# Constrained Palette-Space Exploration

Nicolas Mellado, David Vanderhaeghe, Charlotte Hoarau, Sidonie Christophe, Mathieu Brédif, Loic Barthe

# Constrained Palette-Space Exploration

NICOLAS MELLADO, IRIT, Université de Toulouse, CNRS, INPT, UPS, UT1C, UT2J
DAVID VANDERHAEGHE, IRIT, Université de Toulouse, CNRS, INPT, UPS, UT1C, UT2J
CHARLOTTE HOARAU, Univ. Paris-Est, LASTIG COGIT, IGN, ENSG
SIDONIE CHRISTOPHE, Univ. Paris-Est, LASTIG COGIT, IGN, ENSG
MATHIEU BRÉDIF, Univ. Paris-Est, LASTIG MATIS, IGN, ENSG
LOIC BARTHE, IRIT, Université de Toulouse, CNRS, INPT, UPS, UT1C, UT2J

Color palettes are widely used by artists to define colors of artworks and explore color designs. In general, artists select the colors of a palette by following a set of rules, e.g. contrast or relative luminance. Existing interactive palette exploration tools explore palette spaces following limited constraints defined as geometric configurations in color space e.g. harmony rules on the color wheel. Palette search algorithms sample palettes from color relations learned from an input dataset, however they cannot provide interactive user edits and palette refinement.

We introduce in this work a new versatile formulation enabling the creation of constraint-based interactive palette exploration systems. Our technical contribution is a graph-based palette representation, from which we define palette exploration as a minimization problem that can be solved efficiently and provide real-time feedback. Based on our formulation, we introduce two interactive palette exploration strategies: constrained palette exploration, and for the first time, constrained palette interpolation. We demonstrate the performances of our approach on various application cases and evaluate how it helps users finding trade-offs between concurrent constraints.

CCS Concepts: • **Computing methodologies** → **Computer graphics**;

Additional Key Words and Phrases: Color, Interactive editing

**ACM Reference format:**
Nicolas Mellado, David Vanderhaeghe, Charlotte Hoarau, Sidonie Christophe, Mathieu Brédif, and Loic Barthe. 2017. Constrained Palette-Space Exploration. *ACM Trans. Graph.* 36, 4, Article 60 (July 2017), 14 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073650

## 1 INTRODUCTION

Colors are central components of any visual information. In a computer-generated image, they can directly be pixel values or parameters of a visualized model (e.g. a material component). Their individual manipulation is tedious while being of primary importance for controlling the image appearance. This raises the need for high level color representations encoding the image colors in a compact set of representative parameters. A *Color palette* is a well established representation [Chang et al. 2015; Lin et al. 2013], which stores the colors that are the most representative of the content, regarding its usage and application context. Providing colors satisfying both aesthetic aspects and application constraints requires however extensive efforts from experienced users [Zhou and Hansen 2015].

We identify four main strategies followed by artists to design palettes. In the first, the user starts with an existing palette, either selected from a palette database [Brewer 2015; Murray et al. 2012], or extracted from the principal colors of an image [Chang et al. 2015]. This palette is usually considered as a bootstrap and can be further adjusted by the user [Jalal et al. 2015]. In the second, an existing palette is modified using interactive tools [Adobe 2015]. Here, the user edits the palette colors and lets the system ensure the palette coherence by considering color harmony rules (e.g. analogous, complementary) [Itten 1974; O'Donovan et al. 2011]. In the third, one or several palettes are generated by the optimization of a set of specifically defined constraints between colors. Even though palettes are not generated interactively, such systems enable the modeling of complex constraints, e.g. simulating color vision deficiencies [Yanagida et al. 2015] or decreasing energy consumption of display devices [Chuang et al. 2009]. In the last strategy, probabilistic constraints are learned in pre-process from sets of segmented images and their palettes [Lin et al. 2013; Ritchie et al. 2015]. Then, starting from an input segmented image, a set of palettes generated by stochastic sampling is proposed to the user.

From the end-user point of view, computer driven palette design can be done either by (1) using interactive tools limited to simple constraints, or by (2) browsing palettes suggestions designed by other users or generated computationally. To our knowledge, no solution has been proposed for the interactive exploration of palettes complying with advanced constraints

In this work, we introduce a new versatile formulation enabling the creation of constraint-based interactive palette exploration systems. We define the exploration of palettes as a minimization problem that we solve efficiently to provide real-time feedback. Our technical contribution is a graph-based palette representation (Figure 1-b) in which nodes are colors and edges are constraints modeled as cost functions. Based on our formulation, we introduce interactive strategies for constrained palette exploration and constrained palette interpolation (Figure 1-c,d, Section 5). We also provide guidelines to setup constrained palettes by defining their sets of colors and constraints (Section 6). These exploration schemes are versatile enough to serve as core components for the creation of interactive tools, which we illustrate with several applications (Section 7). Thanks to their interactive feedback, the proposed palette exploration systems assist the user in finding a palette considered as a

This work was partially funded by the French National Research Agency (Mapstyle project [ANR-12-COORD-0025]). The authors wish to thank the anonymous reviewers for their insightful remarks and suggestions, Mathias Paulin and Pascal Barla for their comments, and Kartini Thomas for her support and figure materials.

0730-0301/2017/7-ART60 $15.00
DOI: http://dx.doi.org/10.1145/3072959.3073650

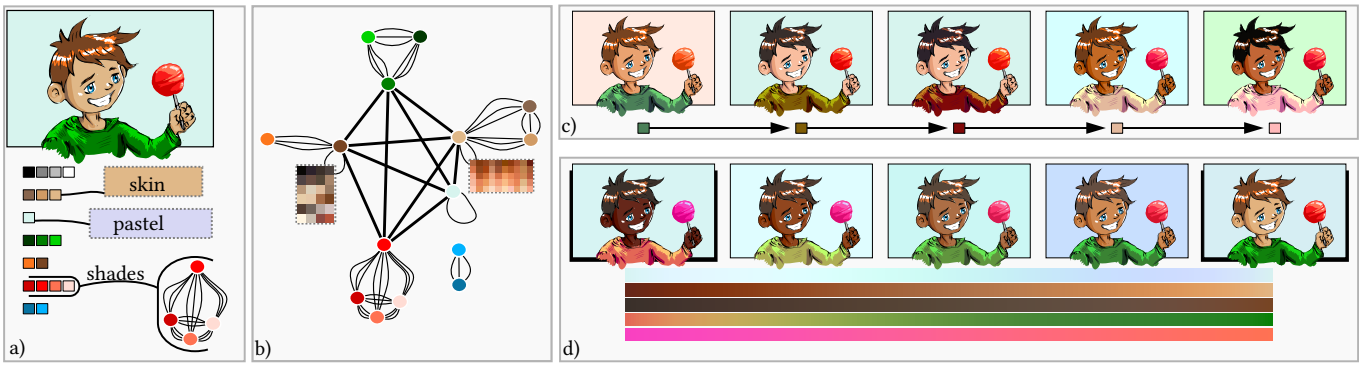



Fig. 1. a) In this work we represent the color relations (e.g. shades, anchors and contrasts) of an input palette as a constraint hypergraph (b), from which we derive two optimization-based exploration strategies. c) While the user manipulates a color, here the sweater color, the palette is updated to satisfy the constraints stored in the hypergraph. d) Given two input palettes, we compute continuous color interpolation paths to smoothly blend between the two inputs while preserving their interpolated constraints.

trade-off between multiple constraints. With our approach, users have a direct control over the exploration by adjusting the colors and the constraints if needed, while the system ensures the constraints preservation. We evaluate the benefit of our approach on exploration and interpolation scenarios (see user study in Section 8).

## 2 PREVIOUS WORK

In this section, we review the main approaches for creating and modifying color palettes. We first present user-centered approaches for the interactive design of color palettes and we follow by the exploration techniques (e.g. optimization and stochastic search) used in computational approaches. As most of the proposed approaches come with their own set of constraints, we present in a third section an overview of the principal types of color constraints.

### 2.1 User-centered palette exploration

Today, several tools are available for the interactive design of palettes. Meier et al. [2004] and more recently Jalal et al. [2015] study typical creation workflows, e.g. palette design, color replacement, and the generation of palettes similar to a given input palette. An important conclusion of their work is that users tend to experiment with the colors: they *explore* the space of feasible palettes by tweaking the individual colors and observe their arrangement.

Semi-automatic tools have been designed to ease this process. Adobe *Kuler* [Adobe 2015] and *Paletton* [Staníček 2015] are very popular tools that allow the creation and exploration of palettes with four or five colors, by moving color points within a color wheel. Those systems recompute the palette by applying hue and harmony templates (e.g. triad, complementary) interactively. The *Color gradients explorer* [Jégou 2014] focuses on cartographic palettes controlled by color gradients.

With these tools, the user has a strong feeling of control on the palette, and relies on the system to ensure constraint conformity according to simple rules (e.g. color harmonies). The strength of these systems is their predictability, however they rely on hard constraints and cannot deal with over or under-constrained configurations.

### 2.2 Computational palette exploration

Computational approaches have been proposed to generate or modify palettes. In opposition to the aforementioned approaches, the goal is not to focus on user control during the generation process, but rather to support more advanced constraints. We divide these approaches in two categories: those targeting specific applications and those for general purpose application.

*Application-specific exploration:* In the last decades, computational palette exploration has been largely studied for map design, energy efficiency (i.e. to reduce the energy required to display the palette) and adaptation to Color Vision Deficiencies (CVD).

Palette optimization for map design has mainly been addressed as a problem of color contrasts related to perceptual and semantic constraints. Constraints are designed to convey the notions of difference, order and association between colors [Bertin 1983; Brewer 1994; Buard and Ruas 2009]. Christophe [2011] models a Constraint Satisfaction Problem (CSP) to handle cartographic and artistic constraints. This formalism implies to consider discrete color spaces, and uses heuristics to handle over-constrained problems.

Palette optimization for Color Vision Deficiencies (CVD) rather focuses on finding a balance between aesthetic and accessibility constraints. Such problems are usually over-constrained, and enable the use of optimization techniques such as genetic algorithms [Ichikawa et al. 2003; Troiano et al. 2008] and probabilistic exploration [Zhou et al. 2014]. More recently, Yanagida et al. [2015] present an exploration method based on Fuzzy Constraint Satisfaction Problem (F-CSP). In contrast to the CSP used in map design, F-CSP can partially satisfy the input constraints while providing an associated satisfaction degree.

Recent advances on display devices have led to the diffusion of OLED screens, which require a variable amount of energy according to the displayed colors. Energy Aware Color Sets [Chuang et al. 2009] is an approach to optimize the color palette controlling the appearance of the displayed content in order to reduce the display energy cost. Colors are modified using a non-linear optimization scheme in order to satisfy both a global energy cost and local contrasts between each pair of colors.

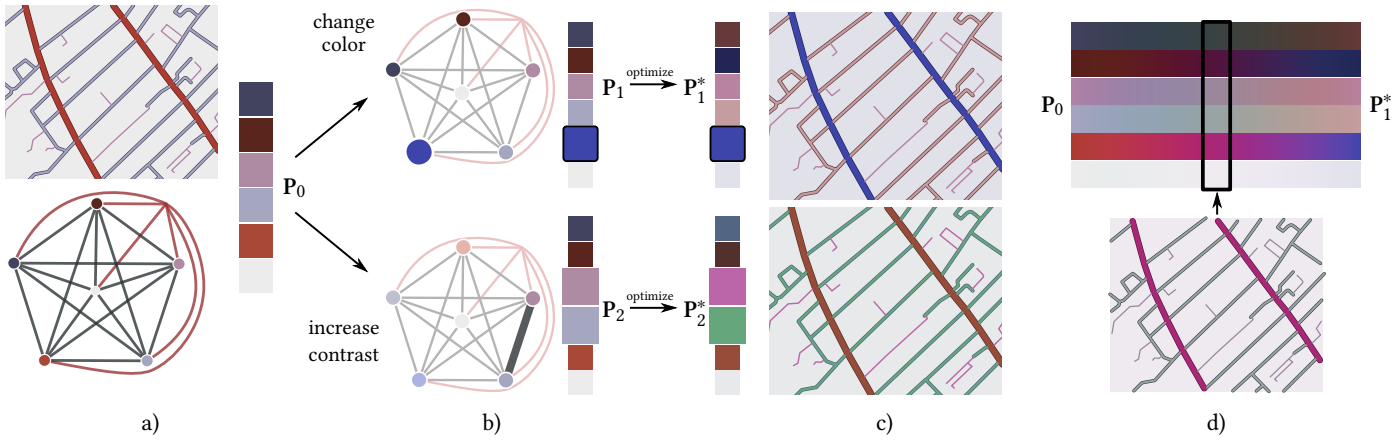



Fig. 2. a) Our approach takes as input a constrained palette defined as a set of colors connected in a graph by $n$-ary constraints. b) When the user changes a color or a constraint parameter (e.g. target contrast between colors) , c) the system updates in real-time the other colors so they respect the constraints. d) Starting from two constrained palettes, our system continuously interpolates both colors and constraints to generate new designs.

*General-purpose exploration:* More recently, computational tools have been proposed to generate palettes for arbitrary applications, i.e. palettes that do not rely on explicit *a priori* knowledge to constrain the problem definition.

*I want Hue* [Jacomy 2015] is a system generating a palette from a set of input colors controlled by the user using a k-means algorithm. The key idea of this approach is to prevent similar colors in the output palette by extracting well distributed clusters in color space. Other constraints, such as saturation bounds, can be applied during the generation. This approach can generate multiple palettes from a single input (using random seed initialization), but it remains limited by the definition of the metric used for the classification.

An orthogonal work by Lin et al. [2013] tackles this problem by learning the constraints as discrete probability distribution functions from a representative set of pairs of palettes and images. Palette generation is then performed by sampling the probability space using either Markov Chain Monte Carlo or Hamiltonian Monte Carlo [Ritchie et al. 2015] methods. While very attractive, these approaches are limited in two ways regarding our context. Firstly, the diversity of the generated palettes is restricted by the learning set, which must be carefully designed *a priori* to produce representative distribution functions. Secondly, once learned, the distribution functions are fixed and constraints cannot be adjusted easily, which limits the user controls during exploration.

## 2.3 Color constraints

Most of the aforementioned approaches provide their own constraint definitions. We classify these constraints in three categories that we use later in our formulation.

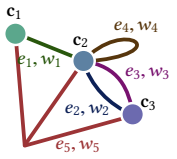
*Color properties:* A first type of constraints is related to the control of the properties of a color, regardless of the other palette colors. Low level constraints operate directly on the color coordinates in a given color space, e.g. hue or lightness. High level constraints have also been considered, for instance to control a color according to its temperature or activity [Flatla et al. 2013; Ou et al. 2004].

*Color relations:* A second type of constraints applies on small groups of colors in relation, such as contrasts and harmonies. These relations are usually modeled by preserving inter-color distances in a dedicated color space or subspace [Ichikawa et al. 2003; Sharma 2002], and have been deeply theorized and experimented by artists [Itten 1974; Munsell and Birren 1973; von Goethe 1971]. Hue and harmonic templates coming from these fundamental works are now provided by all color manipulation tools as fixed sets of rotations on the hue color wheel in the HSV space. While usual harmony templates ignore the HSV value components, Meier et al. [2004] also propose non-planar harmony templates between colors. In addition to distances, Chang et al. [2015] propose to preserve the luminance order between colors, as this property has a strong influence on human perception.

*Context-based measures:* The aforementioned constraints consider color palettes regardless of their application context. A third type of constraints takes into account the colors, the associated media (e.g. vector graphic drawing, photography or map), and/or its usage (e.g.display type and context). A typical example consists in modulating the influence of color contrasts according to the spatial relations of the elements in a represented image [Ichikawa et al. 2003], or the spatial contrasts in maps [Christophe et al. 2011]. Regarding the palette usage, several approaches constrain the palette colors to reduce the energy consumption required to display the media on specific devices while preserving its legibility [Chen et al. 2014; Chuang et al. 2009; Hoarau 2011]. In the context of Color Vision Deficiencies (CVD) adaptation, several works preserve contrasts and harmonies on a palette generated by simulating the CVD on the input palette [Ichikawa et al. 2003; Troiano et al. 2008]. In their recent work on Factor Graphs, Lin et al. [2013] learn distribution functions of color and relations properties (e.g. respectively lightness and perceptual difference), and also estimate labels on the input images segments (noise, foreground, background).

## 3 OVERVIEW

We define our palettes as a set of colors constrained by a set of properties called *constraints*, for instance enforcing the contrasts between the palette colors (Section 4.2). More formally, a palette is represented as an hypergraph $\mathbf{P} = (\mathbf{C}, \mathbf{E})$ whose nodes $\mathbf{C} = \{\mathbf{c}_i \in \mathcal{G}, i = 1..k_\mathrm{C}\}$ are the $k_\mathrm{C}$ palette colors $\mathbf{c}_i$ expressed in a color gamut $\mathcal{G}$. The hyperedges $\mathbf{E} = \{(e_j, w_j), j = 1..k_E\}$ are the $k_E$ constraints $e_j$ on these colors with their respective weights $w_j$. Constraint weights are used to modulate the relative influence of each constraint within the palette.




The right inset illustrates a palette $\mathbf{P}$ with three colors $\mathbf{c}_i$ and five weighted constraints $(e_j, w_j)$. Constraints $e_j$ can be of different arity, varying from one to the number $k_\mathrm{C}$ of colors in the palette. In this example $k_\mathrm{C} = 3$, constraints $e_1$, $e_2$ and $e_3$ are binary, $e_4$ is unary and $e_5$ is ternary.

We express a constraint of arity $n_j$ as a function $e_j : \mathcal{G}^{n_j} \rightarrow \mathbb{R}$ taking as input a subset $\mathbf{C}_j \subset \mathbf{C}$ of $n_j$ colors. The real value $e_j(\mathbf{C}_j)$ indicates how much the constraint is violated by the palette colors, i.e. the smaller the constraint absolute value, the better it is satisfied by the colors.

Using the weighted constraint values, we introduce the palette conformity error $\Pi^P$ of a palette $\mathbf{P}$ (*conformity* for short), which measures how much palette colors $\mathbf{C}$ are respecting the weighted constraints $\mathbf{E}$ (Section 4.3), i.e. $\Pi^P = 0$ when all palette constraints are exactly satisfied.

We design the exploration of palettes as an optimization process operating on the palette colors and minimizing the palette conformity. We present in Section 5 two interactive palette exploration strategies based upon this representation. The first strategy consists in editing parameters of an existing palette (e.g. colors or/and constraints, see Figure 2-b), which modifies its constraint values and thus its conformity $\Pi^P$. New palettes are then produced in real-time by optimizing the colors so that the conformity is minimized (see Section 5.1). The second strategy is a palette interpolation scheme based on the optimization of interpolation paths (in color space) between paired colors of two interpolated palettes (Figure 2-d), so that the interpolation conformity error $\Pi^I$ is minimized. This conformity error, called *interpolation conformity*, is the integral of the conformity $\Pi^P$ of all intermediate palettes $\mathbf{P}$ obtained when the interpolation parameter varies in $[0, 1]$ (Section 5.2). While the definition of a palette is agnostic to the color space, we use the Lab space to represent colors and interpolation paths and thus benefit from a perceptually uniform spacing of colors [Jain 1989].

We present in Section 6 how users can design colors and constraints to setup a palette $\mathbf{P}$. We explain how high level relations between palette colors (e.g. harmony and semantic meaning) can be expressed with constraints. We illustrate the pertinence and the practical capabilities of our exploration strategies on numerous potential application cases (Section 7). We conduct a user study to evaluate (1) how our approach helps users to perform a trade off between constraints, and (2) how well users appreciate the intermediate palettes obtained by our approach (Section 8).

## 4 CONSTRAINTS AND CONFORMITY

As explained in the previous section, a constrained palette $\mathbf{P}$ is composed of colors $\mathbf{c}_i$ and constraints $e_j$ defined as cost functions. The goal of these constraints is to evaluate the deviation of the palette colors properties $p(\mathbf{c}_i)$ - e.g. color coordinates, hue or energy consumption - to a target value $\mathrm{f}_{ref}$. We first present the color properties used in this work (Section 4.1) and, based on these properties, the derivation of unary, binary and $n$-ary constraints (Section 4.2). Once our constrained palettes well defined, we present the computation of our palette conformity error in Section 4.3.

### 4.1 Color properties

*Color space coordinates:* In our approach, a color is represented by its coordinates $\mathbf{c}$ in the Lab color space. These coordinates define a property of the represented colors from which they can be compared. In practice, any other color space or sub-space can be used to constrain different color properties. For instance, we can use any of the Lab coordinates, the luminance, the hue, saturation and value (HSV) components to measure contrasts, harmonies or shades. In that case, properties are defined by the transformations to the new color space or by the computation of a color component. For instance $p_\mathrm{hsv}(\mathbf{c})$ denotes the transformation of $\mathbf{c}$ to the HSV color space and defines as property the coordinates of $\mathbf{c}$ in that space. Following equivalent notations, $p_\mathrm{hue}(\mathbf{c})$ is the property for the hue color component and $p_\mathrm{lum}(\mathbf{c})$ the one for the luminance.

*Color Vision Deficiencies:* Several Color Vision Deficiencies (CVD) are represented as a transformation of the color coordinates expressed in the XYZ space. We simulate two types of CVD using the approach proposed by Rasche et al. [2005]. We denote as $p_\mathrm{deut}(\mathbf{c})$ and $p_\mathrm{prot}(\mathbf{c})$ the color properties returning the Lab coordinates of the color as seen by respectively a deuteranope and a proteranope.

*Energy consumption:* Recent display devices, such as OLED screens, consume a variable amount of energy for the display of different pixel colors. Based on the results presented by Chuang et al. [2009], we derive the property computing the energy required for the display of a color as:

$$p_\mathrm{cons}(\mathbf{c}) = max(p_\mathrm{red}(\mathbf{c}), p_\mathrm{green}(\mathbf{c}), p_\mathrm{blue}(\mathbf{c})),$$

where $p_\mathrm{red}(\mathbf{c})$, $p_\mathrm{green}(\mathbf{c})$ and $p_\mathrm{blue}(\mathbf{c})$ are respectively the red, blue and green color coordinates in the RGB space.

### 4.2 Constraints

*General formulation:* A constraint function $e$ is a real-valued signed cost function computing how much a set of colors violates a constraint, zero representing a perfect match. In general, any real-valued function $f(\mathbf{C})$ can be used to define a constraint function, by evaluating its deviation to a known target value $\mathrm{f}_{ref}$, using:

$$e_j(\mathbf{C}_j) = f(\mathbf{C}_j) - \mathrm{f}_{ref}. \quad (1)$$

In this research, we apply our constraints on the color properties. These constraints are presented below and in their definitions, we denote as $p(\mathbf{c})$ any of the aforementioned properties. Also notice that even when $f$ is strictly positive, $e$ can be negative when $f$ is lower than the reference $\mathrm{f}_{ref}$.

*Contrasts:* We define contrast constraints $e^c$ as binary constraints penalizing the variation of difference between two colors properties. This difference is signed and takes into account the components ordering if the property is a single color component (e.g. the hue), while it is a distance if the property is a vector (e.g. Lab coordinates). Following Eq. 1, contrast constraints are thus defined with:

$$f^c(\{\mathbf{c}_i, \mathbf{c}_j\}) = \begin{cases} p(\mathbf{c}_i) - p(\mathbf{c}_j) & \text{if } Card(p(\mathbf{c})) = 1 \\ |p(\mathbf{c}_i) - p(\mathbf{c}_j)| & \text{if } Card(p(\mathbf{c})) > 1, \end{cases}$$

and the reference value $\mathrm{f}_{ref}$ is the desired contrast between the two color properties. In this equation, $Card(p(\mathbf{c}))$ is the cardinality of the color property $p(\mathbf{c})$ and we use $|\bullet|$ as the $L_2$ norm for computational performance, even though alternative definition such as $\Delta E^*_{ab}$ in Lab might also be used.

We also define the *clamped contrast* constraint $\bar{e}^{\,c}$, which penalizes only contrast decrease. It is defined by clamping the color distance to $\mathrm{f}_{ref}$ as follow:

$$\bar{f^c}(\{\mathbf{c}, \mathbf{c}'\}) = \begin{cases} min(f^c(\{\mathbf{c}, \mathbf{c}'\}), \mathrm{f}_{ref}) & \text{if } f^c(\{\mathbf{c}, \mathbf{c}'\}) > 0, \\ max(f^c(\{\mathbf{c}, \mathbf{c}'\}), \mathrm{f}_{ref}) & \text{otherwise.} \end{cases}$$

*Anchors:* We define anchor constraints $e^a_{\mathcal{R}}$ as unary constraints penalizing the distance between a color $\mathbf{c}$ and a reference set of colors $\mathcal{R}$ with:

$$f^a_{\mathcal{R}}(\{\mathbf{c}\}) = \min_{\mathbf{c}_{ref} \in \mathcal{R}} f^c(\mathbf{c}, \mathbf{c}_{ref}).$$

In this constraint, the anchors target is defined by $\mathcal{R}$, and thus the reference value $\mathrm{f}_{ref}$ in Eq. 1 is set to 0.

We present two types of anchors, defining $\mathcal{R}$ either as a single target color $\mathbf{c}_{ref}$ or as a target color set or manifold. Anchors provide a way to model semantic information, for instance, forcing a color representing the sea within a map to remain in blue shades (e.g. by setting $p$ as $p_{\text{hue}}$, and $\mathcal{R} = \{blue\}$).

*Mean property target value:* N-ary targets constraints provide a control over a set of colors by constraining the weighted mean value of the colors' properties to a desired value $\mathrm{f}_{ref}$. They are defined with:

$$f^{\,t}(\{\mathbf{c}_1, .., \mathbf{c}_n\}) = \frac{1}{\sum_{i=1}^{n} w_{\mathbf{c}_i}} \sum_{i=1}^{n} w_{\mathbf{c}_i} p(\mathbf{c}_i).$$

The weights $w_{\mathbf{c}_i}$ set the colors importance within the constraint.

### 4.3 Conformity

As mentioned in the overview, we introduce the notion of palette conformity. The conformity $\Pi^P$ of a palette $\mathbf{P}$ is a real-valued function evaluating the deviation of the palette colors from their constraints. It also takes into account eventual out-of-gamut colors. We thus define the conformity as the sum of two error functions:

$$\Pi^P = \Pi^P_{constr} + \lambda \Pi^P_{gamut}. \tag{2}$$

where, $\Pi^P_{\text{constr}}$, is the sum of the palette weighted squared constraint values:

$$\Pi^P_{constr} = \sum_{j=1}^{k_E} w_j \left(e_j(\mathbf{C}_j)\right)^2, \tag{3}$$

and $\Pi^P_{\text{gamut}}$, which penalizes out-of-gamut colors:

$$\Pi^P_{gamut} = \sum_{i=1}^{k_c} g(\mathbf{c}_i), \tag{4}$$

with $g(\mathbf{c}_i)$ the distance to the gamut for out-of-gamut colors:

$$g(\mathbf{c}_i) = \begin{cases} \left|\mathbf{c}_i - \text{proj}_{\mathcal{G}}(\mathbf{c}_i)\right|_2 & \text{if } \mathbf{c}_i \notin \mathcal{G} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The term $\lambda$ in Equation 2 balances the two errors when dealing with out-of-gamut colors (when all colors are valid, $\Pi^P_{gamut}$ is equal to 0 and $\Pi^P = \Pi^P_{constr}$). We empirically set $\lambda = 10^3$ for all our applications, as we expect palette colors to stay within their gamut. Note that what matters is the magnitude of $\lambda$ and we observed that computations remain unsensitive to slight modifications of its value.

## 5 PALETTE-SPACE EXPLORATION

On top of our palette representation and its conformity computation, we introduce two palette exploration strategies. The first is based on a direct editing of the input palette parameters while the second relies on the interpolation between two input palettes.

### 5.1 Palette exploration

As presented in Section 2, interactive exploration techniques use simple, yet efficient constraints to adapt a palette to user inputs. Following the same concept, our interactive palette exploration strategy produces new palettes $\mathbf{P}^*$ with colors optimized from an edited palette $\mathbf{P}$ so that $\mathbf{P}^*$ always minimizes the conformity $\Pi^P$:

$$\mathbf{P}^* = \arg\min_{\mathbf{P}} \Pi^P. \tag{6}$$

In general, the palette exploration is guided by the user who directly edits the palette parameters (colors or/and constraints). Another interesting solution is to generate new sets of palette colors with a stochastic search as input editings for conformity minimizations, and let the user select the palette $\mathbf{P}^*$ he prefers (as final result or starting point for an exploration).

In practice, we use the Levenberg Marquardt algorithm (LM) to solve this non-linear minimization problem interactively (see the timings in Table 1 and the accompanying video). By its nature, LM will not necessary find global optimal results but rather palettes corresponding to local minima. This can be seen as trade-offs between the user inputs and the conformity measure, and since in most cases the conformity is a non-convex function, these local minima provide a variety of configurations to be explored. The conformity value of these configurations is given by the optimization residuals.

When optimizing the colors, care must be taken to always generate final colors inside the gamut. During the optimization, Equation 4 penalizes colors that are out-of-gamut, but it does not actually force the colors to remain in gamut. In practice, we observed that invalid colors are unlikely to be far from the gamut at the end of the optimization and therefore we project them on the gamut at the end of the process. This approach, allowing the optimizer to follow paths going temporary out-of-gamut, provides adequate results that are better than those obtained by always projecting the colors in the gamut during the optimization.

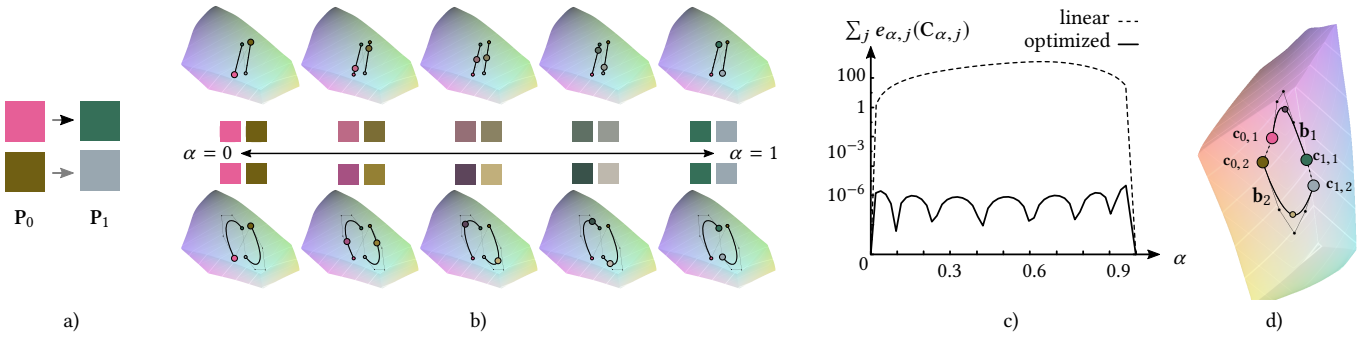



Fig. 3. Interpolation-based exploration, expressed as path-set optimization in color space. a) Two input palettes, constrained by a contrast function and global luminance criterion. b) 1D parametric exploration spaces generated by linear interpolation (top) and our approach (bottom). c) Conformity values for the intermediate palettes, with log scale. d) The paths from another point of view, with the notations used in the paper. Constraints between colors are shown with dashed lines, and the gray meshes represent the Lab gamut.

## 5.2 Palette interpolation

In the past decade, several approaches have been proposed to explore high-dimensional parametric domains (e.g. shape deformation) using linear subspaces. The key idea behind such techniques is to compute a linear parametric domain in which the user can explore a set of valid designs interactively. Recently, Nguyen et al. [2015] have presented an approach to extract color manifolds from labeled image datasets, in order to define one or two dimensional linear color subspaces associated to a label (e.g. sky, skin, apples).

In this work, we present the computation of a one-dimensional linear palette subspace by interpolating between a pair of constrained palettes $\mathbf{P}_0(\mathbf{C}_0, \mathbf{E}_0)$ and $\mathbf{P}_1(\mathbf{C}_1, \mathbf{E}_1)$ (see Figures 3 and 4). We assume that the mapping between the colors is provided by the user, such that $\mathbf{c}_{0,i}$ is mapped to $\mathbf{c}_{1,i}$.

*5.2.1 Palette matching.* In order to perform a meaningful interpolation, both palettes need to have the same topology, i.e. the same number of colors and the same constraint hypergraph topology. When the number of colors is different, we assume that parts of the colorized media are affected by respectively one color in a palette, and multiple colors in the other, as in Figure 16 (road colors). In that case, colors are duplicated, as illustrated in right inset, in order to provide a bijective mapping between palette colors as: each colors $\mathbf{c}_{0,i} \in \mathbf{C}_0$ is associated to the color $\mathbf{c}_{1,i} \in \mathbf{C}_1$, with $i \in \{1..k_C\}$, $k_C$ being the new number of colors in both palettes. If the two hypergraphs have different topologies, we extend them with the minimal number of dummy edges (constraint functions uniformly returning zero) so that they have the same topology, as illustrated in Figure 5.

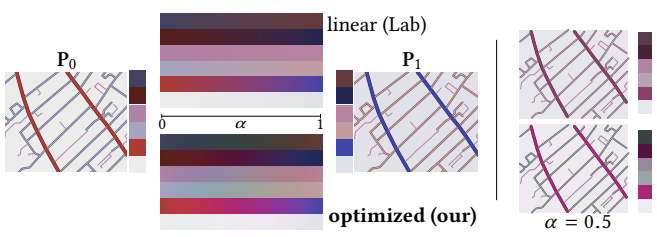



Fig. 4. Comparison between linear (top) and our optimized (bottom) interpolation spaces. Note that the contrasts, encoded in our constrained palette representation, are better preserved for the intermediate palettes with our approach. Comparisons with linear interpolations in Luv, HSV and RGB color spaces are provided in the supplementary materials.

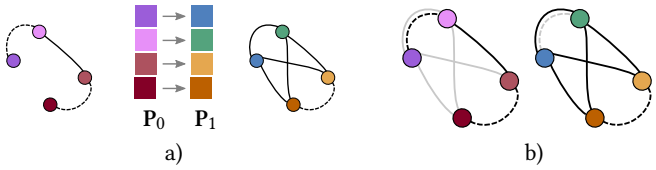



Fig. 5. Hypergraph symmetrization. a) Two input constrained palettes with different hypergraphs, composed by contrast and component-wise differences constraints, represented respectively as plain and dashed lines. b) Adapted constraint graphs, dummy edges are shown in gray.

*5.2.2 Parametric space.* Our exploration outputs an interpolation space parametrized by $\alpha \in [0, 1]$ and defined by $\mathbf{B}$ the set of paths $\mathbf{b}_i(\alpha)$ linking the associated palette colors with $\mathbf{b}_i(0) = \mathbf{c}_{0,i}$ and $\mathbf{b}_i(1) = \mathbf{c}_{1,i}$. We note $\mathbf{C}_\alpha = \{\mathbf{b}_1(\alpha), .., \mathbf{b}_{k_C}(\alpha)\}$ the set of interpolated palette colors.

*5.2.3 Palette interpolation.* As we define the palette exploration as the result of an optimization minimizing the conformity $\Pi^P$(with Equation 6), we obtain the interpolated colors $\mathbf{C}_\alpha$ by computing the set $\mathbf{B}^*$ of curves whose parameters optimization minimizes an interpolation conformity error $\Pi^I$:

$$\mathbf{B}^* = \arg\min_{\mathbf{B}} \Pi^I. \tag{7}$$

*5.2.4 Interpolation conformity.* In this equation, the interpolation conformity $\Pi^I$ is defined as the sum of three error functions: $\Pi^I_{constr}$ evaluating the constraints deviations of all palettes $\mathbf{C}_\alpha$ as the integral of the error function $\Pi^P_{constr}$ (see Equation 3)

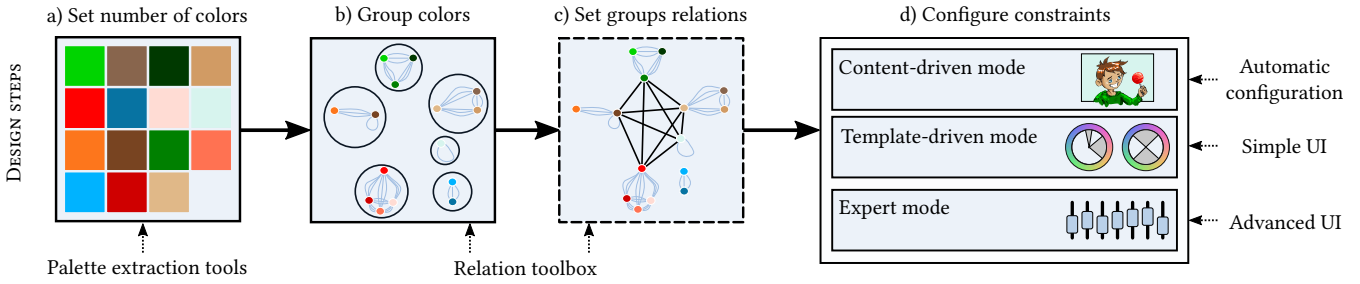



Fig. 6. Overview of the palette design pipeline: a) Select the number of colors within the palette, possibly using palette extraction tools. b) Select groups of colors and assign them a relation type provided by the system (anchor, level of shade, harmony). c) Optionally select additional relations between groups and/or colors. d) Configure palette parameters: colors (when not already provided), constraints reference values and weights.

when $\alpha$ varies in $[0, 1]$, $\Pi^I_{gamut}$ penalizing out-of-gamut colors, and $\Pi^I_{smooth}$ favoring smoother interpolation curves. $\Pi^I$ is thus defined as:

$$\Pi^I = \Pi^I_{constr} + \lambda\Pi^I_{gamut} + \gamma\ \Pi^I_{smooth},$$

where $\lambda$ is set as in Section 4.3 and $\gamma$ limits the influence of the smoothness criterion to ambiguous cases only, i.e. when $\Pi^I_{constr}$ and $\Pi^I_{gamut}$ are small (see supplementary materials for numerical values). The evaluation of constraints deviations is computed as:

$$\Pi^I_{constr} = \sum_{j=1}^{k_E} w_j \int_0^1 \left(e_{\alpha,j}\left(\mathbf{C}_{\alpha,j}\right)\right)^2 \partial\alpha,$$

with constraints $e_{\alpha,j}$ defined for any value $\alpha \in [0, 1]$ as the linear interpolation of the constraints of the interpolated palettes $\mathbf{P}_0$ and $\mathbf{P}_1$:

$$e_{\alpha,j}(\mathbf{C}_{\alpha,j}) = (1-\alpha)e_{0,j}(\mathbf{C}_{\alpha,j}) + \alpha e_{1,j}(\mathbf{C}_{\alpha,j}).$$

The penalization of out-of-gamut colors is computed as the integral of the out-of-gamut penalization $\Pi^P_{gamut}$ (see Equations 4 and 5) for all palettes obtained when $\alpha$ varies in $[0, 1]$:

$$\Pi^I_{gamut} = \sum_{i=1}^{k_c} \int_0^1 g(\mathbf{b}_i(\alpha))\ \partial\alpha.$$

Finally, when the problem defined by the constraints is underconstrained, it might exist an infinity of valid paths connecting the two palettes (see Figure 7). In consequence, we define $\Pi^I_{smooth}$ to prompt the optimization to converge on smooth interpolations closer to linear paths in Lab:

$$\Pi^I_{smooth} = \sum_{i=1}^{k_c} \int_0^1 \Delta\mathbf{b}_i(\alpha)\ \partial\alpha,$$

where $\Delta$ denotes the Laplace-Beltrami operator.

We implement the interpolation paths with Bézier curves. They offer a convenient parametric domain, are easy to differentiate and manipulate with a few control points. We use the Levenberg-Marquardt algorithm for the optimization of the curves' control points. The end-points of our paths are constrained by the interpolated color coordinates ($\mathbf{b}_i(0) = \mathbf{c}_{0,i}$ and $\mathbf{b}_i(1) = \mathbf{c}_{1,i}$), i.e. the first and last Bézier curves control points are directly set and thus, they are not degrees of freedom for the optimization. All the interpolation results shown in this paper are obtained with quartic Bézier curves providing three free control points per path for the optimization. We illustrate how the palette conformity value can oscillate during an interpolation in Figure 3-c, and experimentation with other Bézier curve degrees (up to 12) are shown in supplementary materials.

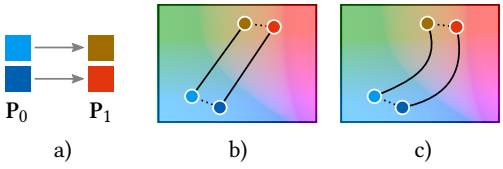



Fig. 7. Example of an underconstrained problem taking benefit from the Laplacian constraint. a) Two input palettes, with a unique contrast constraint $f^c$ between two colors expressed in Lab. b) Interpolation path produced by our approach, which boils down to a linear interpolation naturally preserving the contrast. c) Another solution preserving the contrast, but producing unexpected color variations.

## 6 CONSTRAINED PALETTE DESIGN

The design of a palette is done with the following pipeline, composed of 4 main steps illustrated in Figure 6:

a) setting the number of colors,
b) grouping the colors w.r.t. their relations (which generates constraints between colors),
c) optionally setting the groups relations (which generates constraints between groups of colors),
d) configure the constraints (e.g. reference value and weights) and when required, generate a first set of colors.

Step a) can be performed in several ways. When working with raster images, colors might be extracted using palettes extraction techniques [Chang et al. 2015]. In some other cases (e.g. raster images with indexed colors or vector images), input colors are directly extracted from the media. Alternatively, palettes can be designed from scratch using random colors or manual settings.

We detail in Sections 6.1 and 6.2 the systems required to design the constraint graph (steps b,c) and to set the constraint parameters (step d) respectively. Two examples of systems implementing this palette design pipeline are presented in Section 6.3.

### 6.1 Design system for constraints definition

In this work, we model high-level relations between colors (e.g. levels of shade, color harmonies, or even semantics) as arrangements of constraint functions, represented by the constraint graph. Palette design is usually a hierarchical process: colors are first grouped w.r.t. to their relations (e.g. harmony template). Then, groups of colors can be connected, for instance to preserve contrast between two harmony groups. We identified 4 main types of high-level relations between colors that can be implemented with the following guidelines:

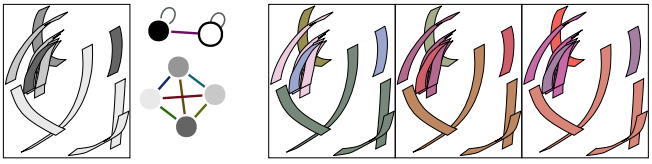
- Color *readability* is preserved by constraining the contrasts perceived between the colors. Specifically, the colors of a group are connected by a complete graph of contrast constraints $e^c_{Lab}$.
- Color *ordering* is often related to shading effects (luminance order [Chang et al. 2015]) and gradients. We preserve ordering with component-wise contrast constraints $e^c_X$ with $X$ the color component associated to the effect (i.e. luminance for shading).
- Colors associated to entities with a specific color *meaning* (e.g. the sky) require unary anchors $e^a_{\mathcal{R}}$ to maintain colors similar to a reference color value or a manifold.
- Color *harmonies* are broadly used for palette design and color compositions. They are represented in our approach by a complete graph associating hue, saturation and value contrast constraints between the considered colors. The number of colors and the desired contrast values define the harmony template (e.g. respectively dual, triadic and analogous, complementary).

In a practical system, high-level relations and groups can be designed either via the UI or by learning the relations from data, for instance using an approach like Probabilistic Color-by-Numbers [Lin et al. 2013]. With this approach, the *color*-based relations learned from data directly translate to our approach as follows: (1) color properties define anchors (lightness as $e^a_{\mathcal{R},\mathrm{L}}$, saturation as $e^a_{\mathcal{R},\mathrm{sat}}$), and (2) pairwise color functions define binary constraints (e.g. perceptual difference as $e^c_{Lab}$, relative saturation as $e^c_{\mathrm{sat}}$). As a result, a constrained palette can be generated automatically from a factor graph trained on a given learning set. The constraint weights can be set directly from the learned weights, and the initial reference values can be initialized from the constraints probability distribution functions (e.g. maximum probability value).

### 6.2 Design system for constraints configuration

At this step, the graph topology is set; colors, constraint reference values $\mathrm{f}_{ref}$ and weights $w_i$ remain to be defined. Colors and reference values are tightly related, each influencing the others. Consequently, we propose two scenarios: the computation of the reference values from input colors, and the estimation of colors from reference values.

- *Reference values from colors*: Using Equation 1, we compute the reference values as $\mathrm{f}_{ref} = f(\mathbf{C}_j)$ to obtain a conformity equal to zero.



Input image + palette — Stochastic search output

Fig. 8. Color estimation from a constraint graph with known topology and reference values using stochastic search. Input gray-scales illustrate the areas with different labels in the image.

- *Colors from reference values*: Unknown colors are obtained by stochastic search, i.e. initial colors are randomly generated and then optimized to minimize the palette conformity (see Figure 8). This scenario usually occurs for palettes with small cardinality, for instance following harmony templates.

When neither colors nor reference values are known, references can be initialized with a default template (e.g. analogous or complementary colors). Then, the values can be refined during the exploration process, as illustrated in Figure 11.

Without prior knowledge, constraint weights are uniformly set as $w_i = 1/k_E$. When the application context is known, different heuristics might also be applied. For instance, in case of images, we define the importance of a color as the relative coverage of its associated regions in the image, and set its constraint weights as the product of their color importance.

From the user perspective, these different initialization procedures can be classified in three design modes:

- *Content-driven mode.* Assuming an input media defining the input colors (e.g. color image), the system automatically computes the constraints' reference values and presets the graph for the optimization. This mode is fully automatic and does not require further user action or specific user expertise.
- *Template-driven mode.* In this mode no assumption is made on the input media. It is rather done on the color relations: the system needs to provide a catalog of color relations (e.g. level of shades, color harmonies) and expose parameters to the user (e.g. luminance spacing, template type). This mode follows the idea of existing palette design tools such as Paletton [Staníček 2015] and Adobe Kuler [Adobe 2015] (see Section 6.3 for practical use cases).
- *Expert mode.* In this mode both colors and constraints' references are unknown. Two main approaches can be considered: in the first, the user is able to provide the full set of reference values (e.g. desired contrasts), from which the system automatically infers a first set of corresponding colors using stochastic search. In the second, colors and/or references are set using heuristics (random, uniform, preset), and the user refines the parameters by exploration.

### 6.3 Use cases

According to the recent study by Jalal et al. [2015], users tend to manipulate "*relationships among* [the palette colors]*, but usually*

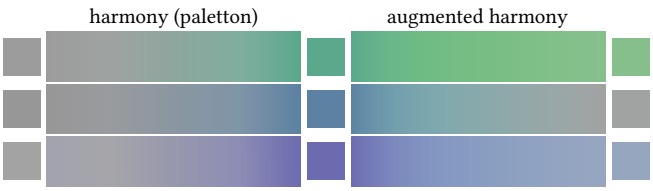



Fig. 9. Center: input palette. Left: desaturation of a palette with a color harmony (triadic, analogous) defined and edited in Paletton. Right: desaturation of a palette with an augmented harmony using our approach, by desaturing the central color. Bottom: hue distribution of input and output images, with palette colors shown as vertical bars.

*in an ad hoc way, combining features from multiple tools or creating their own techniques*". This section presents two practical systems with which users can easily create and modify color palettes while manipulating both the colors and their relations. The systems are tailored for color harmony relations, one extending interactive palette design tools [Adobe 2015] (Section 6.3.1), and the other following the workflow introduced by Cohen-Or et al. [2006] (Section 6.3.2).

*6.3.1 Harmony-based interactive palette design.* Interactive color palette design tools such as Paletton [Staníček 2015] and Kuler [Adobe 2015] provide catalogs of harmony templates. In this catalog, the user selects a type of harmony (e.g. analogous, complementary) and a number of colors. Starting from colors initialized by the system, the user manipulates the palette by changing one of its colors at a time, and the palette parameters when available (e.g. the angular distance between complementary colors). In real-time, the system updates the other colors to conform the palette to the selected template. The main limitation of these systems is to be restricted to harmony templates expressed as geometric configurations. Thus, they cannot generate palettes corresponding to compromises expressed by heterogeneous constraints.

This exploration metaphor directly translates in our formalism: the catalog of templates correspond to predefined palettes composed of a unique color *harmony* group (see Section 6.1 for the definition of its graph). From the user perspective, there is no visible change between the two approaches: starting from a template, the user edits the palette colors and some template parameters, and the result is provided in real-time.

In addition to providing the same functionalities as existing systems, our optimization-based exploration opens exciting opportunities to improve harmony-based palette design. For instance, a known limitation of harmony templates is the generation of unexpected contrast variations during the exploration. With our representation, we can model the perceptual distance between colors and find the compromise between harmony and contrast preservation. This is done by adding a contrast constraint $e^c_{Lab}$ between the colors. The advantage of this representation is to produce results comparable to harmonies in most of the cases, while preserving the readability in extreme cases, as shown in Figure 9 for a strong desaturation of a palette.

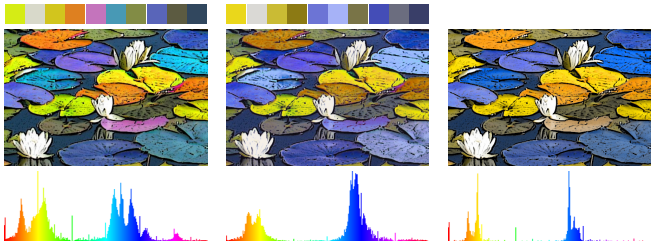
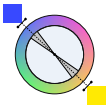



Fig. 10. Left: input image and its palette extracted by Chang et al. [2015]. Center (our): palette obtained by penalizing deviations to a target harmony template (complementary colors). The final image is obtained from the new palette and using Chang et al.'s recomposition method. Right: harmonization result from Cohen-Or et al. [2006]. Image courtesy Cohen-Or et al. [2006].

*6.3.2 Color harmonization.* This system takes as input a raster image and harmonizes its palette colors, following the ideas presented in the seminal work by Cohen-Or et al. [2006]. In practice, the system works as follows: the user composes the image palette using the palette extraction tool introduced by Chang et al. [2015]. Then, she provides a harmony template defined by a set of colors and tolerance values along the hue axis. The right inset shows an example for the analogous template used in Figure 10. From an input template, the system automatically generates constraints as follows (see Figure 7 in supplementary materials):



(1) Colors are constrained by anchors penalizing deviation in hue from the template colors ($f^a_{\mathcal{T}}$ with $\mathcal{T}$ the template colors).
(2) Colors are constrained by anchors in saturation and value.
(3) A complete graph of Lab contrast connects all the colors.
(4) A global constraint is added to preserve the palette luminance. The constraint parameters (reference values and weights) are automatically computed from the image: contrast values are set from the initial contrasts, and the target anchor distances are set to 0.

As the anchors penalize the deviation from the template colors, the conformity of the initial palette is unlikely to be 0. Running our optimization provides a compromise between the palette color contrasts and distances to the harmony template. There is no explicit assignment between the palette colors and the individual template colors: our anchor formulation automatically choose the reference color that minimizes the distance.

Optimization results are illustrated in Figure 10. Our approach operates on the palette and not directly on the pixel colors as in the original approach, which explains the differences in the resulting images and histograms. We link the pixel and the palette colors with Chang et al. [2015] image recolorization process. This produces wider picks in the hue histogram, while maintaining local contrasts in the image.

| Scene | $k_c$ | #Constraints | | | Timing | #Iter. |
|---|---|---|---|---|---|---|
| | | Unary | Binary | Global | (ms) | |
| Teaser | 13 | 4 | 52 | - | 15 | 26 |
| Small Map | 6 | 6 | 15 | 1 | 9 | 81 |
| Harmonization | 10 | 30 | 15 | 1 | 25 | 25 |
| Task 1 | 3 | - | 12 | - | $< 1$ | 15 |
| Task 2 | 6 | 2 | 7 | - | 8 | 76 |
| CVD Test | 7 | 2 | 27 | - | 17 | 13 |
| Basket | 9 | 1 | 28 | - | 10 | 20 |
| Cell | 55 | - | 1485 | - | 3000 | 40 |

Table 1. Statistics for the interactive exploration of palettes. The palette size $k_c$ counts only the optimized colors We computed the mean time required to optimize the palette (over 1000 runs) during an interactive exploration session.

| Scene | $k_c$ | #Constraints | | | Timing | #Iter. |
|---|---|---|---|---|---|---|
| | | Unary | Binary | Global | (ms) | |
| Interpo. Test (Fig. 3) | 2 | - | 1 | 1 | 170 | 110 |
| Small Map (Fig. 4) | 6 | 6 | 15 | 1 | 900 | 38 |
| Task 2 | 6 | 2 | 7 | - | 65 | 5 |
| Small Map (Fig. 15) | 6 | - | 15 | 2 | 4302 | 147 |

Table 2. Statistics for the interpolation-based exploration of palettes. Timings are reported for paths optimization only.

## 7 RESULTS

### 7.1 Implementation and timings

We use the Levenberg-Marquardt implementation provided by the Eigen library [Guennebaud et al. 2010], and we employ numerical differentiation to compute the Jacobian matrix of the conformity error. We report in Tables 1 and 2 the timings required to solve Equation 6 and Equation 7 respectively, for all the scenes shown in the paper and in supplementary material. All our experiments were done on single core, with a processor Intel(R) Core(TM) i7-950, with 12GB DDR3 memory.

Our first exploration scenario takes as input a palette and optimizes it to reduce its conformity. The optimization is performed interactively when the user edits a color in the interface. The edited color is considered as locked, and so excluded from the optimization variables to conform to the user edit. Any color of the palette can be locked in the interface, so that it is not optimized but its constraints are still considered (except unary constraints). In most of the cases (see Table 1), the optimization is done in less than 40 milliseconds, enabling real time updates of the palette w.r.t. to user inputs. As a result, our approach requires a competitive time budget against existing interactive tools, while being able to handle complex constraints and large palettes.

The interpolation-based exploration of palettes requires to solve a much larger optimization problem, and requires more computation time. However, our implementation provides an interactive feedback to the user for most scenes, by estimating the paths in seconds, depending on the palette complexity. Note that the timings do not include the computation of intermediate palettes once the paths are known, since the evaluation of few Bézier curves requires a negligible amount of time.

### 7.2 Exploration

This section presents several use cases of our approach, with the description of the hypergraph setup.

*7.2.1 Semantic-aware palette design.* The palette of the drawing shown in Figure 1 is composed of 19 colors with strong relations, e.g. contrasts, color shades, semantic meaning. The palette setup is illustrated in the accompanying video: we first start by identifying 7 color groups containing from one to four colors. We consider two groups as static (e.g. gray-scales, eye colors) and do not consider them for the exploration. The second group has only one color, the background, which we constrain to pastel colors using two anchors $e^c_L$ and $e^c_{sat}$. Each of the five other groups represents a shade with an almost uniform hue: we encode these relations using complete graphs with $e^c_L$, $e^c_a$ and $e^c_b$. Groups corresponding to hair and skin have a strong color meaning: we constrain them with anchors to restrict colors w.r.t. to tone reference charts. We ensure the readability of the palette by connecting the different groups with a complete graph of clamped contrasts between each group representative.

During the exploration, when the user moves one color, the other colors of its group follow by keeping an almost rigid structure in Lab, and thus preserve the ordering in luminance and the differences in colors. When the representative colors of two groups come too close, the group moved by the user *pushes* the other so that the clamped contrast is still satisfied. We demonstrate in the additional video how the exploration of colors with semantic meaning is affected by the lack of unary anchors, and the differences between single color and color chart anchors.

*7.2.2 Editing constraints parameters.* We illustrate in Figure 11 the exploration of palettes by editing a constraint reference $f_{ref}$. The hypergraph setup defines two shade groups (one by characters), and fixed colors for the lips, tongue, black and white. The other colors are set to ensure contrast between each others and with the two shade groups. While the principal color of left character is locked, the user controls the expected contrast with principal color of the other character. When the contrast is reduced to zero, the two

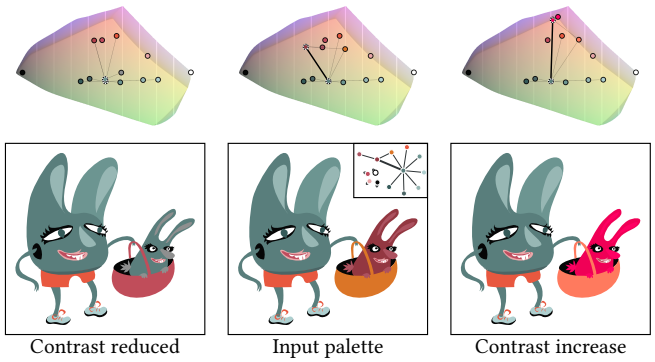



Fig. 11. Contrast editing between the two characters colors. Middle: initial setup, left: contrast reduced, right: contrast increased. The first row illustrates the palette hypergraph in Lab space, with the edited constraint emphasized. Image courtesy Kartini Thomas.

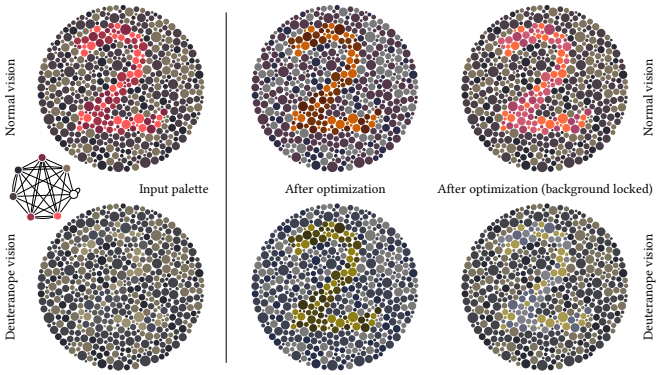



Fig. 12. Left: original, middle: after optimization, right: after optimization with disk colors locked.

characters have merely the same color, while when the contrast is increased, the color of the second character reaches gamut limits to maximize the color difference between the two characters. The color of the basket changes during the exploration to preserve contrasts with the colors of the second character.

*7.2.3 Contrast enforcement for CVD.* We illustrate our approach capabilities for CVD-aware palette editing by optimizing a CVD test image to make it readable by people with CVD (Figure 12).

*7.2.4 Large palettes.* We illustrate in Figure 13 the exploration of a palette with large cardinality (55 colors including all color shades) using a complete graph of contrast constraints ($e^c_{Lab}$, see graph topology in supplementary materials). This simple graph generation provides limited exploration capabilities compared to a fine palette tuning. It is however simple to generate, and can be further edited if required.

*7.2.5 Photo editing.* We also combined our approach with the recent work by Chang et al. [2015] on Palette-Based Photo Recoloring. We use their approach to extract a palette from an input

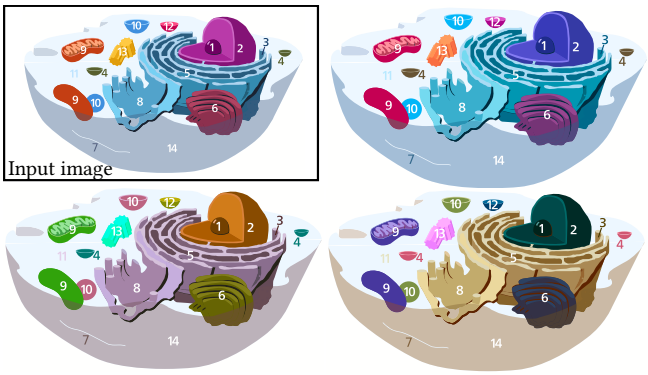



Fig. 13. Image samples from an interactive exploration session. The palette constraint has been automatically computed from the input colors for contrast preservation (white is not optimized). The hypergraph has 55 colors and 1485 pairwise $e^c_{Lab}$ constraints. Image courtesy Kelvinsong.

Fig. 14. Left: input image and its palette extracted with the method proposed by Chang et al. [2015] . Right: palette obtained after modifying the buildings and the sky colors (black borders) using our approach. The final image is recomposed with the method proposed by Chang et al. [2015]. Image courtesy Bernard Spragg NZ.

image, from which we design a constraint graph. Here we want to preserve the image coherence when the user modifies the palette colors. For the example shown in Figure 14, we first grouped the colors corresponding to level of shades (3 groups), and constrained their colors by component-wise contrast constraints in L, a and b (see Section 6.1). Then, we constrained the groups to preserve readability using contrast edges ($e^c_{Lab}$) between colors. We also add anchors to the sky and the window color (dark brown) to convey lighting semantics, and a global constraint to preserve the luminance of the photo. The palette generated by our approach is then given as input to Palette-Based Photo Recoloring to generate the final output (we disabled the luminance correction originally applied on the input palette).

## 7.3 Interpolation

We consider in a first example (see Figure 15) the case where we need palettes to be used in a GPS with an OLED screen, such that the palettes can adapt to varying display energy requirements. We start by a palette applied on a simple map, and modify it using our interactive exploration technique by lowering its display energy consumption. The interpolation between the two palettes (lower and higher energy requirements) ensures contrast preservation for all intermediate palettes.

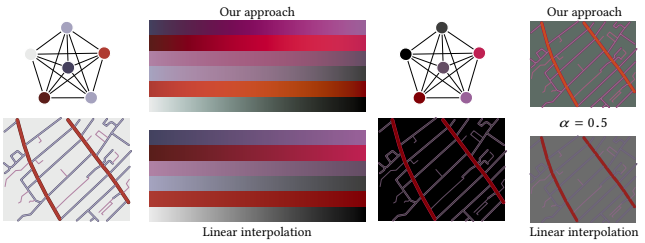



Fig. 15. Interpolation between an original palette (left) and low-energy consumption counterpart, obtained by exploration (see accompanying video). The constraint graph encodes full contrasts, global luminance and energy consumption interpolation.

Figure 16 shows interpolation results between the reference topographic color map and a Pop art map. Results generated by our

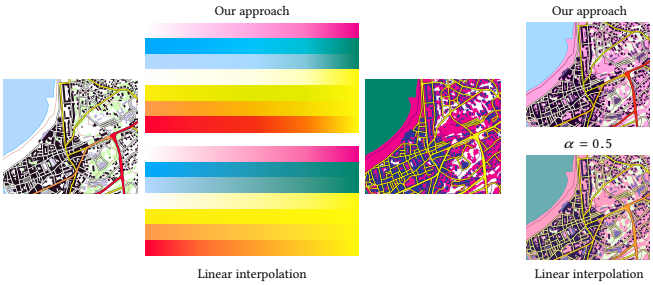



Fig. 16. Interpolation of colors between topographic and Pop art palettes. Top-row: linear interpolation; Bottom row: Cartographic constraint-based optimized interpolation, using anchored colors, hue and value contrasts. Notice how our approach produces a more readable map, with increased contrast e.g. on the road network.

approach respect the cartographic constraints from the first input, while mixing the colors from the second input.

Finally, these interpolation-based explorations show the expressiveness of our modeled constraints, which would allow a mapmaker to subtly mix and refine salient colors in both map endings.

## 8 USER STUDY

We performed a user study to evaluate our palette exploration approaches. We recruited 30 people, all with normal vision (see statistics in additional materials). We divided the study in two tasks, a first task on the exploration of an analogous triad under contrast constraints for normal and impaired vision, and a second task on the interpolation between two palettes.

### 8.1 Task 1: Augmented harmony and color blindness

*8.1.1 Description.* The goal of this task is to measure how our approach helps users to find a trade-off between constraints. Users were asked to modify an existing palette to reach a specified target color tone, while preserving simultaneously three goals: (1) contrasts for people with normal vision, (2) contrasts for people with deuteranope vision, (3) color harmony. No formal definition of these quantities was given to the user, however we provided real-time feedback during the exploration to help them evaluating their design (see supplementary materials for more details).

*8.1.2 Scene setups.* We designed three different constrained palettes, sharing the same graph topology (shown in the right inset) but with different constraints between the colors. Colors have been generated using a triadic analogous harmony template with the online tool Paletton[1]. We design the palettes from these colors as follows:

(a) Harmony template: each pair of colors is constrained by contrast constraints $e^c_{\text{hue}}$, $e^c_{\text{sat}}$ and $e^c_{\text{val}}$, respectively on their hue, saturation and value properties. This template aims at reproducing the expected behavior of common harmony exploration tools, and is considered as a baseline for comparisons.

(b) Extended harmony template: extends (a) by adding a contrast constraint $e^c_{Lab}$.

(c) CVD-aware extended harmony template: extends (b) by adding a contrast constraint $e^c_{Lab}$ on the colors after CVD simulation.

These palettes are designed to preserve the harmony (a,b,c), contrasts for normal vision (b,c) and deuteranope vision (c).

*8.1.3 Task.* We asked the users to modify the palette to obtain one of the two colors shown in the right inset. This task was repeated 6 times in a random order, so that the user explored with the three templates (a-c) for the two target colors.

*8.1.4 Results and discussion.* Explorations are evaluated by reporting the residuals of the harmony and contrasts (normal vision and deuteranope) constraints, defined as:

$$e_{\text{h}} = \sum_{i=1}^{3}\sum_{j=i+1}^{3} |e^c_{\text{hue}}(\mathbf{c}_i,\mathbf{c}_j)| + |e^c_{\text{sat}}(\mathbf{c}_i,\mathbf{c}_j)| + |e^c_{\text{val}}(\mathbf{c}_i,\mathbf{c}_j)|$$

$$e_{\text{c}} = \sum_{i=1}^{3}\sum_{j=i+1}^{3} |e^c_{\text{Lab}}(\mathbf{c}_i,\mathbf{c}_j)|$$

$$e_{\text{deut}} = \sum_{i=1}^{3}\sum_{j=i+1}^{3} |e^c_{\text{Lab}}(p_{\text{deut}}(\mathbf{c}_i),p_{\text{deut}}(\mathbf{c}_j))|$$

Final mean errors are reported in table 3 (see error distributions in supplementary materials). We measure how well each user reached the target color based on her own appreciation, using a rating out of five. No significant difference has been found between the scenes (mean satisfaction rate: 3.5).

| Palette | Mean $e_{\text{h}}$ | Mean $e_{\text{c}}$ | Mean $e_{\text{deut}}$ |
|---|---|---|---|
| (a) | 0.00 | 5.87 | 10.96 |
| (b) | 2.34 | 2.95 | 8.22 |
| (c) | 4.70 | 2.26 | 4.75 |

Table 3. Mean errors values for the harmony and contrasts (for normal and deuteranope visions).

For the harmony template (a), the palette definition ensures the generation of color sets that all perfectly meet the harmony criterion. On the other hand, due to the non-linear mapping between the HSV and Lab spaces, users mostly fail at preserving contrasts between the colors. The introduction of a contrast constraint with the augmented harmony template (b) gives more chances to the user to obtain better contrasts, and in practice divide the mean contrast error by 2. A similar pattern is observable for the contrast preservation for deuteranope vision, with the introduction of the associated constraint in the CVD-aware extended harmony template. In the case of template (b) and (c), the contrast improvement is obtained at the cost of an increase of the harmony error. For template (c), the palette constraints encode the three measures the user has to minimize, so the trade-off is performed by the system while the user could focus on the target color tone.

In summary, this study confirms the ability of our approach to represent and find palettes compromising between different constraints. With template (a), some user explored and found a palette that reached a good trade-off between the three measures, while most of them gave up with a relatively high residual error. From

[1] See http://paletton.com/#uid=53z0t0keGvq5jR69VG8jjqCnvm8

the user point of view, our approach hides the complexity of the constraints without limiting the exploration experience.

## 8.2 Task 2: Interpolation

*8.2.1 Description.* The goal of this task is to evaluate if palettes generated by interpolation are perceived as such by users.

*8.2.2 Task.* The study interface presents 3 palettes applied side-by-side on the same vector image, with from left to right: the *source*, the candidate and the *target* palettes. The 12 candidates are shown sequentially to each user with randomized order. No information is given to the users about the constraints or the method used to generate the images. Instructions are given as follow: *Considering two images, called* source *and* target, *please indicate if a third image is a good intermediate between the* source *and* target *images. This comparison must be made only considering the image colors. It is up to you to decide, whether an image is a good intermediate, or if it is either too different or too similar to input palettes, considering the two examples* (shown in Figure 17).

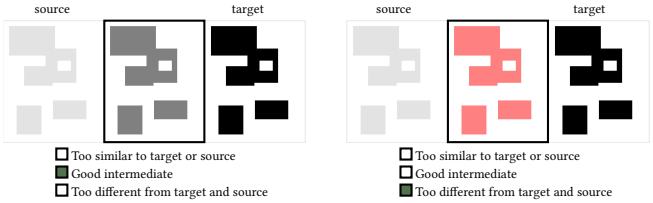



Fig. 17. Examples given to the users before the study to help them defining the notion of intermediate image.

*8.2.3 Images generation.* The *source* palette is composed of a group of 4 contrasted colors, i.e. constrained by a complete contrast graph with $e^c_{Lab}$ pairwise constraints. The background (white) and shape borders (black) colors are kept in place using color anchors $e^a_{\mathbf{c}_{ref}}$ and a pairwise contrast distance $e^c$. We designed the *target* palette by exploration from the *source* palette.

We generate a set of twelve palettes (see Figure 18) using 4 different methods:

- **Our**: palettes obtained along the interpolation paths ($\alpha = \{0.25, 0.5, 0.75\}$) optimized with our approach between *source* and *target*,
- Linear: palettes obtained by linear interpolation between *source* and *target* ($\alpha = \{0.25, 0.5, 0.75\}$),
- Stochastic: palettes obtained by stochastic search from the *source* palette, with low residual error,
- Stochastic_bad: palettes obtained by stochastic search from the *source* palette, with high residual error.

*8.2.4 Results and discussion.* Figure 19 presents the repartition of the labels assigned by the users to the different palettes. We propose to analyze the results by label:

**Too different** As expected, this label is mostly assigned to the stochastic and stochastic_bad palettes, which are generated from the input *source* image only. Also notice that it is also assigned to 43% of the linear palette ($\alpha = 0.5$), probably due to the contrast loss at this intermediate interpolation step.

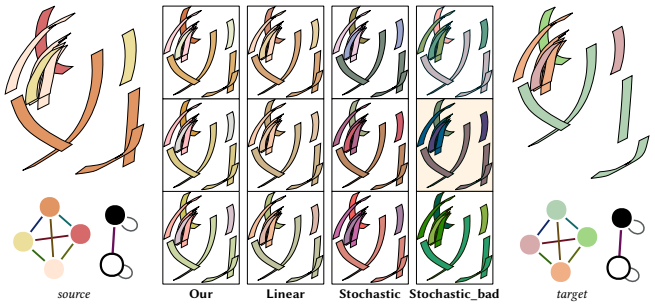



Fig. 18. *Target*, *source* and *candidate* palettes used for the study, applied on a vector image. For Our and Linear, the interpolation parameter $\alpha$ is set from top to bottom at 0.25, 0.5 and 0.75.

**Too similar** Unsurprisingly, this label is mostly assigned to the palettes generated by the interpolation methods (our, linear) for $\alpha = \{0.25, 0.75\}$. This confirms the ability of interpolation methods to produce smooth transitions to input palettes.

**Good intermediate** Palettes obtained by our method are clearly the most recognized as good intermediates. We also observe a notable difference between our approach (70%) against linear interpolation (36%) when $\alpha = 0.5$.

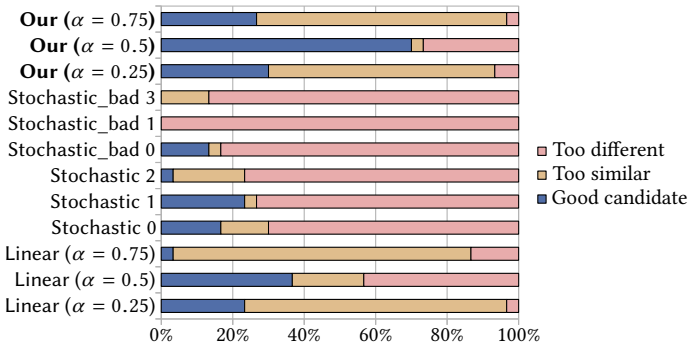



Fig. 19. Distribution of the labels assigned by the users.

## 9 LIMITATIONS

*Scalability:* In our representation, the larger the hypergraph, the lower the influence of each constraint. Hence, moving a color in a large graph may have a small influence on the optimization process, which would reduce the user control on the exploration. A solution could be the use of meta-vertices in the graph to group colors and their relations in a multi-level fashion. A similar hierarchical approach has been used successfully for grouping geometric relations between 3D primitives [Monszpart et al. 2015].

*Numerical constraints:* Our constraints are defined as real-valued cost functions. While very effective for representing and optimizing continuous constraints, it prevents the inclusion of labels and Boolean relations in our palettes. A solution could be to represent such constraints as probability distribution functions, which can

be directly inserted in constrained palettes. Another would be to optimize the conformity and labels with mixed-integer programing.

*Extreme contrasts:* In Lab space, if a constrast constraint is set between colors with very strong constrasts (e.g. with highly saturated colors and different hues), the color variation space maintaining a low conformity will be restricted by the gamut boundaries. During exploration, this may produce very high conformity errors that would prevent the optimization to provide adequate color solutions.

## 10 CONCLUSION

We present a new definition of constrained palettes based on an hypergraph representation. This definition enables interactive optimization and is well suited for palette space explorations. We present two exploration strategies whose efficiency and robustness are illustrated on real world use cases, such as artistic/cartographic palette design, and photo recoloring. In addition to contrast constraints, our approach naturally handles context-based constraints, such as Color Vision Deficiencies simulation and energy consumption optimization.

Our work opens several promising directions of investigation. For instance, multiple-palettes interpolation would enable the generation of palettes manifolds, following the same spirit of color manifolds [Nguyen et al. 2015]. We also believe that our approach can be applied to other parametric spaces than digital color: we plan to study the exploration of parametrized appearance properties, such as hatching patterns and procedural textures. Finally, we would like to consider the human perception system with higher accuracy in our constraint definitions and optimizations.

## REFERENCES


Adobe. 2015. Kuler. https://color.adobe.com/. (2015). accessed 25/08/2015.

Jacques Bertin. 1983. *Semiology of Graphics: Diagrams, Networks, Maps.* University of Wisconsin Press, Madison.

Cynthia A Brewer. 1994. Color use guidelines for mapping and visualization. *Visualization in modern cartography* 2 (1994), 123–148.

Cynthia A. Brewer. 2015. Color Brewer. http://www.ColorBrewer.org. (2015). accessed 25/08/2015.

Elodie Buard and Anne Ruas. 2009. Processes for improving the colours of topographic maps in the context of Map-on-Demand. In *24th International Cartographic Conference (ICC'09), Santiago de Chile, Chile.*

Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. 2015. Palette-based Photo Recoloring. *ACM Trans. Graph.* 34, 4, Article 139 (July 2015), 11 pages. DOI: https://doi.org/10.1145/2766978

Haidong Chen, Ji Wang, Weifeng Chen, Huamin Qu, and Wei Chen. 2014. An image-space energy-saving visualization scheme for OLED displays. *Computers & Graphics* 38 (2014), 61 – 68. DOI: https://doi.org/10.1016/j.cag.2013.10.020

Sidonie Christophe. 2011. Creative colours specification based on knowledge. *The Cartographic Journal* 48, 2 (May 2011), 138–145.

Sidonie Christophe, Christine Zanin, and Hugo Roussaffa. 2011. Colours harmony in cartography.. In *25th International Cartographic Conference (ICC'11), Paris, France.*

Johnson Chuang, Daniel Weiskopf, and Torsten Möller. 2009. Energy Aware Color Sets. *Computer Graphics Forum* 28, 2 (2009), 203–211. DOI: https://doi.org/10.1111/j.1467-8659.2009.01359.x

Daniel Cohen-Or, Olga Sorkine, Ran Gal, Tommer Leyvand, and Ying-Qing Xu. 2006. Color Harmonization. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* 25, 3 (2006), 624–630.

David R. Flatla, Katharina Reinecke, Carl Gutwin, and Krzysztof Z. Gajos. 2013. SPRWeb: Preserving Subjective Responses to Website Colour Schemes Through Automatic Recolouring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13).* ACM, New York, NY, USA, 2069–2078. DOI: https://doi.org/10.1145/2470654.2481283

Gaël Guennebaud, Benoît Jacob, and others. 2010. Eigen v3. http://eigen.tuxfamily.org. (2010).

Charlotte Hoarau. 2011. Reaching a Compromise between Contextual Constraints and Cartographic Rules: Application to Sustainable Maps. *Cartography and Geographic Information Society Journal* 38, 2 (April 2011), 79–88.

Manabu Ichikawa, Kiyoshi Tanaka, Shoji Kondo, Koji Hiroshima, Kazuo Ichikawa, Shoko Tanabe, and Kiichiro Fukami. 2003. Web-page Color Modification for Barrier-free Color Vision with Genetic Algorithm. In *Proceedings of the 2003 International Conference on Genetic and Evolutionary Computation: PartII (GECCO'03).* Springer-Verlag, Berlin, Heidelberg, 2134–2146. http://dl.acm.org/citation.cfm?id=1756582.1756698

J. Itten. 1974. *The Art of Color: The Subjective Experience and Objective Rationale of Color.* Wiley. https://books.google.fr/books?id=D-skaDZAumIC

Mathieu Jacomy. 2015. I Want Hue. http://tools.medialab.sciences-po.fr/iwanthue/. (2015). accessed 25/08/2015.

Anil K. Jain. 1989. *Fundamentals of Digital Image Processing.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Ghita Jalal, Nolwenn Maudet, and Wendy E. Mackay. 2015. Color Portraits: From Color Picking to Interacting with Color. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems.* ACM, New York, USA, 4207–4216. DOI: https://doi.org/10.1145/2702123.2702173

Laurent Jégou. 2014. Color Gradient Explorer. http://www.geotests.net/couleurs/gradients_inflex_en.html. (2014). accessed 30/11/2015.

Sharon Lin, Daniel Ritchie, Matthew Fisher, and Pat Hanrahan. 2013. Probabilistic Color-by-numbers: Suggesting Pattern Colorizations Using Factor Graphs. *ACM Trans. Graph.* 32, 4, Article 37 (July 2013), 12 pages. DOI: https://doi.org/10.1145/2461912.2461988

B.J. Meier, A.M. Spalter, and D.B. Karelitz. 2004. Interactive color palette tools. *Computer Graphics and Applications, IEEE* 24, 3 (May 2004), 64–72. DOI: https://doi.org/10.1109/MCG.2004.1297012

Aron Monszpart, Nicolas Mellado, Gabriel Brostow, and Niloy Mitra. 2015. RAPter: Rebuilding Man-made Scenes with Regular Arrangements of Planes. *ACM SIGGRAPH 2015* (2015).

A.H. Munsell and F. Birren. 1973. *A Grammar of Color: A Basic Treatise on the Color.* Van Nostrand. https://books.google.fr/books?id=s1NSmgEACAAJ

Naila Murray, Sandra Skaff, Luca Marchesotti, and Florent Perronnin. 2012. Toward automatic and flexible concept transfer. *Computers & Graphics* 36, 6 (2012), 622 – 634. DOI: https://doi.org/10.1016/j.cag.2012.01.008 2011 Joint Symposium on Computational Aesthetics, Non-Photorealistic Animation and Rendering, and Sketch-Based Interfaces and Modeling.

Chuong H. Nguyen, Tobias Ritschel, and Hans-Peter Seidel. 2015. Data-Driven Color Manifolds. *ACM Trans. Graph.* 34, 2, Article 20 (2015), 9 pages. DOI: https://doi.org/10.1145/2699645

Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2011. Color Compatibility From Large Datasets. *ACM Transactions on Graphics* 30, 4 (2011).

Li-Chen Ou, M. Ronnier Luo, Andrée Woodcock, and Angela Wright. 2004. A study of colour emotion and colour preference. Part I: Colour emotions for single colours. *Color Research & Application* 29, 3 (2004), 232–240. DOI: https://doi.org/10.1002/col.20010

K. Rasche, R. Geist, and J. Westall. 2005. Detail preserving reproduction of color images for monochromats and dichromats. *Computer Graphics and Applications, IEEE* 25, 3 (May 2005), 22–30. DOI: https://doi.org/10.1109/MCG.2005.54

Daniel Ritchie, Sharon Lin, Noah D. Goodman, and Pat Hanrahan. 2015. Generating Design Suggestions Under Tight Constraints with Gradient-based Probabilistic Programming. *Comput. Graph. Forum* 34, 2 (May 2015), 515–526. DOI: https://doi.org/10.1111/cgf.12580

Gaurav Sharma. 2002. *Digital Color Imaging Handbook.* CRC Press, Inc., Boca Raton, FL, USA.

Petr Staníček. 2015. Paletton - The Color Scheme Designer. http://paletton.com/. (2015). accessed 25/08/2015.

Luigi Troiano, Cosimo Birtolo, and Maria Miranda. 2008. Adapting Palettes to Color Vision Deficiencies by Genetic Algorithm. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation.* ACM, New York, USA, 1065–1072. DOI: https://doi.org/10.1145/1389095.1389291

J.W. von Goethe. 1971. *Goethe's color theory.* Van Nostrand Reinhold. https://books.google.fr/books?id=Z3YpAQAAMAAJ

Takuto Yanagida, Katsunori Okajima, and Hidenori Mimura. 2015. Color scheme adjustment by fuzzy constraint satisfaction for color vision deficiencies. *Color Research & Application* 40, 5 (2015), 446–464. DOI: https://doi.org/10.1002/col.21913

Lina Zhou, V. Bensal, and Dongsong Zhang. 2014. Color adaptation for improving mobile web accessibility. In *Computer and Information Science, IEEE 13th International Conference on.* 291. DOI: https://doi.org/10.1109/ICIS.2014.6912149

L. Zhou and C.D. Hansen. 2015. A Survey of Colormaps in Visualization. *Visualization and Computer Graphics, IEEE Transactions on* (2015). DOI: https://doi.org/10.1109/TVCG.2015.2489649


Received January 2017; accepted March 2017; final version April 2017