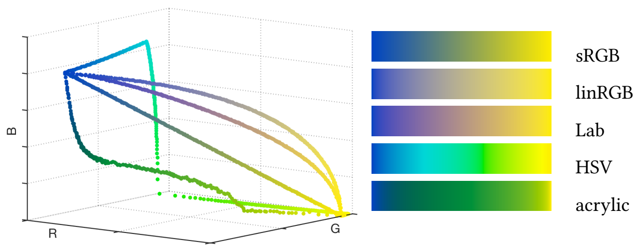


Fig. 2. **Blending Behaviors:** Linear interpolation between blue and yellow in RGB, linear RGB, CIELAB (Lab) and HSV color spaces, and between blue and yellow acrylic paints [Okumura 2005] rendered using Kubelka-Munk equation (fast transition from yellow matches paint's behavior).

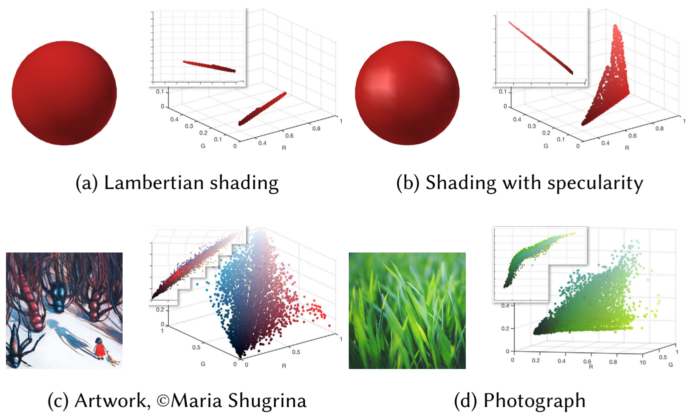


Fig. 3. **Naturally Occurring Color Distributions:** RGB plots of colors sampled from shaded objects and more complex images. While some distributions are closely approximated by a plane (b,c), others exhibit strong nonlinearity (d). Upper left inset shows alternative view angle.

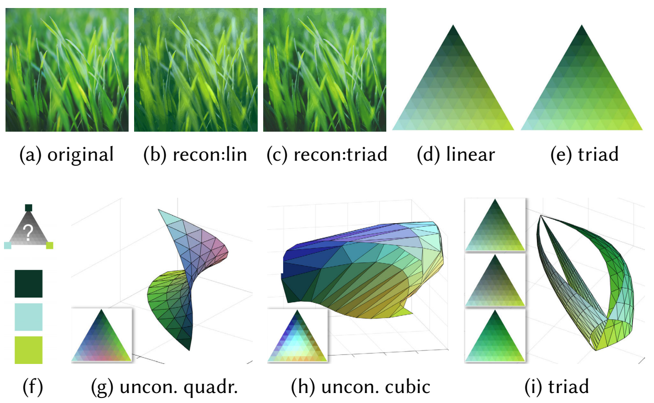


Fig. 4. **Nonlinear blending:** Image (a) reconstructed with linear 3-color blend (b, d) and with color triads (c, e). While keeping vertex colors fixed at (f), result of editing *nonlinearity only* of unconstrained quadratic (g), unconstrained cubic (h) and color triad (i) 3-color blends.

3 INTUITION AND MOTIVATION

We explain the intuition behind our design, inspired by discrete palette's simplicity and the expressiveness of other representations.

3.1 Requirements

Our direct goal is to extend the appealing properties of discrete palettes with a representation that can model much more varied color distributions. Therefore, by direct analogy with discrete palettes, this representation must be:

R1 Sparse: comprised of few degrees of freedom
R2 Interactive: easy to construct and edit interactively
R3 Structured: well-structured for learning and statistical analysis
R4 Versatile: able to model a wide array of color distributions

Here, R1-R3 are shared with discrete color palettes, and R4 implies the ability to model both discrete color gamuts and more complex color manifolds, common in art and design.

R3 is motivated by the uses of discrete color palettes for interactive color editing, statistical analysis, compression and deep learning. More versatile palette representations that fulfill R4 tend to be harder to work with. For example, the ease of indexing into a discrete palette enables image compression strategies based on a limited palette (e.g. supported by PNG and GIF image formats). However, indexing into continuous representations like [Nguyen et al. 2015; Shugrina et al. 2017] is not well-defined, making it difficult to use them for compression or recoloring, even when fit to novel images [DiVerdi et al. 2019]. Freeform interactive palettes [Meier et al. 2004; Shugrina et al. 2017, 2019] allow the same color gamut to be represented with vastly different palette configurations. Such ambiguity makes these representations hard to work with for more structured tasks. We design a palette representation that is *as easy to work with as discrete color palettes* (R1-3), but much more versatile (R4).

3.2 Intuition

To find balance between such simplicity (R1-3) and flexibility (R4), we draw intuition from the physical world. The classical shaded sphere in a diffuse material can be represented by a 2D gradient, but specular reflections result in a color distribution that is better approximated by a solid triangle in RGB (Fig. 3a,b). We make a choice to restrict our color representation to areas or objects with colors related in some way, and take the blend of three colors as the basis of our representation. Although the modeling power of three colors (see further sections) may seem surprising, it should not be: shaded objects are prevalent not only in photography, but in many artistic domains, and it is often such semantically coherent areas that require color representation and color editing.

However, restricting three color interpolation space to linear blending would severely restrict its usefulness. Although colors of some images are well approximated by a linear interpolation space (Fig. 3b,c), others exhibit a pronounced nonlinearity (Fig. 3d). In fact, a variety of color blending behaviors is to be expected, as color transitions in natural imagery could result from a variety of disjoint phenomena, for example from the nonlinear blending of physical paint media, from linear RGB blending in graphic design programs, or scattering of light within a physical substrate (Fig. 2). Simply selecting a specific space for linear interpolation would necessarily exclude many such phenomena. For example, linear RGB blend (Fig. 4b, d) optimized to approximate colors in (a) results in a significantly muted approximation, while a nonlinear three-color blend (c, e) is true to the original. Color triads parameterize color blending behavior with an interactive control.

3.3 Modeling Nonlinearity

We choose a cubic Bezier triangle [Farin 2002] to model nonlinear blending behavior. While the choice of this underlying representation is less critical, the way it is parameterized for user control is key for interactive applications (R2). For example, in the quadratic case, users could directly modify the shape of the gradients along triangle's edges, which makes control of the distribution's center difficult due to saddle point effects (Fig.4g). An unconstrained cubic case, while allowing users direct control of the distribution's center, has far too many parameters (7 control points, excluding vertices) that could easily result in degenerate self-intersecting distributions (Fig.4h). In addition, unconstrained control of the triangle shape allows modifying the distribution to the point where it has little relation to the 3 colors being interpolated (Fig.4d,g,h,i share the vertex colors in f). To constrain nonlinearity, we parameterize the control points of a cubic Bezier triangle with 3 bounded variables controlling the middle of the distribution. This allows a range of effects, but prevents degenerate distributions that stray too far (Fig.4i).

4 COLOR TRIAD REPRESENTATION

We propose color triads, a versatile, compact, interactive representation for a variety of color distributions. This section details a full mathematical formulation, which is fully differentiable and can be used as a module in a Deep Learning architecture trained with Stochastic Gradient Descent (§5.3, §7).

4.1 Overview

Each color triad is a triangular patch in RGB color space, defined by the interpolation of its vertex colors. Additional parameters control the triangle's 3D shape, resulting in nonlinear blending behaviors. The final parameter sets the discrete set of colors modeled by the triad. Formally, each color triad $\mathcal{T}$ is parameterized by:

- **vertex colors** $\mathbf{V} = [\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2]$: RGB colors of the 3 vertices
- **bending** b : a float value, defining the relative magnitude of nonlinear deformation of the triangle (sign defines direction)
- **focus point** (p_u, p_v) : barycentric coordinates of the point within the triangular patch where bending is maximal
- **subdivision level** s : an integer parameter set to 0 or above

We first formalize the sampling of colors (§4.2), and then extend the mathematical formulation to include nonlinear blending (§4.3).

4.2 Interpolation and Subdivision

The vertex colors $\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2$ define a flat triangular face in RGB space, with color at every point resulting from continuous interpolation of the vertex colors. However, many creative domains also use discrete distributions. For example, a graphic designer may choose to create a few discrete shading options in a flyer (See Fig. 12). Further, discrete samples allow integer indexing into triad colors and simplify optimization formulation. The subdivision level s determines how many distinct colors are represented. A setting of $s = 1$ includes only the 3 vertex colors, whereas $s = 16$ visually approaches a continuous distribution. Intuitively, s is defined to be exactly equal to the *number of discrete values along every edge* (Fig. 5d-e) for $s \geq 2$.

In this section we consider only linear blending. Let $\mathrm{C_L}(\mathcal{T}) = \{\boldsymbol{c}_1 \dots \boldsymbol{c}_n\}$ be the discrete set of colors defined by the color triad $\mathcal{T}$. Each such color can be defined as a linear combination of the vertex colors: $\boldsymbol{c}_i = u_{i0}\boldsymbol{v}_0 + u_{i1}\boldsymbol{v}_1 + u_{i2}\boldsymbol{v}_2 = \mathbf{V}\boldsymbol{u}_i$, where $u_{i0} + u_{i1} + u_{i2} = 1$ for all i. For a given s, each u_{ij} can take on values in the discrete set $U_s = \{0,\ 1/(s-1),\ 2/(s-1), \dots (s-1)/(s-1)\}$. Thus, the set of (linearly blended) colors in $\mathcal{T}$ is formally defined as:

$$\mathrm{C_L}(\mathcal{T}) = \{u_{i0}\boldsymbol{v}_0 + u_{i1}\boldsymbol{v}_1 + u_{i2}\boldsymbol{v}_2 \mid u_{ij} \in U_s,\ \Sigma_j u_{ij} = 1\} \quad (1)$$

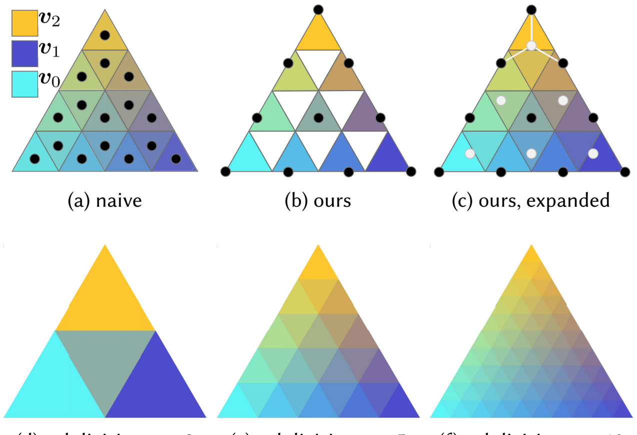


Fig. 5. **Discrete Interpolation:** Colors for subdivision levels s (d-f) are computed using the interpolation points in (b) and (c). Interpolation with barycentric coordinates at face centroids (a) would undesirably exclude the three vertex colors and pure color gradients between any two of them.

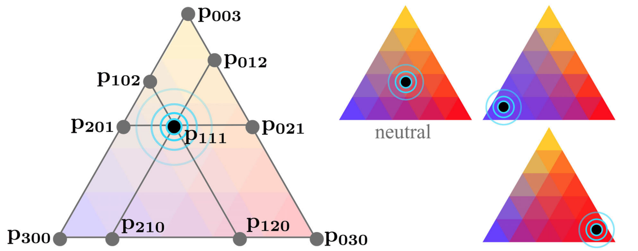
Due to the sum constraint, the size $\|\mathrm{C_L}(\mathcal{T})\|$ is exactly $1{+}2{+}\dots{+}s = s(s+1)/2$. [1] Conveniently, this is equal to the number of upright triangular patches in a triangle subdivided with s segments per side (Fig. 5b), and when rendering the triad, we set the colors of these triangles accordingly. Note that this formulation results in desirable and correct behavior of including the vertex colors themselves into the set of triad colors (See interpolation points in Fig. 5b) [2]. We also include the colors of the unfilled upside-down triangles, which are computed by averaging the neighbor's barycentric coordinates $\boldsymbol{u}_i$ (Fig. 5c). After including these additional colors into $\mathrm{C_L}(\mathcal{T})$, s^2 is the exact number of total colors (for $s \geq 2$).[3]

4.3 Nonlinear Blending

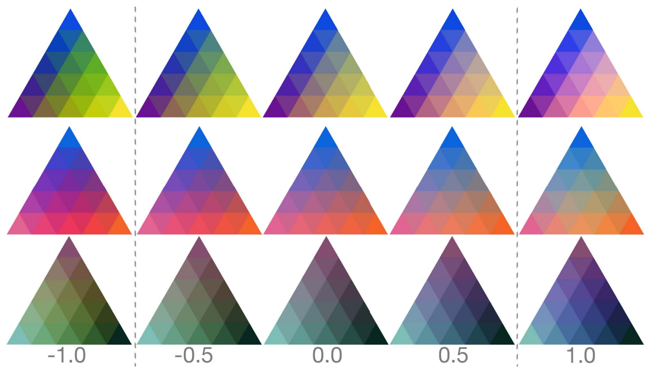
Each color triad is modeled as a cubic Bezier triangle[Farin 2002], where the position (i.e. color) of the three vertices can be set directly, while the remaining 7 control points are constrained to respond to two user-provided controls: signed bending magnitude b and the focus point (p_u, p_v). As we discuss (§3.3), these constraints prevent degeneracies and ensure cohesiveness of the resulting gamut.

The shape of the unconstrained Bezier triangle is defined by 10 control points $\mathbf{P} = \{\boldsymbol{p}_{ijk}, i + j + k = 3\}$ in RGB (Fig. 6a). Specifically, the 3D location $\boldsymbol{c}_\mathbf{P}$ of a planar color triad point parameterized by barycentric coordinates $(u_0, u_1, 1 - u_0 - u_1)$ over the triangle vertices (such as our planar interpolated colors in Eq. 1), can now be

[1]Sketch of proof: suppose 1 is split into $(s-1)$ discrete chunks, with s chunk boundaries. To pick u_{i0}, u_{i1}, u_{i2} subject to $= 1$ constraint, we must select 2 boundaries, with and without replacement, which is exactly the sum of binomial coefficients $\binom{s}{2} + \binom{s}{1}$.
[2]Conceptually this is equivalent to shrinking the triangle vertices by a factor of $(s-1)/(s * \sqrt{3})$, to align with the centroids of the subdivided triangles. We do not use the barycentric coordinates for the centroids themselves, as this would exclude the three vertex colors themselves and gradients involving only two of the interpolated colors.
[3]Sketch of proof: if counting subtriangles by rows in the triangular patch, it is an arithmetic series with s entries, first element equaling $s * 2 - 1$ and last equaling 1.



(a) Bezier triangle control points; effect of varying (p_u, p_v) for $b = 0$



(b) Effect of varying b for constant (p_u, p_v)

Fig. 6. **Nonlinear blending:** we model nonlinear blending as a cubic Bezier triangle with control points constrained to respond to only two interactive controls b and (p_u, p_v).

expressed in terms of $\boldsymbol{p}_{ijk}$ using Bernstein polynomials B_{ijk}:

$$\mathbf{c_P}(u_0, u_1) = \sum_{i+j+k=3} B_{ijk}(u_0, u_1)\boldsymbol{p}_{ijk} \tag{2}$$

$$B_{ijk}(u_0, u_1) = \frac{3!}{i!j!k!} u_0^i u_1^j (1 - u_0 - u_1)^k \tag{3}$$

To allow simple interactive control, we define the location of the triad control points $\mathbf{P}_\mathcal{T}$ in terms of the focus point (p_u, p_v) and bending magnitude b, as well as the vertex colors $\mathbf{V}$:

$$\boldsymbol{p}_{ijk} = \mathbf{V}\boldsymbol{u}_{ijk}(p_u, p_v) + f(d_{ijk}^2) \cdot b\vec{n} \tag{4}$$

where the first term is the point location in the triangle plane, and the second term is its displacement in the normal direction. First, each control point $\boldsymbol{p}_{ijk}$ is assigned barycentric coordinates $\boldsymbol{u}_{ijk}$ with respect to the triangle vertices $\mathbf{V}$. We define $\boldsymbol{u}_{111} = (p_u, p_v, 1 - p_u - p_v)$ as the focus point, and set corner control points $\boldsymbol{u}_{300}, \boldsymbol{u}_{030}, \boldsymbol{u}_{003}$ to the vertex colors. The remaining coordinates are likewise naturally expressed in terms of (p_u, p_v), as shown in Fig. 6a. Then, we displace all control points $\boldsymbol{p}_{ijk}$ except the corner points in the direction $\vec{n}$ normal to the triangle plane. The focus point control point $\boldsymbol{p}_{111}$ is displaced the most, and the displacement of other control points falls off with the distance squared d_{ijk}^2 to the central control point. (See Supplemental Material A.1 for details.)

Thus, after amending Eq. 1 with nonlinearity, the final set of colors defined by the color triad $\mathcal{T}$ is:

$$\mathbf{C}(\mathcal{T}) = \{\mathbf{c_P}(u_{i0}, u_{i1}) \mid u_{ij} \in U_s,\ \Sigma_j u_{ij} \le 1,\ \mathbf{P} = \mathbf{P}_\mathcal{T}\} \tag{5}$$

where u_{i0}, u_{i1} are exactly as in Eq. 1, and the 10 control points $\mathbf{P}_\mathcal{T}$ interpolated to obtain $\mathbf{c_P}$ are expressed in terms of the color triad $\mathcal{T}$ parameters. Our specific choices make it possible to express $\mathbf{C}(\mathcal{T})$ in matrix form with clean derivatives of the interpolated colors with respect to the color triad parameters.

4.4 Interpretation and Prior Art

Color triad is a generalization of the two color gradient, a ubiquitous tool in graphic design. As a tradeoff to simple formulation with a fixed number of parameters, color triads are designed to model colors that are related in some way, e.g. pixels of a region or shaded object – distributions that are feasible to edit interactively. More complex distributions can be modeled with multiple color triads.

Other recent exploration of sparse multi-color blends [Shugrina et al. 2017, 2019] focus on interfaces for the creation of novel artwork, while we propose a simple model for approximation, editing and analysis of colors in existing imagery. Ability to model and interactively edit nonlinear blending is critical to our applications, which rely on accurate approximation of existing distributions. Other multi-color blends either lack this representative power due to linear blending [Shugrina et al. 2019], do not support unambiguous indexing into the palette when fit to a novel artwork [DiVerdi et al. 2019; Shugrina et al. 2017] (i.e. for recoloring), or disallow interactive editing altogether [Nguyen et al. 2015]. Nonlinear color triads complement the set of existing representations for applications where requirements in §3.1 are important.

5 APPROXIMATING DISTRIBUTIONS

The full power of the color triad comes to light when it is fit to existing distributions. In this section, we optimize the parameters of a color triad $\mathcal{T}$ to approximate colors $\boldsymbol{y}_i \in Y$, where Y is an image or region. We solve this problem both using iterative optimization (§5.2) and using a Neural Network (§5.3). While the solution in §5.2 is more accurate, the formulation in §5.3 is faster and can be trivially plugged into modern Deep Learning architectures (e.g. §7).

5.1 Cost Functions

A representative color triad (palette) must contain all the colors in Y. An obvious cost is a greedy L_2 reconstruction loss in RGB:

$$E_{L2}(Y, \mathcal{T}) = \frac{1}{\|Y\|} \sum_{\boldsymbol{y}_i \in Y} \min_{\boldsymbol{c}_j \in \mathbf{C}(\mathcal{T})} \|\boldsymbol{y}_i - \boldsymbol{c}_j\|_2 \tag{6}$$

This value tracks the average distance from each target color $\boldsymbol{y}_i$ to its closest match in the set of triad colors $\mathbf{C}(\mathcal{T})$ (Eq. 5).

Because average $L2$ error does not necessarily correlate well with perceived quality, we define a more perceptually motivated metric. To assess visual quality, we compute the fraction of target colors that are not approximated "well enough", i.e. within a certain δ:

$$E_{\%}(Y, \mathcal{T}) = \frac{1}{\|Y\|} \sum_{\boldsymbol{y}_i \in Y} \mathbb{1}\left[\min_{\boldsymbol{c}_j \in \mathbf{C}(\mathcal{T})} \|lab(\boldsymbol{y}_i) - lab(\boldsymbol{c}_j)\| \ge \delta\right] \tag{7}$$

where $\mathbb{1}$ is 1 if the condition inside the brackets evaluates to true and 0 otherwise. To ensure a consistent "barely noticeable" distance δ we first map values to CIELAB space, where distances are more perceptually uniform (we use $\delta = 10$, see Fig. 13b).

A well fitting palette should also contain few irrelevant colors. Yet, a triad $\mathcal{T}$ will incur zero E_{L2} and $E_{\%}$ costs for a red patch Y if one of the vertex colors $\boldsymbol{v}_i$ is red and the remaining vertex colors bear no connection to Y. There is no clear way to measure how *relevant* a palette color is to an image.[4] We choose a measure that encourages the distribution of colors in $\mathcal{T}$ to resemble the distribution of Y, namely Kullback-Leibler divergence:

$$E_{KL} = KL(H_{\mathcal{T}} || H_Y) = -\sum_{b \in H_{\mathcal{T}}} H_{\mathcal{T}}(b) \log \frac{H_Y(b)}{H_{\mathcal{T}}(b)} \tag{8}$$

where H are $n \times n \times n$ histograms in RGB normalized to sum up to 1[5], and the summation is over histogram bins. To ensure function smoothness during optimization, $H_{\mathcal{T}}$ is a soft histogram, where the contribution of each color is a Gaussian with a fixed variance σ^2. The asymmetric nature of KL-divergence is a good fit for evaluating palette color relevance, but it is not perfect: a palette need not contain colors in the same proportion as the image itself. To mediate this discrepancy, we compute H_Y by fist computing small patch histograms, then taking the max of every bin across patches, and normalizing. This encourages every significant color in Y to attain a maximal value in its histogram. We use $n = 10$ and $\sigma = 0.5/n$.

5.2 Fitting Using Iterative Optimization

Due to its sparseness and simplicity, color triad parameters can be robustly optimized using out-of-the-box optimization methods. We use interior-point implementation in MATLAB with E_{L2} cost function and E_{KL} regularization weighed by κ. The indicator function in $E_{\%}$ makes it ill-suited for optimization, and we use it only for evaluation and tuning of κ on a small held out evaluation set (we found $\kappa = 0.0001$ to work best).

Initialization: In the case of a simple image input, we initialize Y to 10000 color samples. We initialize to linear blending with $b = 0.0$, $(p_u, p_v) = (\frac{1}{3}, \frac{1}{3})$, and set vertex colors with a heuristic: $\boldsymbol{v}_0$ is set to $\boldsymbol{y}_i \in Y$ furthest from the mean; $\boldsymbol{v}_1$ is the point furthest from $\boldsymbol{v}_0$; $\boldsymbol{v}_2$ is the point with the largest product of distances from $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$. This simple initialization works well in practice.

Tuning Subdivision Level: We exclude subdivision level s from the optimization, because the desirable value depends on the application: smallest s is desirable for compression, but large s result in highest fidelity approximation and editing results. Because lager s results in larger set of colors in the triad (Eq.1), in general higher values of s allow for lower approximation error $E_{\%}$. To tune s, we first optimize with $s = s_{max}$. Then, we perform binary search for $s < s_{max}$ that does not increase $E_{\%}$ by more than τ. In reported results, we use $s_{max} = 16$, equivalent to 256 colors, and $\tau = 0.0025$ (at most 0.2% more pixels not approximated within δ, Eq.7).

5.3 Fitting Using a Neural Network

As an alternative, we train a feed-forward Neural Network to fit color triads to distributions. This formulation can be used as a module in Deep Learning architectures, e.g. §7.

[4]E.g. inverse reconstruction loss is meaningless here, as a single matching pixel in the image would render a palette color "relevant", which is clearly not the case, especially if Y has noise or compression artifacts.

[5]To ensure numerical stability each bin is assigned a tiny minimal value.

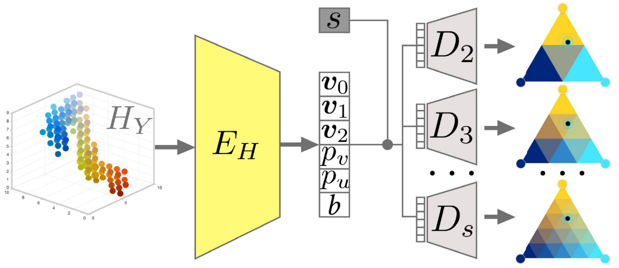


Fig. 7. **Palette Neural Network** used to fit color triads to distributions as an alternative to iterative optimization.

Input: The input to our network is a normalized histogram H_Y (as for Eq.8, we use n^3 RGB bins, $n = 10$). Unlike image input, this allows the same trained network to fit triads to color distributions from any source: whole images, soft image regions (§7), etc.

Palette Network: We use a common encoder-decoder architecture. Unlike a typical scenario, where the encoding has no direct interpretation, our encoder E_H maps an input histogram to the parameters of a color triad, while the decoder D_s is deterministic and computes the triad's colors according to Eq.5. As in §5.2, we do not explicitly optimize for the subdivision level, but use s as a switch specifying active D_s (Fig.7). We found that modeling E_H as a simple 4-layer fully connected network works well, and that performance hinges most on the loss function and training data selection.

Training: As in §5.2, we use $E_{L2} + \kappa E_{KL}$ ($\kappa = 0.0001$), and optimize the parameters of the encoder E_H using Stochastic Gradient Descent with the Adam optimizer [Kingma and Ba 2014] (learning rate 10^{-3}) by back propagation through the entire network, including the decoder D_s, which itself has no trainable weights. Color triads are designed to represent color distributions of coherent image regions, not general images. In order to approximate such input, we train the network on random patches sampled from paintings, graphic design, visualization and other image domains. See Supplemental Material for details.

6 APPLICATION: COLOR EDITING

Once fit to an image or region (§5), a color triad can be edited to interactively recolor the original. Such direct manipulation of the color distribution allows an array of novel creative explorations.

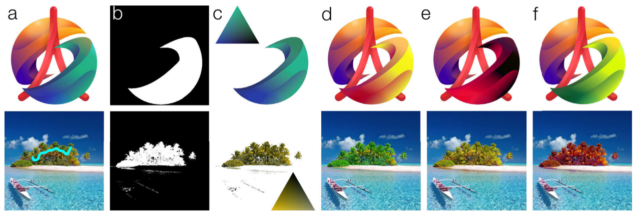


Fig. 8. **Region Editing:** A user-provided region in the SIGGRAPH Asia 2018 logo (a,b) is fit with a triad (c) and recolored (d-f). Second row shows scribble-based selection and recoloring.

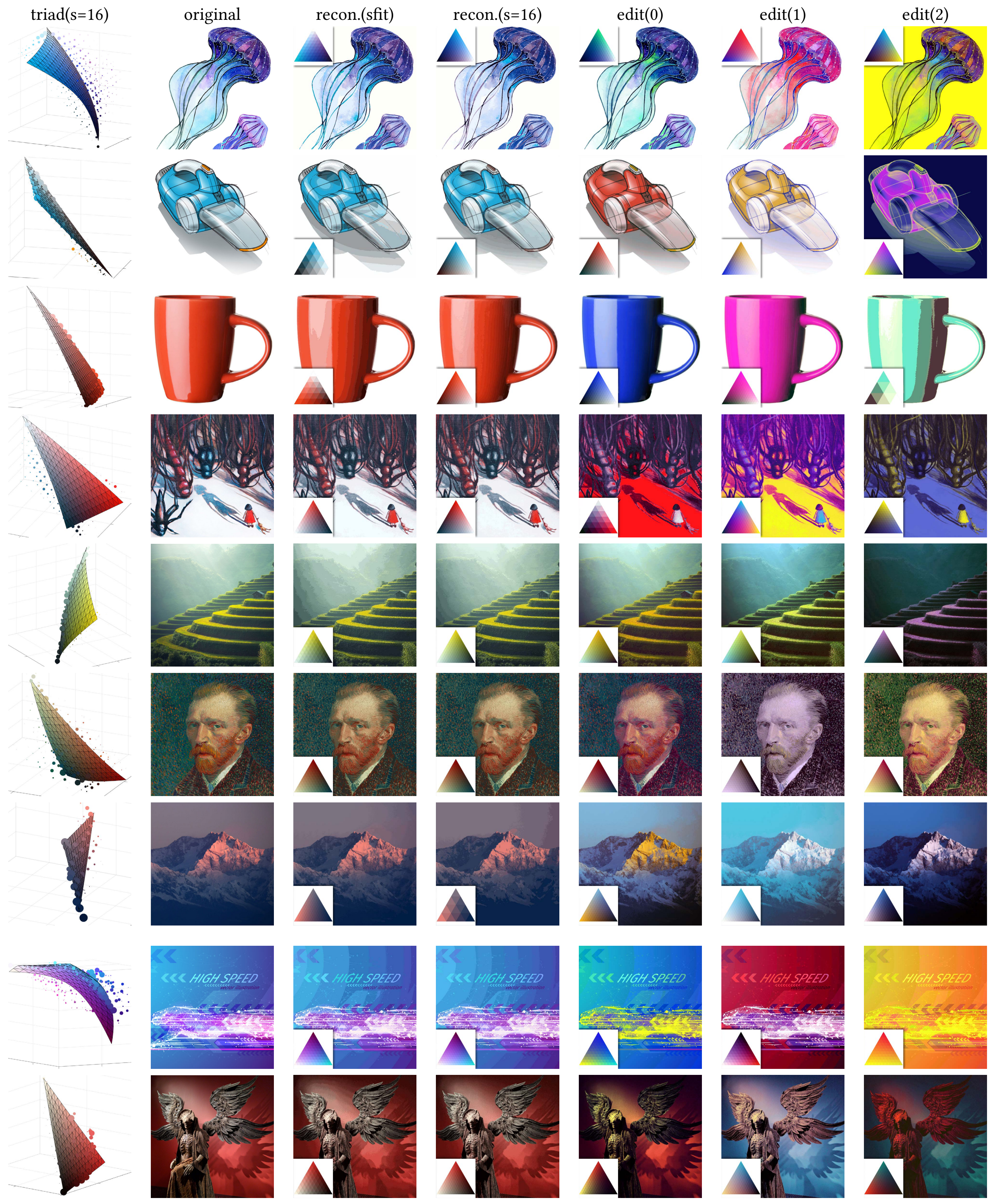


Table 1. **Fitting and Editing Results**: images reconstructed with colors from a single optimized color triad (§5.2) at $s = 16$ and with s optimized to a lower value. A variety of recoloring results (§6) by editing each triad. Rows 2-4, ©Spencer Nugent, ©George Dolgikh, ©Maria Shugrina; row 7 ©Joyston Judah.

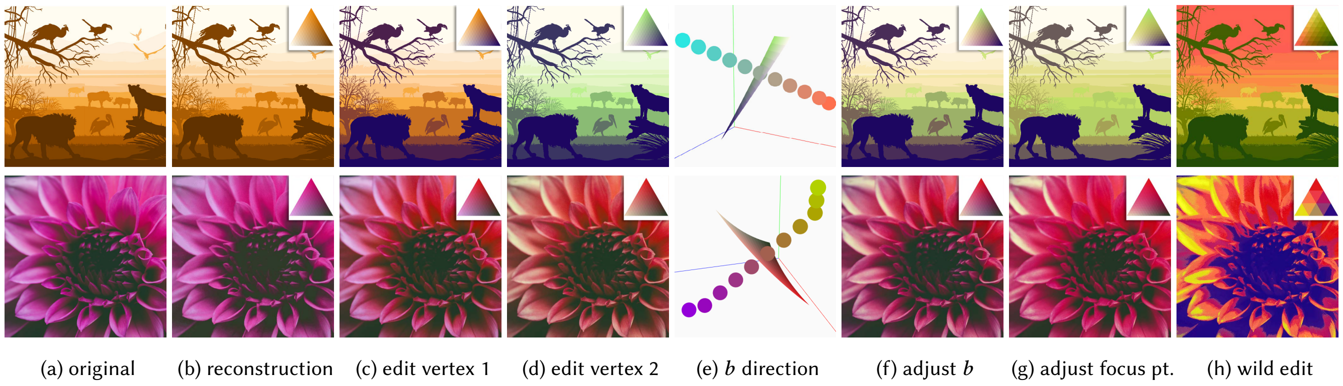


Fig. 9. **Recoloring Controls:** the effect of adjusting specific triad parameters. Row 2, ©Jeswin Thomas.

6.1 Recoloring Method

After optimizing triad $\mathcal{T}$ to an image (or region) $\mathbf{Y}$, for each pixel $\boldsymbol{y}_i \in \mathbf{Y}$ we find its closest color $\boldsymbol{c}^*(\boldsymbol{y}_i)$ among color triad colors $\mathrm{C}(\mathcal{T})$ (similarly to Eq.7) and save its barycentric coordinates idx(.) within the triad triangle. After modifying triad parameters to $\mathcal{T}'$ we can compute recolored pixel color $\boldsymbol{y}_i(\mathcal{T}')$ as follows:

$$\boldsymbol{y}_i(\mathcal{T}') = w_i \; \mathrm{C}(\mathcal{T}')[\mathrm{idx}(\boldsymbol{c}^*(\boldsymbol{y}_i))] + \Delta_i \quad (9)$$

$$\boldsymbol{c}^*(\boldsymbol{y}_i)) = \underset{\boldsymbol{c}_j \in \mathrm{C}(\mathcal{T})}{\mathrm{argmin}} \; \|lab(\boldsymbol{y}_i) - lab(\boldsymbol{c}_j)\| \quad (10)$$

where idx(.) allows indexing into the color triad colors $\mathbf{C}(.)$ (Eq.5), and $w_i = 1, \Delta_i = 0$ in the simplest case. For $\mathcal{T}' = \mathcal{T}$, this is the best reconstruction for $\boldsymbol{y}_i$ with the triad $\mathcal{T}$.

Because triads are discrete by their nature, for all but large values of s, the above equation will yield quantization errors even when $\mathcal{T}' = \mathcal{T}$. The choice of Δ_i can allow high-fidelity recoloring even with low-fidelity (small s) approximation. We found the simplest setting of $\Delta_i = \boldsymbol{y}_i - c^*(\boldsymbol{y}_i)$ to work surprisingly well [6], and all reported results use this setting. We suspect that more complex transfer functions (e.g. [Chang et al. 2015]) are unnecessary because in general most colors are approximated well. However, there is room for improvement in the formulation of w_i and Δ_i.

6.2 Recoloring Regions

Above method can be easily applied to image regions. For example, the user can mask a region using existing software, thus selecting the pixels to fit with a color triad and to recolor (Fig.8, Fig.10 row 2). If exact selection is challenging, the user could simply scribble over the range of shaded pixels that need editing, seeding a rough mask with a low flood fill threshold (as all the colors have been sampled by the scribble). This region could then be then approximated by a triad, and soft reconstruction quality map could serve as the weight during recoloring (Fig.8 row 2). We also generalize the notion of such a mask and explore fitting a collection of color triads and corresponding masks to any image using Deep Learning (§7).

[6]We also experimented with tracking displacement of target pixels from the color triad manifold in the normal direction.

6.3 Results and Discussion

Triad-based recoloring opens up a number of creative controls. The user can edit vertex colors, affecting color transitions (Fig.9c,d). The blending can be adjusted by changing b (f). The values of b tend to correspond to warmer (f, top) and cooler (f, bottom) shades and allow stretching the distribution toward the fourth complementary color. Our UI visualizes b samples in the positive and negative directions (e; visualization is clamped to valid RGB values). Finally, the user can shift the focus toward one of the vertices (g), e.g. emphasizing the green of the plain (top) or the vivid pink (bottom). Reducing subdivision level s posterizes the image (h, bottom). Combinations of such edits allow dramatic effects (h).

Color exploration is useful for many creative domains, and we demonstrate results on photography, illustration, fine art and graphic design in Tb.1. Representing semi-transparent areas in the color triad directly allows seamlessly editing the background of the medusa illustration in row 1. Colors of shaded objects can be adjusted (rows 2, 3). The method allows wild major color shifts, including inverting lights and darks (row 2, last column), majorly modifying base colors (row 4, second to last row), changing time of day (row 7), or experimenting with relighting a museum installation (last row). More subtle edits are also possible (e.g., portrait in row 6).

6.4 Comparison to Other Recoloring Methods

Our recoloring approach is complementary to existing methods, providing novel control of blending and color emphasis in the image, while keeping interaction simple. In spirit, our sparse recoloring controls resemble [Chang et al. 2015], who extract a discrete palette from an image and compute pixelwise weights for contributions of color transfer functions. Unlike our approach, their method does not allow directly controlling or visualizing the gamut. Further, the per-pixel weights are computed deterministically and an influence of the given color cannot be altered. User control of triad's nonlinearity in our method can shift "focus" from red, to gold, to dark, and modify color blending behavior interactively, producing a larger set of variations (Fig.10 bottom 2 rows).

Color decomposition approaches of [Aksoy et al. 2017; Tan et al. 2018b, 2017] accurately reconstruct an image with multiple solid-colored alpha layers, allowing layer-based editing. This family of

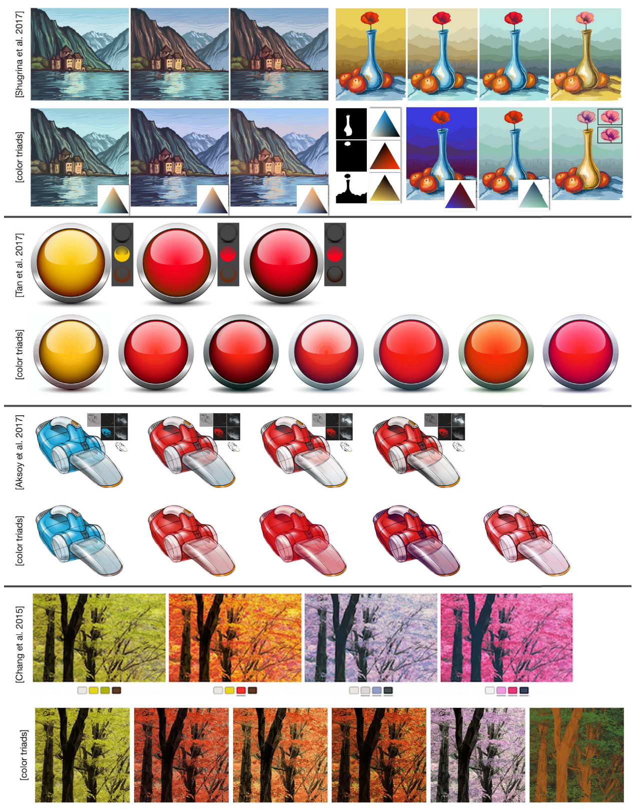


Fig. 10. **Comparison to Other Recoloring Methods:** note that color triads's ability to change blending behavior and color emphasis results in a wider range of variations for the same choice of base colors when compared to previous approaches, where blending behavior is fixed. Original images rows 3-4 ©Roman Sotola, rows 5-6 Spencer Nugen, rows 7-8 from MIT-Adobe FiveK Dataset.

approaches produces high-fidelity results and works well for targeted recoloring (e.g. red body of the vacuum, Fig.10, row 5) and general global shifts. However, just as for [Chang et al. 2015], the blending of colors is baked into the layers, making it hard to edit the balance of shadows and highlights. For example, layers of [Tan et al. 2017] allows changing the light to red and making the shadow layer darker (row 3). The wide range of effects for the interplay of shadows and highlights, as well as the warmth of blending achievable with our method (row 4) is impossible to obtain with layer-based editing. The fact that color relationships are built into the color triad representation can also cause undesired effects, such as the bleeding of the red into the highlights (row 6). This is the price our representation pays for a wide range of possible color explorations.

We also compare our approach to [Shugrina et al. 2017] (Fig.10, top row), where a link between the palette and the painting established during artwork creation allows subsequent recoloring by editing the source palette. In our case, for a novel artwork not created using color triads, user provided masks coupled with our triad fitting and recoloring achieve a similar range of explorations (second row). In addition, modifying the blending behavior of source colors can create additional effects (see flower variations in the inset).

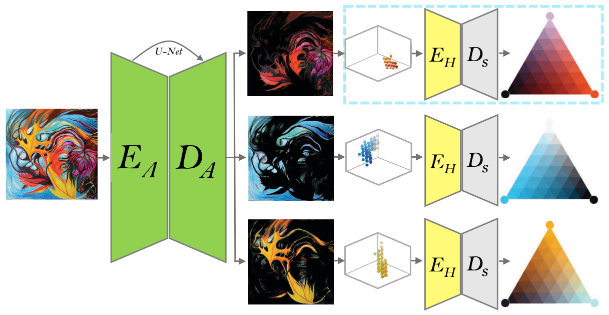


Fig. 11. **Alpha Network** (§7) converts an image into m alpha masks, which are used to weigh histograms passed to the Palette Network (§5.3) in order to fit corresponding color triads for a full reconstruction of the input image.

7 APPLICATION: DEEP LEARNING

We show one application of color triads (and palette network, §5.3) to Deep Learning. Specifically, we design a Deep Convolutional Neural Network (CNN) that approximates any input image using a collection of color triads and alpha masks that assign pixels to triads. The resulting approximation can be used for region-based recoloring (§6.2). This brief demonstration invites future work.

7.1 Formulation

For any input image $\mathbf{Y}$, our CNN $E_A D_A(\mathbf{Y})$ outputs m soft alpha masks at the input resolution. Each mask A_i weighs counts in histogram H_i, which is then passed to the palette network (§5.3) to produce triad parameters $\mathcal{T}_i$ for the masked region (Fig.11). The alpha and palette networks work together to approximate color distributions in images using several color triads. The color of a pixel (x, y) in the reconstructed image $\mathbf{Y}_R$ is computed by blending colors $\mathbf{C}(\mathcal{T}_i)$ (Eq.5) from the resulting color triads:

$$\mathbf{Y}_R(x,y) = \sum_{i=1\ldots m} A_i(x,y) \operatorname*{arg\,min}_{\mathbf{c}_j \in \mathbf{C}(\mathcal{T}_i)} \|\mathbf{Y}(x,y) - \mathbf{c}_j\|_2 \qquad (11)$$

Alpha Network: Following a common trend in image segmentation, we use an encoder-decoder architecture [Long et al. 2015], specifically, a U-Net [Ronneberger et al. 2015] with skip connections that propagate higher resolution information. A softmax on the output channels ensures that alphas sum up to 1 for every pixel.

Loss: Similarly to Eq. 6, we use $L2$ reconstruction loss between $\mathbf{Y}$ and $\mathbf{Y}_R$. Note that the training is fully unsupervised, requiring only an image dataset (Supplemental Material). To promote smoothness, we add a total variation regularization (weight of 10^{-3}).

Training: It is possible to train this model end-to-end using Stochastic Gradient Descent, because the component of the palette network(§5.3) is fully differentiable, allowing back propagation from the reconstruction loss all the way to alpha selection. To train the model, we used the Adam optimizer [Kingma and Ba 2014] with a learning rate of 10^{-3} and other default parameters. We pre-train the palette prediction network and keep it fixed. Optimizing for image reconstruction given a constant palette network forces the alpha network to output masks that correspond to image regions that can be well explained by a distribution of colors in a single color triad.

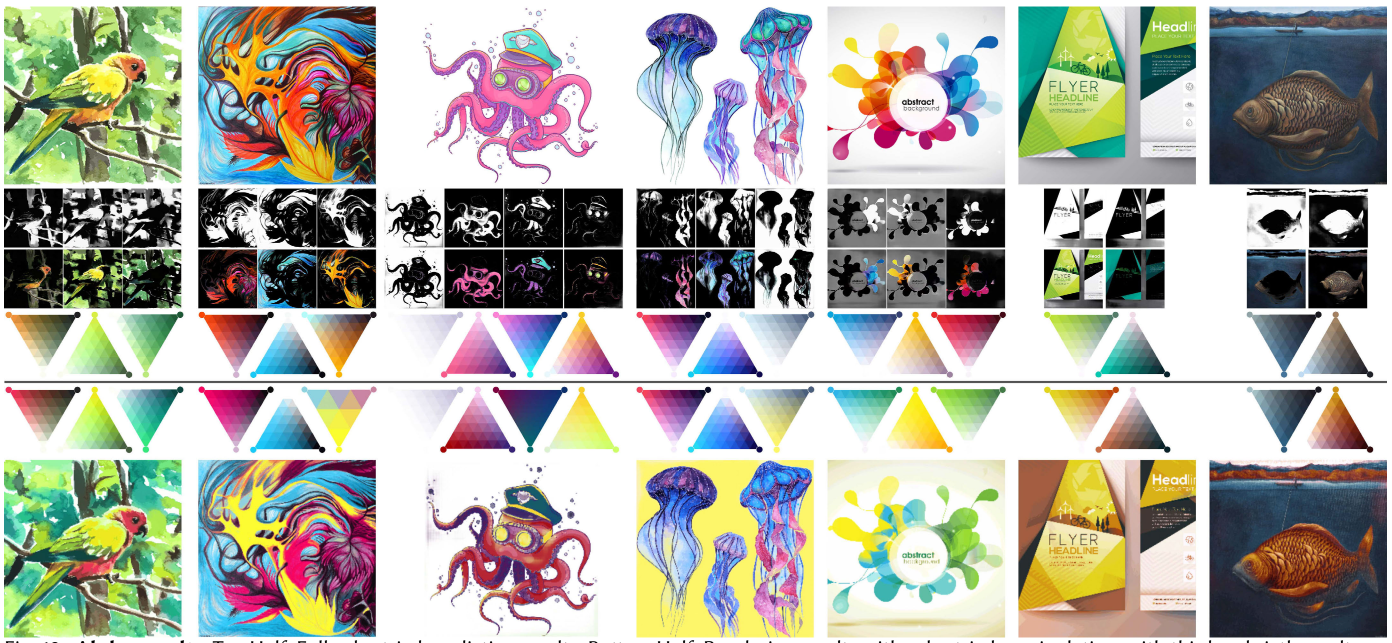


Fig. 12. **Alpha results:** Top Half: Full color triad prediction results, Bottom Half: Recoloring results with color triad manipulation, with third and sixth results taken from our user study (§8.2). Artwork columns 2, 7 ©Maria Shugrina.

Details: Different images are best represented with different *m*. We train multiple models for various *m* and select the best one at runtime using a trade off between *m* and the reconstruction quality. Given a new image, only one forward pass in the network is required to compute the full decomposition of the image into masks and color triads (170 milliseconds on nVidia GTX Titan X), which can then be used for editing. See Supplemental Material for more details.

7.2 Results and Discussion

Results of our joint alpha and triad prediction on held out test images are shown in Fig.12, top to bottom: the original image, predicted alphas as binary masks and image mattes, predicted color triads, user-edited triads and the resulting recoloring (§6). Our network can decompose images from a range of visual domains such as poster designs and paintings, allowing an array of recoloring effects using region-based triads, including subtle adjustments (col. 1, 7), major shifts to the color transitions (col. 3-5) and posterization effects (col. 2). Soft predicted alphas also allow seamless recoloring of a non-trivial background (col. 4). While our method can produce usable masks and triads, it is not perfect. For example, the mouth of the fish is erroneously grouped with the surrounding sea (col. 7), blended alphas can make editing difficult (col. 5), and noise in the alphas can introduce artifacts (col. 3). We believe that this approach is only the first step toward learning user controls specifically optimized for color editing, given *the context of a particular image.*

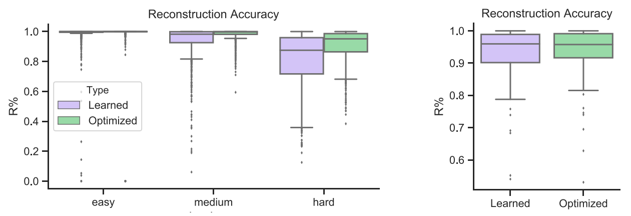
Future Directions: Our model can already re-predict triads for user-corrected alphas fast, but exact corrections are difficult to produce manually. Considering sparse user hints during inference would make our model more practical and accurate, but training for this is an open problem, given lack of data. Because color triads can represent coherent image regions, there also is promise in applying them to segmentation tasks beyond color editing. Simple color representations have already been used for seemingly unrelated tasks like tracking [Vondrick et al. 2018], and we are excited to explore other applications of this more complete color representation.

8 EVALUATION

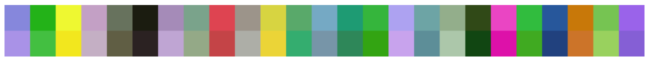
Let us revisit requirements set forth in §3.1. By their definition, color triads have a constant sparse set of user controls (R1). These controls can be edited interactively (R2), as we have demonstrated, e.g. for recoloring (§6), however, whether or not these controls provide natural interactive control requires evaluation (§8.3 below). We have demonstrated that color triads can be used for image approximation, within a deep learning architecture, as well as for compression (Supplemental Material), showing that their structure is amenable to statistical analysis and related applications (R3). However, some evaluation of the success of these techniques is required, and we provide quantitative evaluation of palette fitting (§5) in §8.1 and evaluate usability of mask-based image editing with deep segmentation (§7) in §8.4. In addition, we hope that our brief exploration of possible applications above (and in Supplemental Material) shows the merits of color triad structure (R3) and versatility (R4).

8.1 Color Triad Fitting Experiments

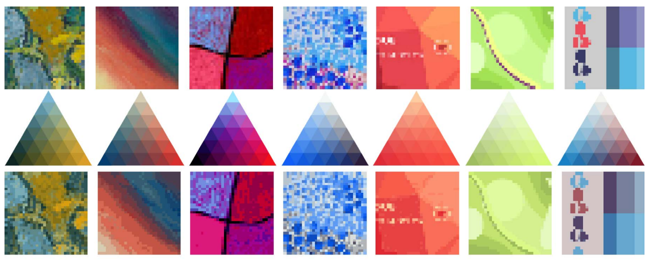
We evaluate the performance of both triad fitting methods (§5.2 and §5.3) on a collection of 3000 patches, with exactly one third designated as easy, medium and hard based on histogram entropy. Evaluation on patches is an approximation for human-marked coherent regions and objects that color triads are designed to represent. Source images have been carefully selected from various categories of art and design (see Supplemental Material). In all cases, tuning of parameters was performed on a separate held out set.



(a) $R_{\%}$ results of both approaches in §5 on 3K patches (left) and 60 full images (right)



(b) Maximal barely noticeable difference in $R_{\%}$ (we use $\delta = 10$).



(c) Palette net results on challenging patches

Fig. 13. **Palette Generation Performance:** reconstruction quality $R_{\%}$, (1-$E_{\%}$, Eq. 7) plotted for patches (based on hardness, see Supplemental Material) and full images (a), with maximal barely noticeable difference in $R_{\%}$ visualized in (b). Results of the palette petwork on particularly difficult patches in (c), top to bottom: input patch, encoded palette, image reconstructed with one best matching palette color.

We found both iterative optimization and pre-training a neural network for the task of triad fitting to work well. We show the median, 25th and 75th quantiles in the boxes, for both both learned and optimized triads. Even on hard patches, we observe a median reconstruction quality of 95.1% / 87.5% for optimization / learning, with the 25th quantile being at 86.4% / 71.7% (Fig.13a (left)). This shows the representation ability as well as the learnability of the proposed triads. The fact that traditional optimization performs better is not surprising, because triad $\mathcal{T}$ parameters are optimized directly for each input example, while with learning the network weights are optimized such that on average $E_H(H)$ produces a well-fitting palette for the input histogram H. In addition, the network takes as input a rather coarse histogram, preventing it from adjusting to subtleties. Despite these limitations, we find its performance competitive (e.g. Fig.13c on hard patches). In fact, in Fig.13a (right), we observe that learning and optimization perform similarly on challenging full images, with both approaches reaching around 96% median reconstruction quality. Also see Tb.1 for performance of iterative optimization on even more full images.

Where the learning-based approach loses in quality, it gains in speed[7]. On average the palette network fits a triad to a mini-batch of a single image in 0.046 seconds, compared to an (unoptimized) MATLAB implementation of our optimization method which ran on average in 13.6 seconds per patch[8]. The bottleneck of both approaches (during training in one case, and $\mathcal{T}$ parameter optimization in another) is the step of estimating the best reconstruction (Eq.6), which requires finding all pairwise distances. Both s and the number of image/region pixels (or samples) have a large effect on the speed of this step. We found $10K$ image samples to produce the best results during iterative optimization, and a technique of quantizing or clustering and weighing image samples during E_{L2} loss computation has an order of magnitude speed up. In addition, we have implemented a faster approximate search for the best matching color that performs an order of magnitude or more faster, depending on the fidelity of the approximation.

While reconstruction quality is very good for most images, both fitting approaches can suffer from averaging effects due to the L2 reconstruction loss (Eq.6), which can overlook rare but salient colors. In general, we found this not to be a problem for regions and images that one would want to edit coherently. For example, in the case of a single white flower in the field, one would edit the flower and the field using separate color triads. In other cases, 10000 samples typically capture even the more rare colors, such as the many flowers in the field. In both approaches, we observed an improvement in palette quality with the introduction of E_{KL} regularization (Eq.8). See Supplemental Material for details.

8.2 User Study Set Up

Experiments below were part of the same user study. We believe that in addition to researchers, color triad representation would be most useful to creative professionals. To assess the utility of this representation to designers and artists in their real world workflow, we recruited 11 participants: 3 professional graphic designers, 2 professional product/ux designers, 2 professional or advanced digital illustrators, 2 traditional artists, and 2 people regularly using graphic design software for their work. We interviewed every participant about their use of color, and asked them to perform several tasks (below). All participants were compensated with a $25 gift certificate to Starbucks for this 1 hour study.

[7]Our palette network occupies only 42MB of memory including all auxiliary data structures and independently of image size, so there is no memory-speed trade off. For reference, loading 2000x2000 RGB image into a 32bit float array requires 48MB.

[8]Both methods were evaluated on the CPU only on Mac Book Pro with 2.8 GHz Intel Core i7.

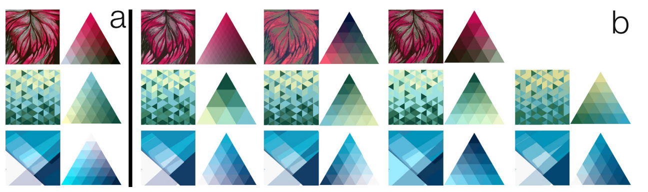


Fig. 14. **Color Matching User Study:** users were asked to edit a random color triad to match colors in one of source images (a). Palettes created by users and best reconstructions of the source images using those palettes are shown in (b). Note that users were not shown ground truth palettes (a), or given any assistance to improve reconstruction.

8.3 Color Matching Experiment

Task Set Up: To measure how intuitive it is to mentally link interactive color triad parameters to an image, we have designed an inverse recoloring task, which simulates a user trying to translate an image edit in their mind's eye to the parameters of a color triad. Rather than using the triad to recolor the image, we have asked each participant to edit a color triad (initialized to unrelated colors) to best match colors in one of three input images. To ensure fairness, all input images were edited to be representable by a color triad. A system standard color chooser for vertex colors and the same UI we instrumented for recoloring were provided, with no other special assistance (e.g., we did not compute image reconstruction as the user edited the triad). At most 3 minutes were allotted for this task.

Results: Although this seemed like a rather challenging task, the users were able to reconstruct palettes with a surprising accuracy just by looking at the image (Fig.§14). The mean $E_{\%}$ (Eq.7) for the reconstructions computed with user-edited palettes was 0.004336 (i.e. on average > 95% of pixels were reconstructed within allowable delta). E_{L2} loss was likewise low at 0.007181 per pixel for RGB mapped to $[0, 1]$. Coupled with positive response to the color editing task (below), this suggest that manipulating a direct visualization of an image's color gamuts with color triads affords intuitive control.

8.4 Color Exploration Experiments

In this more open ended part of the user study we ask participants to use the color triad interface with regions masks supplied by our deep learning method (§7), as well as a desktop application they were most familiar with to explore colors in an image. Our aim was to collect high-level feedback about the usability of color triads for color editing and exploration. With this study we focused on quick exploration, not generating high-fidelity final result.

Interview: To understand if color exploration is a real need, we asked each subject whether they create multiple versions of their designs specifically to visualize different color choices, not other design elements. 6 out of 11 emphasized that they do create different versions specifically for color exploration, while 3 reported that they create versions, but usually do not focus solely on color (versions do not really apply to 2 traditional artists). Several users stressed that they can only create multiple versions, *if there is time*, implying the time commitment necessary to explore different color variations in existing software. When asked if they typically settle on a color theme early on or continue to refine throughout the process, all but one said that color choice is an iterative process and changes throughout the design process (the one subject who replied otherwise works with company-defined branding color themes).

Recoloring Task: We asked each participant to adjust color choices in one of 3 pre-selected designs (Fig.12, col 3, col 6, and one more design) according to a loose inspiration (e.g. adjust this flyer for the fall season) using desktop software they were most comfortable with [9] and using our web-based user interface with automatically computed triads. Both tasks were allocated 7 minutes, and the ordering was randomized. The intention of this task was not to produce a final result, but to visualize an alternative color choice

[9]For users electing to work with Adobe Illustrator, Illustrator layers were provided, simulating a realistic use case of adjusting a vector design.

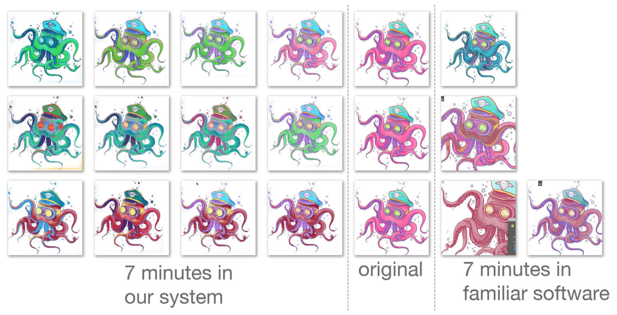


Fig. 15. **Color Exploration User Study:** Most participants were able to create many more quick variations using color triads they were not familiar with than desktop applications they were intimately familiar with in the allotted 7 minutes. This figure shows color explorations of 3 users, using Adobe Illustrator and a layered vector octopus artwork, compared to using our system with automatically computed color triads.

to gain design intuition. In Fig. 15 we show the number of different explorations three Adobe Illustrator users were able to create for the same design using our unfamiliar software, and the tool they know very well.

User feedback suggests that color triad-based exploration can augment rather than replace existing workflows, designed primarily for producing polished final results, but not ideation and exploration. In the questionnaire at the end of the study 8 out of 11 participants responded that this tool would complement the tools they already use in their work (5 strongly agree, 3 agree on a 5-point Likert scale), and 8 out of 10 felt they could be more efficient in their work if they had access to this system (4 strongly agree, 4 agree). Users also highlighted the need to visualize the behavior of the blending parameter b, which we subsequently added to our user interface (See Video). Most users made heavy use of the focus point parameter, because it has a very natural interpretation. With this initial exploration, we did not evaluate other interface possibilities, such as interactive adjustment of alpha masks, or recoloring inside user-specified regions. The positive response to color-triad-based color exploration despite the lack of these creative affordances indicates the promise of this approach for augmenting existing creative workflows.

9 DISCUSSION

We present nonlinear color triads, a new representation that balances the power of a continuous color space with the structure and artistic appeal of a discrete palette and is based on the fundamental insight that colors in shaded objects are well-approximated by a non-linear blend of three colors. We show the versatility of the proposed representation by sketching out several applications, including color editing, usage in conjunction with deep neural networks, image compression and pigment modeling (Supplemental Material). While presenting a comprehensive solution to any of these is a paper in itself and subject to future work, our explorations serve as a proof of concept. The positive response to our user study confirms the value of color triads for novel interactive applications. Furthermore, we believe that novel color representations like ours can lay the foundation for new breakthroughs in color modeling, editing, analysis, perception and creative interfaces.

ACKNOWLEDGEMENTS

We thank our user study participants for their honest feedback and insights. We are also grateful to Nicole Sultanum for help with the user study design, and to Chris De Paoli for discussion. Sanja Fidler acknowledges the Canada CIFAR AI Chair award at Vector Institute, and Maria Shugrina acknowledges CGS-D NSERC award.

REFERENCES

Adobe. 2017. Adobe Color CC. https://color.adobe.com. (2017).

Yağiz Aksoy, Tunç Ozan Aydin, Aljoša Smolić, and Marc Pollefeys. 2017. Unmixing-based soft color segmentation for image manipulation. *ACM Trans. on Graphics (TOG)* 36, 2 (2017), 19.

Yağız Aksoy, Tae-Hyun Oh, Sylvain Paris, Marc Pollefeys, and Wojciech Matusik. 2018. Semantic Soft Segmentation. *ACM Trans. on Graphics (TOG)* 37, 4 (2018), 72:1–72:13.

Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. 2015. Palette-based photo recoloring. *ACM Trans. on Graphics (TOG)* 34, 4 (2015), 139.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2016. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915* (2016).

Xiaowu Chen, Dongqing Zou, Jianwei Li, Xiaochun Cao, Qinping Zhao, and Hao Zhang. 2014. Sparse dictionary learning for edit propagation of high-resolution images. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition.* 2854–2861.

CIE. 2001. *Improvement to industrial colour-difference evaluation.* Technical Report 142-2001. Central Bureau of the CIE.

Colormind.io. 2018. Colormind. http://colormind.io/. (2018).

COLOURlovers. 2017. COLOURlovers CC. http://www.colourlovers.com. (2017).

Cassidy J Curtis, Sean E Anderson, Joshua E Seims, Kurt W Fleischer, and David H Salesin. 1997. Computer-generated watercolor. In *Proc. of the 24th annual conference on Computer graphics and interactive techniques.* ACM Press/Addison-Wesley Publishing Co., 421–430.

Chris Dannen. 2012. The Magical Tech Behind Paper For iPadĂŹs Color-Mixing Perfection. https://www.fastcompany.com/3002676/magical-tech-behind-paper-ipads-color-mixing-perfection, (2012).

Stephen DiVerdi, Jingwan Lu, Jose Echevarria, and Maria Shugrina. 2019. Generating playful palettes from images. In *Proceedings of the 8th ACM/Eurographics Expressive Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering.* Eurographics Association, 69–78.

Gerald E Farin. 2002. *Curves and surfaces for CAGD: a practical guide.* Morgan Kaufmann.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2015. A Neural Algorithm of Artistic Style. *arXiv:1508.06576* (2015).

Harry G. Hecht. 1983. A Comparison of the Kubelka-Munk, Rozenberg, and Pitts-Giovanelli Methods of Analysis of Diffuse Reflectance for Several Model Systems. *Applied Spectroscopy* 37, 4 (1983), 315–403.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Sharon Lin and Pat Hanrahan. 2013. Modeling how people extract color themes from images. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, 3101–3110.

Sharon Lin, Daniel Ritchie, Matthew Fisher, and Pat Hanrahan. 2013. Probabilistic color-by-numbers: Suggesting pattern colorizations using factor graphs. *ACM Trans. on Graphics (TOG)* 32, 4 (2013), 37.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 3431–3440.

Jingwan Lu, Stephen DiVerdi, Willa A Chen, Connelly Barnes, and Adam Finkelstein. 2014. RealPigment: Paint compositing by example. In *NPAR.* 21–30.

Barbara J Meier, Anne Morgan Spalter, and David B Karelitz. 2004. Interactive color palette tools. *IEEE Comp. Graph. App.* 24, 3 (2004), 64–72.

Nicolas Mellado, David Vanderhaeghe, Charlotte Hoarau, Sidonie Christophe, Mathieu Brédif, and Loic Barthe. 2017. Constrained palette-space exploration. *ACM Trans. on Graphics (TOG)* 36, 4 (2017), 60.

Chuong H. Nguyen, Tobias Ritschel, and Hans-Peter Seidel. 2015. Data-Driven Color Manifolds. *ACM Trans. Graph.* 34, 2 (March 2015), 20:1–20:9.

Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2011. Color compatibility from large datasets. *ACM Trans. on Graphics (TOG)* 30, 4 (2011), 63.

Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2014. Collaborative filtering of color aesthetics. *Computational Aesthetics* (2014), 33–40.

Yoshio Okumura. 2005. *Developing a spectral and colorimetric database of artist paint materials.* Master's thesis. Rochester Institute of Technology.

Paper Fifty Three Inc. 2017. Paper by FiftyThree - Sketch, Diagram, Take Notes. https://itunes.apple.com/us/app/paper-by-fiftythree-sketch/id506003812?mt=8. (2017).

Preferred Networks Inc. 2017. PaintChainer. (2017). http://paintschainer.preferred.tech/ Accessed: 2017-12-25.

Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 2017. 3d graph neural networks for rgbd semantic segmentation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition.* 5199–5208.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer, 234–241.

Patsorn Sangkloy, Jingwan Lu, Chen Fang, FIsher Yu, and James Hays. 2017. Scribbler: Controlling Deep Image Synthesis with Sketch and Color. *Computer Vision and Pattern Recognition, CVPR* (2017).

Xiaoyong Shen, Aaron Hertzmann, Jiaya Jia, Sylvain Paris, Brian Price, Eli Shechtman, and Ian Sachs. 2016. Automatic portrait segmentation for image stylization. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 93–102.

Maria Shugrina, Jingwan Lu, and Stephen Diverdi. 2017. Playful palette: an interactive parametric color mixer for artists. *ACM Trans. on Graphics (TOG)* 36, 4 (2017), 61.

Maria Shugrina, Wenjia Zhang, Fanny Chevalier, Sanja Fidler, and Karan Singh. 2019. Color Builder: A Direct Manipulation Interface for Versatile Color Theme Authoring. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems.* ACM, 456.

Alvy Ray Smith. 1978. Color gamut transform pairs. In *SIGGRAPH.* 12–19.

Alvy Ray Smith and Eric Ray Lyons. 1996. HWB—A More Intuitive Hue-Based Color Model. *Journal of Graphics Tools* 1, 1 (Jan. 1996), 3–17.

Jianchao Tan, Stephen DiVerdi, Jingwan Lu, and Yotam Gingold. 2018a. Pigmento: Pigment-based image analysis and editing. *IEEE transactions on visualization and computer graphics* 25, 9 (2018), 2791–2803.

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

Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. 2017. Decomposing images into layers via RGB-space geometry. *ACM Trans. on Graphics (TOG)* 36, 1 (2017), 7.

Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. 2018. Tracking emerges by colorizing videos. In *Proceedings of the European Conference on Computer Vision (ECCV).* 391–408.

Baoyuan Wang, Yizhou Yu, Tien-Tsin Wong, Chun Chen, and Ying-Qing Xu. 2010. Data-driven image color theme enhancement. In *ACM Trans. on Graphics (TOG)*, Vol. 29. ACM, 146.

Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. 2017. Real-time user-guided image colorization with learned deep priors. *CoRR abs/cs/0605035* cs.CV (2017).

Jun-Yan Zhu, Philipp Krahenbuhl, Eli Shechtman, and Alexei A. Efros. 2016. Generative Visual Manipulation on the Natural Image Manifold. *arXiv:1609.03552* (2016).

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *ICCV.*