Computational Production of Colour Harmony. Part 1: A Prototype Colour Harmonization Tool

Giovanni Moretti, Paul Lyons,* Stephen Marsland

School of Engineering and Advanced Technology, Massey University, Ag/Hort Building, Private Bag 11-222, Palmerston North, New Zealand

Received 10 October 2010; revised 5 July 2011; accepted 24 August 2011

Abstract: Although web page and computer interface developers generally have little experience in generating effective colour schemes, colour selection appears rarely in user interface design literature, and there are few tools available to assist in appropriate choice of colours. This article describes an algorithmic technique for applying colour harmony rules to the selection of colour schemes for computer interfaces and web pages. Our software implementation of this approach—which we term the Colour Harmoniser—adapts and extends classical colour harmony rules for graphical user interfaces, combining algorithmic techniques and personal taste. A companion article presents the experimental evaluation of the system presented here. Our technique applies a set of rules for colour harmony to specific features of the interface or web page to create abstract colour schemes; the user then modifies the overall colour cast, saturation, and light–dark distribution, producing colourings that are both harmonious and usable. We demonstrate experimentally that the software is relatively simple to use and produces colourings that are well-received by humans. In this article, we define a fitness function that numerically evaluates the colour harmony of a user interface and underpins a genetic algorithm for creating harmonious schemes. We show how abstract, hue-independent, colour schemes may be mapped to real colour schemes, leaving the abstract colour harmony unchanged, but accommodating the developer's personal preferences for overall colouring, light–dark contrast, and saturation. This abstract/concrete separation automates the creation of harmonious schemes and allows unskilled developers to express their aesthetic preferences using simple direct manipulation controls. © 2011 Wiley Periodicals, Inc. Col Res Appl, 00, 000 – 000, 2011; Published online in Wiley Online Library (wileyonlinelibrary.com). DOI 10.1002/col.20736

*Correspondence to: Paul Lyons (e-mail: p.lyons@massey.ac.nz).

Key words: colour harmony; Colour Harmoniser; abstract colour scheme; concrete colour scheme; genetic optimization; colour scheme evaluation

INTRODUCTION

In this article, we will present a set of criteria for a software tool intended to create user interface colour schemes and then introduce a software tool, the "Colour Harmoniser," which applies many of the criteria and largely automates the generation of harmonious and usable colour schemes in computer interfaces. The tool combines algorithmic techniques with the human expression of personal taste.

Our Colour Harmoniser is a research tool that is meant to explore the possibility of automatically producing harmonious and usable user interface colour schemes by modeling the interactions between a colour designer and a client in software. The Colour Harmoniser takes on the role of the colour designer and then a developer or web designer wanting to colour an interface acts as a client. We will present a method of algorithmically evaluating user interface colour schemes and (in a second article) experimental results that demonstrate that the colour schemes prepared by the Colour Harmoniser are generally well-received by independent viewers. This can be seen as validation of our design criteria.

We believe that many different computer users have a need for assistance in the selection of harmonious colours. They include people preparing presentations using slides produced by software such as Microsoft PowerPoint, as well as anybody personalizing blogs and Facebook pages. Of course, professional designers of web pages and graphical user interfaces (GUIs) face the same problems, and yet these topics are seldom covered in standard human-computer interaction (HCI) texts. The five pages (out of 579 content pages) devoted to colour in *Designing the*

User Interface: Strategies for Effective Human–Computer Interaction by Shneiderman, 2009 is not atypical. Currently, the only options commonly available for those wishing to choose colours are the severely limited colour selectors provided by the operating systems and application programming interfaces (APIs), and a selection of "pre-canned" palettes. In this article, we propose a method by which users can be involved in designing a colour scheme for themselves that satisfies their requirements and yet is harmonious.

PREVIOUS WORK

Colour harmony is an intuitively simple idea that can be expressed as "a set of colours that look good when seen together." However, a more precise definition is not easy to find, and there seems to be no consensus on what "colour harmony" actually means. Several authors have proposed definitions, including: "selection of colors that give pleasure" and "grouping of colors to suit some practical use"[1]; "when two or more colours seen in neighbouring areas produce a pleasing effect, they are said to produce a colour harmony"[2] as cited by[3]; "in the visual arts the rules for color relations have been limited to a few suggestions on how to obtain 'harmony', that is, colors that do not clash"[4]; "the 'suitability' of juxtaposed colors,"[5] and "colors seen together to produce a pleasing affective response are said to be in harmony."[6] Evidently, there is no single agreed-upon definition, but the general intent is clear: it is a subjective judgement about how pleasant a set of colours is when viewed as a whole.

Although a definition of colour harmony is problematic, there are many heuristics that have been found to be helpful in choosing sets of harmonious colours. The most widely used are those based on colour wheels[7–10] and templates, including those for selecting complementary colour and split-complementary schemes among others. Users typically specify a key colour and use paths in a colour space to guide the selection of related colours to create palettes. Another common approach is to produce a monochromatic colour scheme comprising various saturations and lightnesses of a single hue.

More sophisticated approaches use information about the intended use of the colour scheme to constrain the resultant colours. Two notable examples are the Colour Image Scale[11] and the "Canon Color Advisor."[12] Both have their origins in experimentally derived associations between textual terms and colours. In a later book, Kobayashi[13] provides sets of palettes categorized by emotional effect. The Canon Color Advisor allowed the user to specify (via drop-down selectors) the intended audience, the occasion, the setting and the style of a colour scheme. These parameters were used to select appropriate palettes that used those colours that had been found via the analysis of experimental data to best match the selected parameters. Further details on colour harmony and methods for producing harmonious colour schemes can be found in the literature.[14,15]

Although these heuristics can be used to create palettes of harmonious colours, they are not, by themselves, sufficient for colouring user interfaces. They do not address the functional aspects of items in a user interface that constrain the colour choices, primarily with respect to the contrast between overlapping elements. In particular, if items of text are to be easily legible, it is important to have sufficient contrast with their background. A related concern is the colouring of overlapping interface elements that if coloured too similarly may be difficult to distinguish from one another.

Several approaches have been proposed to address the needs of interface developers. The simplest are merely guidelines for interface designers. These are not methods of choosing actual colours or schemes but rather lists of points to consider when colouring an interface.[16–19] There are many ways of using the colours from a palette to colour the items in an interface. However, not all combinations are visually pleasing and not all will satisfy the readability and distinguishability constraints, which make this allocation problematic. Kobayashi's palettes[13] give some guidance as they define both a set of colours and the intended relative proportions.

Few theories use the sizes of the objects being coloured when deriving sets of colour. Of those that do, the most well-known is Munsell's Theory of Colour Harmony[1] which has strong experimental support,[20–24] although only in the limited testing environment of pairs of coloured swatches. This is a much simpler environment than a typical user interface, where there is a multiplicity of sizes and colours.

None of these methods of producing harmonious colour sets, even the ones that allow for the sizes of objects, address the functional requirements of user interface colour schemes. Nakakoji's "eMMaC" system[25] takes a different approach. It does not attempt to allocate colours but instead acts as a design assistant by critiquing the designer's choices, without constraining them.

Those systems intended to create complete GUI colour schemes are the most sophisticated and the least common. There are very few implemented systems that can automatically produce sets of colours (palettes) along with bindings of these colours to specific interface elements to yield complete interface colour schemes.

An early example of a method of automating the design of user interface colour schemes was the ACE (A Color Expert) system.[26] It was implemented as a forward-chaining production-rule system using the OPS5 expert-system framework and coloured interfaces based on elements found on the XEROX Star and early models of the Apple Macintosh. The knowledge base contained rules about colouring single items (such as ensuring distinguishability from the background) and colour relationships. The result of using the system is a single colour scheme, containing a colour for each interface element. The scheme is displayed in numeric form that could then be used as the

basis for further development, but such exploration was not part of the ACE system. The system was an innovative attempt to encode the "soft knowledge" of colour scheme design into production rules. In the absence of a general colour harmony assessment function, this assessment is done at a very detailed level, encoding relationships between specific pairs of colours. The schemes produced were stated to be better than random schemes, or those created by naïve programmers, but no results from user trials were reported.

MacIntyre describes a system that was designed to colour multiple related windows in an aesthetically pleasing way, providing usable defaults, allowing the user to specify colouring constraints and customize gradually the results, with the author noting that "typical users make far better critics than [they make] designers."[27] The system is modeled on a dynamic window positioning system that reorganizes windows while the system is being used.[28] To colour a scheme, the user selects two colours and the prototypical colour scheme (represented as a wireframe in a colour space), and the system chooses the remaining colours. It is also possible to modify individual colours or to lock the colour of one or more items and have the system create a new scheme incorporating these fixed colours.

The Cuypers project, which aims to automate the creation of hypermedia presentations from multimedia data for use in a museum of fine art, includes a colour design module.[29,30] The module uses the spatial layout and temporal structure that are defined elsewhere in the system and tries to colour the presentation automatically based on rules and details of the intended user group.

The two preceding approaches are blended in the color rule and font tool (CRAFT) system,[31] which functions as an interactive design assistant. The system uses sets of design rules to constrain the colours available for colouring a window and its text as the user makes choices. The result is a scheme that is selected by the user but constrained so that all choices have to be harmonious.

Kelly provides a different approach to colour scheme design[32] using evolutionary optimization in a "supervised" way, where the user's aesthetic judgment acts as a fitness function. At each stage, sets of colour schemes are created (initially without using a colour harmony model), and shown to the user, who inspects the images and tags the "good" schemes. More schemes are then displayed and tagged, with new schemes being based on the more constrained and visually appealing tagged schemes. This enables a slow evolution toward pleasing colour schemes, without requiring a machine-based colour harmony evaluation function.

OVERVIEW OF THE COLOUR HARMONISER

We begin our description of the Colour Harmoniser with an overview of our approach and then describe the tool in more detail. In defining the characteristics of a tool for creating colour schemes, we have found it helpful to group the criteria into four categories:

- criteria that relate to the characteristics of the interface and scheme to be created;
- criteria that relate to colour scheme creation;
- criteria that relate to users' aesthetic preferences; and
- criteria that specify other desirable characteristics

The structure of the following description is divided into sections that mirror these categories, with the relevant criteria listed at the start of each section.

The overall structure of the Colour Harmoniser is shown in Fig. 1. In the initial phase, the software captures a number of interface characteristics, including the sizes of the interface components and any constraints on colourings. For example, it is important that some pairs of interface components, such as a button and its background, should be visually distinguishable. Some of these characteristics can be captured automatically, while others require user input. During this phase, the interface designer also specifies the type of colour scheme required (complementary, monochromatic, etc.). The tool creates prototype colourings by allocating interface items to atoms—random points on the user's chosen wireframe. We refer to this augmented wireframe as a *colour molecule*, by analogy with the rigid wireframes used by chemists to visualize molecules, which can be repositioned and reoriented without changing the relative positions of the atoms. In our environment, an atom is associated with one or more interface components, and its position in colour space defines the colour of that component or those components. The concepts embodied by the terms *wireframe* and *colour molecule* are similar, and we sometimes use the terms interchangeably. In practice, this does not cause confusion as it is generally clear from the context which interpretation is implied.

Simply allocating colour atoms to random positions on the wireframe does not guarantee the colour harmony of the interface, so in the next phase, the software optimizes their positions using an evolutionary algorithm and a fitness function that evaluates the suitability of a particular arrangement of colour atoms for the interface. Atoms corresponding to larger areas of the interface will usually be moved toward the center of the wireframe, and atoms corresponding to items that must be kept visually distinct will be moved apart, as the optimizer searches for positionings that concurrently satisfy the constraints that we have chosen as being necessary for harmonious colourings (colour balance, readability, and distinguishability) while keeping the colour atoms as close to the wireframe as possible.

The output of this process is a set of possible colour molecules for the interface that match the users' specified requirements and classical colour harmony principles. Each colour molecule represents an abstract colour scheme, in the sense that the position of colour compo-

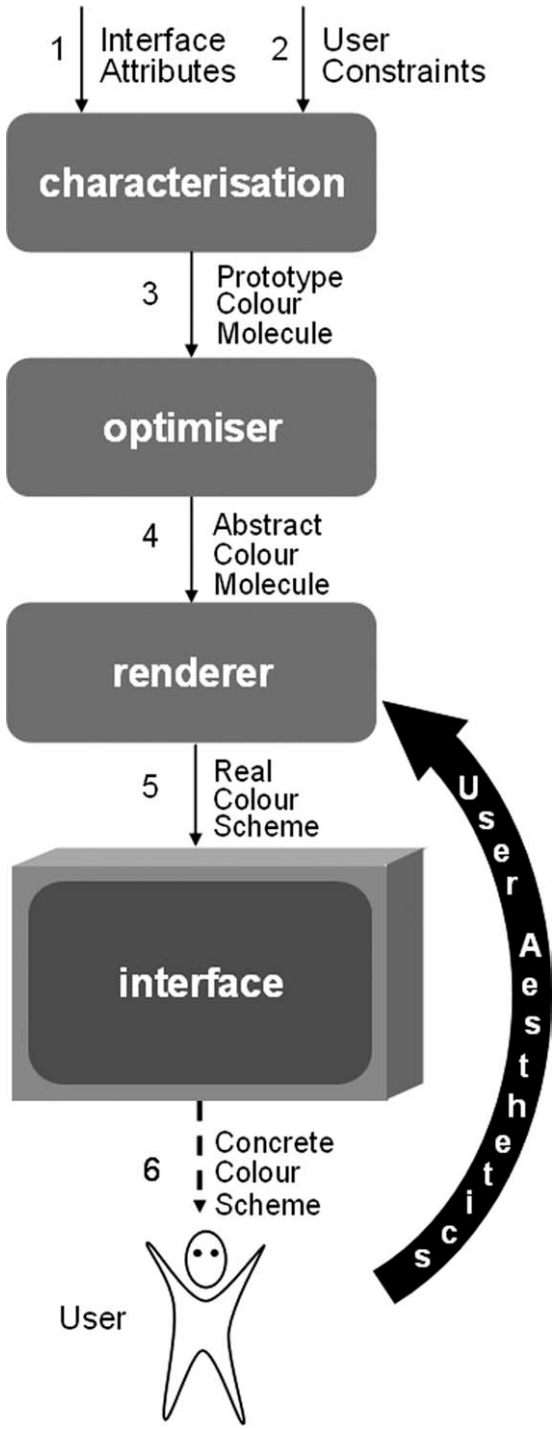

FIG. 1. Colour Harmoniser modules and data flow.

nents is defined with respect to the wireframe, but not with respect to the colour space, as the wireframe's orientation has not yet been fixed in that space. By rotating the wireframe in the colour space, it is possible to produce many real colour schemes with the same relative atom positions, and hence, the same fitness function score. Thus, although the hues of these colour schemes differ, the relative colour positions of the atoms that make them up are constant, and they all have "equivalent" colour harmony.

A colour molecule is only a geometrical configuration of a set of points in colour space. This concept of a set of colour atoms, fixed at various positions on a wireframe, is a useful metaphor for discussing the nature of colour harmony in the abstract, but it is not easy to imagine the visual effect of such a configuration without applying actual colours to an actual image. Therefore, although the wireframes, with their atoms optimized to produce high fitness values, are hue-independent abstract colour molecules, a renderer fixes them at one hue angle, generating real colours for the atoms, and presents the user with a set of real—although still customizable—colour schemes for the interface.

In the final phase of the process, the user can choose any of those colour schemes and, using a simple slider control, rotate its wireframe. This causes the hues in the image to be altered by equal amounts; the colours of the image are updated in real time, as the user moves the hue-rotation control. We believe that this combination of high fitness wireframes with holistic, real-time updating is important, as it allows the user to explore many alternative colour schemes rapidly for the complete interface, without endangering its colour harmony or readability. The user can also invert the colour scheme about the equatorial plane of the space, converting light-coloured interface components into dark ones (or vice versa) without altering their hues.

If the initial set of schemes (defined by a set of colour molecules) is unsatisfactory, the user can either run the optimizer again to find others or check the colour constraints to ensure that all of the important requirements have been included. As will be demonstrated in our follow-up article, the user acceptance of the schemes produced has considerably exceeded our expectations.

The following sections of this article describe the design of the modules shown in Fig. 1.

Interface Characterization Measures the Interface

The information gathered by the Colour Harmoniser in the first phase provides the interface characteristics required by the second phase—the optimizer—to guide the production of a high-fitness colour molecule representation of the colour scheme. As was mentioned in the Introduction section, we have created a set of design criteria that we believe are important for the design of a colour harmonization tool. For the interface characterization, the design criteria are those that relate to the interface characteristics and the type of scheme to be created:

- The tool should have access to the actual interface being designed, and should incorporate a module for automatically measuring the areas of its components, so the created colour scheme can be tailored to that interface, rather than a generic or idealized one.
- The tool should be able to amalgamate coloured areas and colour them as a single entity. In some cases, this may be possible automatically, and in some cases, it may require human intervention.

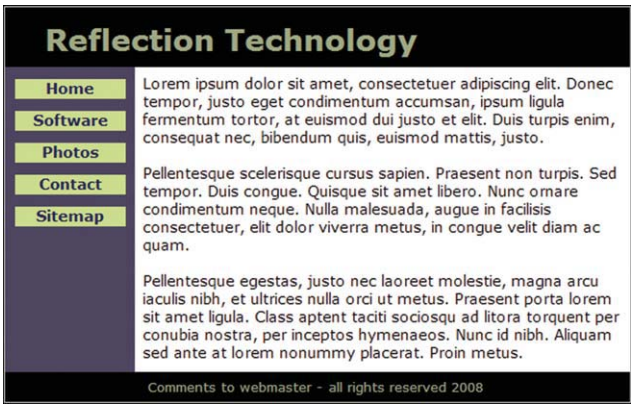

FIG. 2. A typical web page is used to illustrate the effect of the Colour Harmoniser.

- It may therefore be helpful if the tool can make a "best guess" at functional groupings that are to be identically coloured, provided the user can override any such automatic groupings.
- The tool should allow the user to identify pairs of user interface items whose colours should be chosen so that they are distinguishable.

The software needs to know the sizes of the components so that it can take this factor into account when creating a colour scheme. The current implementation of the Colour Harmoniser works with software created in the Delphi IDE (Integrated Development Environment) in which the designer designs an interface by dragging and dropping components. The resultant interface may be programmatically queried to allow the area and position of each interface element to be determined. In other GUI development environments, the layout of interface elements may be determined manually or automatically (by a layout manager), but extracting the same information is unlikely to be problematic.

The following discussion relates to the web page shown in Fig. 2, which incorporates header and footer bars, header and footer text, a sidebar, five buttons, text on each of the buttons, and a main text region with text. Each button is a separate object, and each area of text is a separate object. Although this interface is not complex, its structure is typical of many web sites and the number of coloured elements is sufficient to make the problem of select a harmonious colour scheme, while maintaining readability, a nontrivial one.

In addition to information about the areas of the interface components, the software needs some information about the developer's preferences: which interface components should be the same colour (e.g., all the buttons on a form), and which components must be visually distinguishable (e.g., buttons from their backgrounds). A special case of this second situation is text, whose legibility is compromised if its fine detail does not stand out from its background. The Delphi programming environment provides access to the "type" information of the interface components, allowing them to be grouped and presented to the designer as a tree structure, with items of the same type placed in the same branch. This structure, shown in Fig. 3, panel (A), does not necessarily match the groupings that the designer wishes to enforce, but it is a reasonable first guess, as designers often colour all the buttons or all the panels in the interface identically. The designer can then select some (or all) of those items and drag them to the center panel, creating the tree shown in Fig. 3, panel (B). All items grouped under a common heading in that panel will be given the same colour. It is apparent that, in this case, the designer has retained all the button background colours as a single group, but has separated the textual items into two groups, so that button text and header/footer text are separate. The software will allocate a single colour to each group of items, but will be free to colour each of the text groups independently. Any items that do not need to be colour managed or those that are invisible (those used to group other interface elements) can be left behind in the left panel, with the result that the original tree (A) has been reduced to (C).

Catering to the Designer's Preferences

As mentioned previously, some interface items need to be visually distinguishable. The designer may want banner headers and footers, for example, to be distinguishable from the main portion of a web page, and the buttons to stand out from their background. Identifying pairs of items whose colours must be chosen to ensure that they are distinguishable is not as simple as identifying groups of identically coloured items, so the interface allows the designer to select an item in the tree (B) (e.g., the *Header/Footer text*, in Fig. 3) and specify those items in the duplicate tree in the right-hand panel (D) that should be a visibly different colour from the selected item.

It should be noted that although this interface is functional, it is not ideal. Although it clearly shows which items will be coloured identically (they form a group in the tree), there is no overall view of the distinguishability constraints between items, and the relationship between nodes in the tree and the related interface elements is obscure, although judicious renaming of the elements in the tree can help. A more intuitive alternative has since been designed but has not yet been implemented.

Classical Colour Scheme Prototypes

Once the software has been informed about the interface components, their areas and the colouring constraints that apply to them, the designer can select the type of colour scheme to be used from the complementary, split-complementary, elliptical, monochromatic, and analogous colour schemes. There is then enough information for the software to generate sets of potential colourings. One possible arrangement of elements for each of the complementary and split-complementary schemes is shown in Fig. 4.

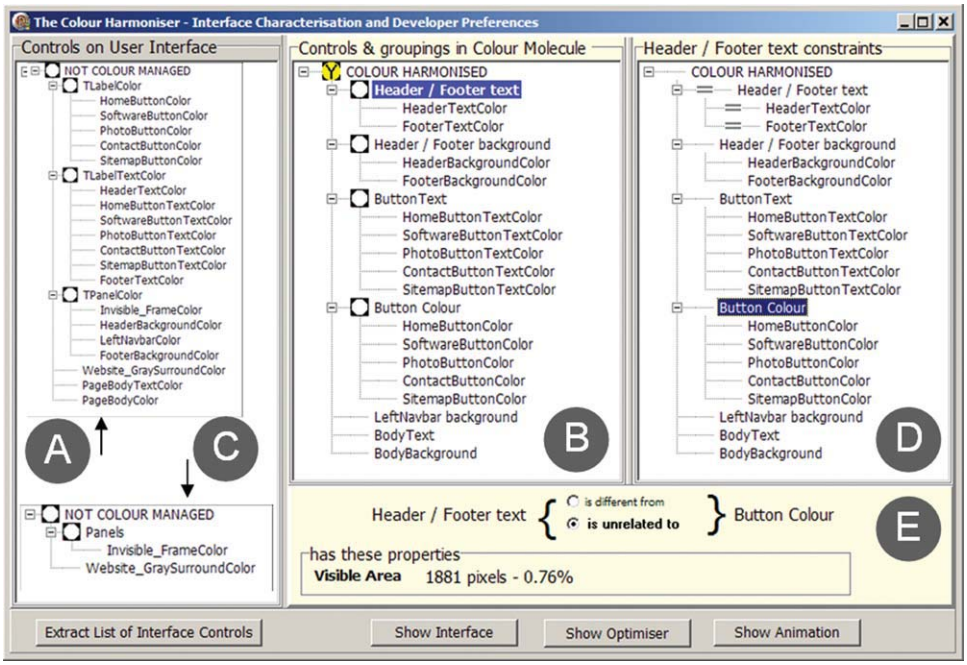

FIG. 3. The designer uses this interface during the characterization phase. Initially, only the tree of interface components labeled (A) is present, having been harvested from the target interface by the Colour Harmoniser. The user drags any items that are to be colour managed (whole branches can be selected) from the tree in (A) into the center pane, creating two partial trees (B) (the items to be colour-managed) and (C) (the remainder, which replaces (A) in the left hand pane). In (B), the designer has reorganized the tree slightly, changing some names and grouping items that should be identically coloured, such as header text and footer text. The Delphi class structure is a useful first guess at the structure; the groups in (A) map quite closely onto the groups in (B). After organizing identically coloured groups in tree (B), the designer works through the tree, choosing parent nodes (groups of identically coloured components) or leaf nodes (individual components) that have to be distinguished from something else. For each such node (Header/Footer text, in this instance), the designer works through the nodes in (D) (which is a copy of the tree in (B)), selecting the interface items from which it needs to be distinguished. The lower panel (E) shows whether or not there is a distinguishability constraint between the selected items in panels B and D.

Blacks, whites, and greys are commonly used in interface colour schemes but these are not always allowed for in conventional rules for colour harmony. To allow these colours to be included, the black–white axis can, if the user desires, be treated as part of the complementary, split-complementary, or elliptical wireframes. This is reminiscent of Itten's colour harmony rule, although when Itten suggested incorporating black or white into a colour scheme, he restricted the remaining colours to the equator of the colour space, where all colours have medium value and maximum saturation. We have not imposed any such restriction.

Abstract Colour Scheme Creation

Once the interface components have been analyzed and the user-specified constraints and the desired colour scheme has been specified, the software generates a set of prototypical colour schemes that define the relationships between the colour elements. The design criteria that underlie this module of the Colour Harmoniser are those that relate to rules for colour harmony:

- The tool should be based on a wireframe model of colour harmony that can be rotated within a three-dimensional, perceptually uniform, colour space, and the wireframe used should be selectable by the user.
- The tool should incorporate a module that is responsible for the creation of harmonious colour schemes and the requisite mathematical calculations.
- The tool should be able to incorporate specific rules (such as a rule for legibility of text) for particular application areas.
- The tool's mathematical functions and data-input mechanisms should be constructed so that they can handle an arbitrary number of colours.

Human aesthetic evaluation of colour seems to function at a holistic level—we can appreciate harmonious colour schemes even when the individual colours used differ significantly from our personal favorites, leading to the generalization that colour harmony depends on hue relationships (among other things), but is itself hue independent.[1,33,34] Westland *et al.*[15] has summarized these and other examples consistent with this premise. This led us to develop the metaphor of a colour molecule that we have previously described; rotating a colour molecule about the center of the colour wheel causes the hues of the colour atoms to change but maintains the relative relationships (hue differences) between them.

Interface items whose colour atoms are located at, or close to, the ends of the wireframe will have high saturation. To avoid large areas of highly saturated colour,

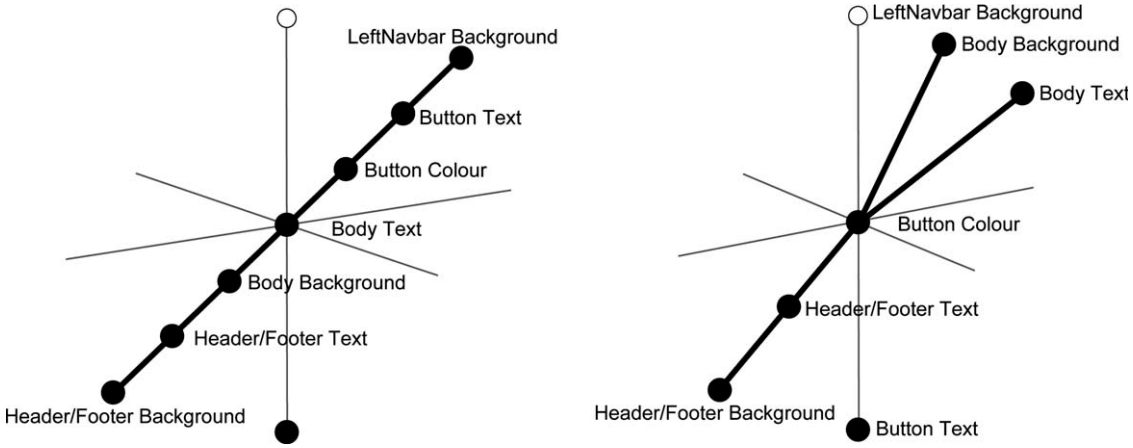

FIG. 4. Two initial arrangements of colour atoms on different wireframes. On the left, a complementary wireframe with the items equally spaced. On the right, a split-complementary wireframe, extended to include the black–white axis.

which is generally unpleasant, it is necessary to incorporate a mechanism, such as Munsell's formula for ensuring that large areas of colour are given a low saturation (although as noted earlier, other formulae could be used instead).

When the colour molecule is rotated, the colour atoms rotate with it, without affecting the hue, saturation, and value relationships between them, so the colour molecule defines an *abstract colour scheme*. That is, it defines the hue, lightness, and saturation differences between elements without requiring their actual hues to be chosen. However, to ensure that the relationships between the colours used in a scheme do not appear to change as the scheme is rotated, it is important that the colour space in which the rotation is performed is perceptually uniform.

Choosing a specific orientation for a colour molecule defines the positions of the colour atoms in colour space, thereby defining (for each orientation of the molecule) a set of real colours for the interface items.

Genetic Optimization of the Abstract Colour Molecule

To ensure that the colour schemes produced by the Colour Harmoniser are pleasant and usable, it is necessary to optimize a number of sometimes contradictory requirements for colour harmony, interface usability, and the aesthetic preferences of the developer. The optimization seeks to maximize a fitness function that we have created with four components:

- The colour balance term is maximized when the lightness and saturation of the colour scheme, each weighted by area, are balanced around mid-grey.
- The wireframe alignment term denotes the average distance of the colour atoms from the wireframe, and is maximum when all the colour atoms lie on the wireframe.
- The distinguishability term is maximized when there is sufficient colour separation between items that the developer has identified as needing to be distinguishable.
- The readability term is maximized when there is a sufficient lightness difference between text and its background to ensure that it is readable.

The fitness function combines these four elements into a value that gives a score to any arrangement of the colour atoms on a wireframe. The fitness function acts a proxy for a human evaluator, allowing the algorithmic assessment and comparison of colour schemes. The fitness function that we have created incorporates the items that we believe to be important for colour harmony, the reasons for which are described later in this article. However, there is nothing to stop additional rules of colour harmony being added if more appropriate colour harmony models are discovered.

The optimizer finds schemes for which the fitness function returns a high value. As the search space is very large and the fitness function is not differentiable, a stochastic search method was chosen. We chose to use a genetic optimization, which is described more fully elsewhere.[35]

In brief, a seed population of abstract colour molecules with the atoms positioned randomly is generated and allowed to evolve through a number of generations. The colour harmony of all members of each generation is evaluated by a fitness function (described below), and the most highly ranked members of that generation and its parent generation are used as the parents of the next generation. Evolution is ensured by random mutation of the genotype (a string comprising the positions of the colour atoms in colour space) and by crossover between the characteristics of pairs of parents. This process can be terminated when a colour scheme of sufficient fitness has been generated.

In general, optimization schemes are designed to discover a single global optimum, but the Colour Harmoniser is intended to allow the designer to choose a personal favorite from a number of candidate colour schemes. Consequently, finding the "best" colour scheme is not

essential (and indeed, for an interface with n components implemented on a system using 24-bit colour there are $(2^{24})^n$ possible colour schemes, so it is not particularly feasible, either). A small population of quite good schemes satisfying the requirements is sufficient.

The Fitness Function—Quantitatively Evaluating Colour Schemes

In this section, we define a fitness function that takes as input the location of the colour atoms in the abstract three dimensional colour space, interface characteristics, and the developer's preferences for the colour scheme, and returns a numerical score for the colour harmony of the scheme. This automatic evaluation of colour scheme quality is the "force" that drives the optimizer's search for those colour schemes that satisfy all four of the constraints (colour balance, wireframe alignment, distinguishability, and readability) simultaneously.

Each term is normalized to lie between zero and one, with a value close to one indicating a high fitness. As the measurements do not fit into this range automatically (and in some cases, decrease in value as the quality increases), they were each normalized during the computation using multiplication by a negative exponential function:

$$N(x) = 10^{-x} \tag{1}$$

For the optimization to work, the fitness function needs to be directly, but not linearly, related to human perceptions of quality. That is, a scheme with a fitness score of 0.5 should appear to be better, but not necessarily "twice as good," as one scoring 0.25, so the addition of the nonlinearity from the normalization should not cause any problems.

For mathematical tractability, the optimization is carried out in the unit sphere—an abstract colour space with limits of ± 1 for each of x, y, z, where y corresponds to the lightness axis and x, z to the chromaticity axes.

The Colour Strength Balance Term. The colour balance requirement is intended to ensure that the value and saturation of the colours in the scheme combine to produce an appearance that is pleasing overall. Munsell has proposed[36] that colour balance for a two-colour scheme will be achieved if

$$A_1 \times V_1 \times C_1 = A_2 \times V_2 \times C_2 \tag{2}$$

where A_i is the area of colour$_i$, V_i is the value colour$_i$, C_i is the chroma (saturation) of colour$_i$.

This equation holds for colour schemes that balance on mid-grey. It allows the lightness, saturation, and area of an interface component to be traded off against one another. However, attempting to balance a colour scheme about mid-grey is only suitable for colour schemes that involve complementary colours. Monochromatic and analogous colour schemes have colours from only one side of the colour space. It is therefore not possible to establish a balance about mid-grey, and we have adopted a different colour balance calculation for these types of colour scheme (which would be also used for other types of colour scheme in which the origin of the colour space [mid-grey] is outside the wireframe).

For the colour schemes in which mid-grey balance is possible, we have taken Munsell's original simple two-element formula for colour strength/area as a starting point, and have extended the idea of balance in a colour scheme to handle an arbitrary number of coloured items, as typical user interfaces contain considerably more than just two interface elements.

In those cases, the colour molecule's balance, C, is based on the distance of its center of colour gravity from the origin of the colour space.

Thus, for molecule m with n colour atoms:

$$C(m) = N\left(\sqrt{b_x^2 + b_y^2 + b_z^2}\right) \tag{3}$$

where $b_x = \frac{1}{n}\sum_{i=1}^{n} a_{i_x} A_v(a_i)$ is the balance of molecule m in the x direction, $A_v(a_i)$ is the visible area of the interface component whose colour is specified by colour atom a_i, and N is the normalization function that was defined earlier.

The designer can choose to add the black/white axis as a pseudo-wireframe element. If this is done, the lightness component (y) is omitted from the calculation, so that only saturation and area contribute to the fitness.

For monochromatic and analogous schemes, we use a variant of the inverse-area rule that trades off saturation against area, maximizing fitness when large areas have a low saturation, small areas have a high saturation, and mid-sized areas have a medium saturation:

$$C(m) = \frac{1}{n}\sum_{i=1}^{n} N\left(\left|(1 - A_s(a_i)\times 0.9) - \sqrt{a_i(x)^2 + a_i(z)^2}\right|\right) \tag{4}$$

$A_s(a_i)$ is the scaled area of component i (with the largest component having $A_s = 1$, the smallest having $A_s = 0$.

To ensure that large areas of strong colour produce low fitness values, the scaled areas are subtracted from 1, to produce the $(1 - A_s)$ term in Eq. (4). In early experiments with the Colour Harmoniser, it was discovered that the colour schemes had large areas of pure black, pure white, or grey, and appeared rather stark. To avoid this, the $(1 - A_s)$ is multiplied by 0.9, which limits the minimum saturation of the largest area to 0.1. This modification was inspired by Manniesing's work on automatic colour choice for multimedia presentations.[29] The nonzero saturation leaves a hint of colour in the largest area and adds a degree of subtlety which significantly improves the resulting colour schemes.

The overall balance $C(m)$ is based on the sum of differences between the desired saturations $(1 - A_s)$ and the actual saturations (the square root term).

The Wireframe Alignment Term. As discussed previously, it is widely held that colour harmony will be achieved when colour atoms are constrained to one of a

set of simple geometric shapes or wireframes; the fitness function therefore includes a term, W, that rewards molecule m when its atoms are on the wireframe and penalizes it when its atoms are remote from the wireframe. The range of values for W is 0 to 1.

$$W(m) = \frac{1}{n}\sum_{i=1}^{n} N(w_i) \tag{5}$$

where w_i is the distance from atom$_i$ to the nearest point on the wireframe.

The Distinguishability Term. The distinguishability calculation produces a score in the range 0–1 that is an average over all pairs of items (or item-groups) that the designer has identified as needing to be distinguishable. To evaluate how distinguishable one item is from another, it is necessary to know the real colour that will be associated with the coordinate of each abstract colour atom. For this to be possible, the transformation that will be used to map from the abstract colour space onto a real colour space must be known. Using this transform and the coordinates of any pair of items, it is possible to calculate their real colour difference.

The function used in the Colour Harmoniser prototype evaluates the distinguishability of a pair of items in two steps. A distinguishability score between a pair of items $D'(m)$ is a function of the Euclidean distance $|a_i,a_j|$ between the two colour atoms.

$$D'(m) = \frac{1}{k}\sum_{i=1}^{n}\sum_{i=1}^{n} \delta(i,j)(1 - N(d(a_i, a_j))) \tag{6}$$

where k is the number of pairs of items that should be mutually distinguishable, $d(a_i,a_j)$ is the Euclidean distance between atoms a_i and a_j in abstract colour space, and $\delta(i, j) = 1$ if components i and j have to be distinct and 0 otherwise, and k is the number of such pairs.

This $D'(m)$ term is an assessment of how well the interface element colouring satisfies the distinguishability criterion for those pairs of items specified as distinguishable by the developer. It has a worst-case value of 0 when each of the pairs of items that are required to be distinguishable have the same colour (i.e., occur at the same position in the abstract colour space). As the pairs of items become more distinguishable, D' increases, and the closer it is to 1 the better.

The final distinuishability score (D) is derived from the D' terms. For some molecules, $D = D'$; for others, a penalty is applied and $D < D'$. The penalty function will be discussed shortly.

The Readability Term. Readability depends on contrast between the text and its background, but it is more important that the contrast be in the light–dark dimension than contrast of hue or saturation.[26,37–40] A contrast ratio of 5.6 ($L_m = 0.7$) is regarded as acceptable by ISO 9241 for colour displays, according to Smith[41] cited in Bangor.[42] Zuffi *et al*.[43] found that a Commission Internationale d'Eclairage (CIE) lightness difference, δL, of 25–30 was sufficient for readability. If $L_d = 10$ then $L_l = 40$ would correspond to a luminance modulation, $L_m = 0.6$ or a contrast ratio of $L_c = 4{:}1$, comparable to the W3C recommendations.[44]

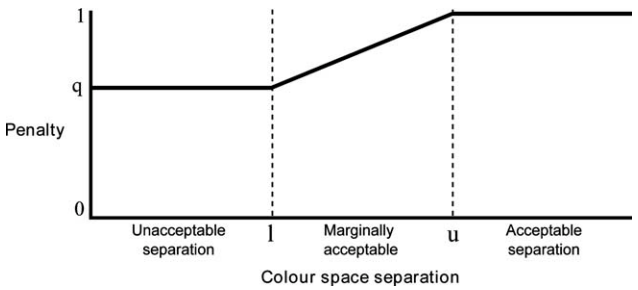

FIG. 5. The penalty function leaves fitness values above an upper threshold, u, unchanged, and progressively penalizes values below u, down to some lower threshold, l.

In informal experiments documented more fully elsewhere,[35] the authors found that, using medium saturation complementary hues at 30° hue increments in the CIELAB space, an average ΔL of 48 was necessary for reading type down to 8 pt for positive (dark type on a light background) and negative (light type on a dark background). The angle subtended by the characters under the conditions of the experiment was slightly smaller than in the experiments reported by Zuffi, which explains the somewhat higher contrast ratio that was found necessary, as readability drops off very quickly with text below 18 min of arc.[42]

To ensure that the text is easily readable, a target ΔL of 48 was chosen. Polarity of the text (whether the background is lighter or darker than the foreground) was not evaluated by the readability fitness term, as the user is able to invert the light/dark values of all the interface components at a later stage of the colour harmonization process (see below), so it is impossible to rely on a particular text/background polarity.

The readability contribution to the fitness function was calculated as:

$$R(m) = \frac{1}{k}\sum_{i=1}^{n}\sum_{j=1}^{n} \delta(i,j)(1 - N(l(i,j))) \tag{7}$$

where k is the number of text items with readability constraints, n is the number of colourable items in the interface, $l_{i,j}$ is the difference in lightness between items i and j in abstract colour space, and $\delta(i, j) = 1$ if two components have to be distinguishable and 0 otherwise.

Minor reductions in the fitness function terms for colour balance or wireframe alignment do not greatly affect the usability of an interface. However, if distinguishability or readability decreases so that a control is indistinguishable from its background, or so that text is unreadable, the interface may be unusable. Therefore, a penalty factor is introduced, to sensitize the fitness value to distinguishability and readability thresholds and to ensure that critical errors (even if only in a single element) in readability and distinguishability have a significant impact on a scheme's fitness.

Figure 5 shows the shape of the penalty function; where the colour space separation is sufficiently large (it exceeds an upper threshold u), the penalty factor is 1, so

multiplying a fitness term by this factor leaves it unchanged. Where the separation is less than some lower threshold, l, so that text would be unreadable or an interface component indistinguishable from its background, the penalty factor is set to a value significantly less than 1. Consequently, when the fitness term is multiplied by the penalty factor, the fitness term is significantly reduced. In the region between u and l, the penalty factor is progressively reduced from 1 to its minimum value.

Combining the Four Fitness Components. The final fitness function is made up of the four scores just described. Each of the four terms has a uniform range, which ensures that no one term is so heavily weighted that it swamps contributions from the others. The four terms are combined as a weighted sum:

$$F = w_1C + w_2W + w_3D + w_4R \tag{8}$$

where F is the overall fitness score, C is the colour strength balance score, W is the wireframe alignment score, D is the penalized distinguishability score $P(D')$, and R is the penalized readability score $P(R')$.

The genetic optimizer uses this weighted sum to evaluate abstract colour schemes and to direct the search. A detailed discussion of the weights chosen is available elsewhere.[35] To provide the designer with a pool of candidate colour schemes from which to make a choice, 10 independently generated populations are optimized and the best of each population is presented to the user.

Although the 10 schemes derive from differing initial populations, they will appear to be related as they are based on the same constraints and on the same wireframe at the same initial orientation in colour space. All 10 schemes will use the same hues, because this initial orientation is the same. However, the order of the atoms on the wireframe differs from one scheme to another, and the optimizer will have positioned the atoms differently, so differing values of lightness and saturation will be used for the elements in each of the schemes. Limiting the number of schemes to 10 seemed a good compromise as it provides a reasonable amount of choice without overwhelming the user with too many options, which has been found to reduce user satisfaction.[45]

Premature convergence—convergence of the optimization to a poor solution, a common problem in genetic optimization—is avoided by random perturbations of the genotype of some members of the population between generations, and using the dynamic restart algorithm,[46] which restarts the optimization from a different point in the search space if no progress is made for a preset number of generations. Further details on the precise optimization method are available elsewhere.[35]

Abstract–Concrete Conversion Involves Three Colour Spaces

The result of the optimization is a set of abstract colour schemes. These have to be transformed from abstract coordinates to real colours.

The mapping between the abstract molecule and its concrete representation is more complex than it might seem; the space in which the optimization process occurs is a sphere, an idealized shape which simplifies the mathematical manipulations, but the space in which the user manipulates the colour scheme needs to be perceptually uniform, so that the colour relationships established during the optimization are preserved under rotation. Perceptually uniform colour spaces are decidedly nonspherical, so true invariance of colour relationships under rotation, which is the ideal, can only be achieved by limiting the length of the wireframe so that there is no rotational angle at which it protrudes beyond the boundaries of the colour space. For many hue angles, this would severely limit the saturation of possible colours. Some compromises are necessary between the desire to preserve the relationships derived during the optimization between the colour atoms and using all of the available saturation at each given wireframe orientation within the colour space.

Ideally, a spherical perceptually uniform colour space could be used for both design and display. However, no such idealized colour space exists. We have therefore found it helpful to work with three different colour spaces. The genetic optimization takes place in a space that is spherical and thus mathematically tractable but is neither perceptually uniform nor constrained by the colour gamut of realistic display devices.

After the optimization, therefore, the coordinates of the atom in the colour molecule in the abstract space are mapped to coordinates in an intermediate space that is perceptually uniform, but strangely shaped (see Fig. 6), because of the shape of the colour space "inside the human brain," and also because of the gamut limitations of display devices.

The reason for using an intermediate, perceptually uniform space is to ensure that, when the user tweaks the colours by rotating the wireframe or inverting it in the light–dark dimension, colour separations that ensure readability of text and distinguishability of interface components are maintained. If the space in which these transformations were performed were not perceptually uniform, rotation could move text colour closer to the colour of its background, so that at some rotational angles of the wireframe, it could become unreadable. The CIELAB colour space is used as the intermediate space, as it is approximately perceptually uniform and there are well-defined mathematical transformations between CIELAB and the final display space, sRGB.

There are various possible approaches in dealing with the inconsistency between the spherical shape of the abstract colour space and the irregular perceptually uniform colour space, none of which is a perfect solution. Limiting the size of the wireframe so that it would never poke outside the available gamut would significantly restrict the saturation of the colours at many orientations, producing very subdued colourings. Optimizing with the wireframe constrained to fit within the CIELAB space at its smallest diameter, and changing the length of the wire-

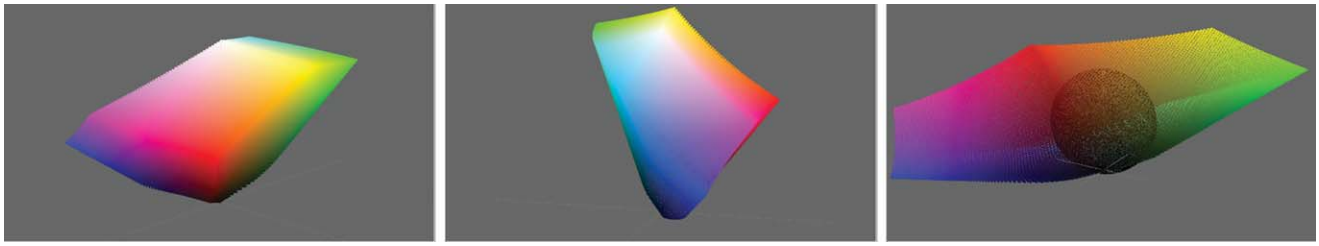

FIG. 6. Three views of the sRGB colour space projected into the CIELAB colour space; the highly irregular shape of this (approximately) perceptually uniform space complicates the mathematics needed to generate an optimized hue-independent colour molecule. In the third orientation, a sphere has been superimposed that shows the (unreasonable) gamut limitations that would be required if the wireframe were never to poke outside the colour space at any orientation.

frame to fit the gamut as it rotated, would satisfy readability and distinguishability constraints, but it would change the saturation as the user rotated the wireframe, violating GUI predictability guidelines.[47] Setting the size of the wireframe to some medium value, and limiting its rotation to those parts of the CIELAB space within which it would fit would prevent certain colours from being displayed.

This last alternative, which is the one we used, modifies only those out-of-gamut colours that would be otherwise unrenderable to a close approximation. This approach will alter the intended relationships between the out-of-gamut and other elements, potentially violating legibility and distinguishability constraints, which sounds like a poor compromise, but in practice, works well for several reasons. First, the optimization generally produces text/background and distinguishability-required component pairs in which the members of the pair are widely separated (i.e., not wireframe neighbors), so even if one of the pair is not as distant as intended, the elements are still well separated. Second, the colours which are out-of-gamut will be affected by the current wireframe rotation and the setting of the user's saturation control (to be discussed shortly). At some settings of these controls, the user can force some atoms out-of-gamut, so their colour is only approximately that intended by the optimizer. However, these changes are being produced in real time as the user manipulates a slider control, so they happen gradually and are immediately reflected in the displayed colour scheme. If the colours look wrong at some settings, this is a transient state that can be easily remedied by continuing to adjust the controls. Finally, this method does not introduce any unexpected colour variation or apparent discontinuities. All colours continue to change smoothly as the user moves the controls, and, although some colours might differ from those calculated, in practice this is not evident. These arguments are not meant to be watertight exercise in formal logic; all of the alternatives involve drawbacks, but the chosen alternative of mapping out-of-gamut colours to close alternatives has not proven to cause problems.

An intermediate-sized wireframe was used for the 10 colour schemes initially presented to the user. It is a compromise: it is sufficiently large so that it includes well saturated colour schemes, including approximated out-of-gamut colours at some orientations and less-than-maximum saturation colours at others, as illustrated in Fig. 7.

Should the Colour Schemes Be Updated to Allow for Dynamic Content?. Updating the colour scheme to allow for dynamic changes in content is intentionally not included as part of the Colour Harmoniser; if the algorithm was strictly applied, any variation in the areas of components (e.g., such as the appearance of an interface component, text being updated, or a page being scrolled) would cause the colour balance to be reassessed, potentially altering the colours of all elements. This would be visually quite disturbing. It was therefore decided to design the colour scheme using the actual interface and typical content, rather than update the scheme at run-time and cause, from a user's perspective, inexplicable alterations to the colour scheme when the structure of the interface changed.

Personalization (Tweaking) of Colour Schemes

An abstract optimized colour scheme can be mapped to many different real colour schemes. This is the stage in the process where the user's personal preferences can be taken into account. Design criteria relevant to this Colour

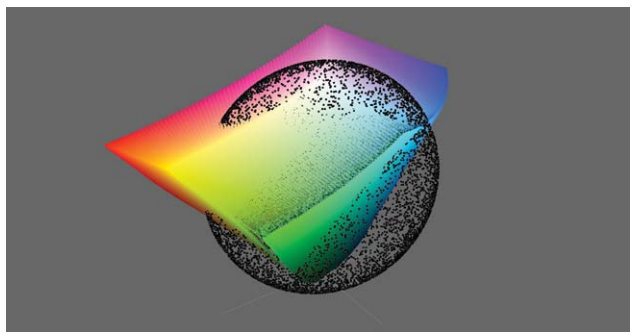

FIG. 7. For the abstract to real mapping, an intermediate-sized wireframe that involves out-of gamut-colours at some orientations and less-than-maximum saturation colours at others was chosen. This means that the user sees reasonably saturated colour schemes and while some atoms will be out-of-gamut and will have their colours mapped to the closest in-gamut colour and therefore not have as wide a separation as intended. However, the colour differences due to clipping are not severe and as the user can see the resultant scheme at all times including the effect of their varying the personalization controls, this gives them the ability to accept schemes in which the separation (colour difference) between items is somewhat less than that intended by the optimizer.

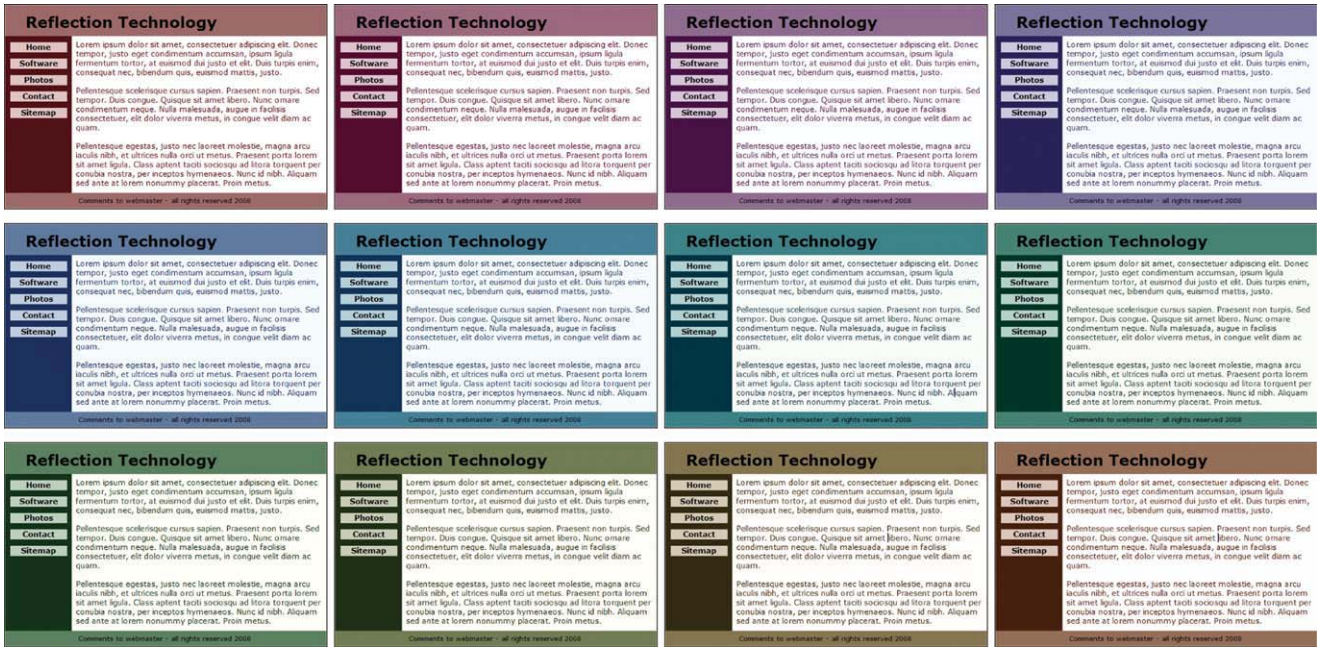

FIG. 8. The user sees a continuously varying image as the hue control slides between its limits. This image shows snapshots at 30° intervals.

Harmoniser module are those that relate to user preferences and those that specify other desirable characteristics:

- The tool should give the user freedom to personalize the overall orientation (hue angle) of the wireframe to alter the actual colours (hues) used in a colour scheme even though the relationships between the hues and value relationships have been determined elsewhere.
- The tool should be able to update the colours of the actual interface being designed, so that the design decisions can take effect instantly, allowing the designer to rapidly explore the design space.
- The aesthetic alterations available to the user should preclude damaging either the colour harmony of the interface or its readability.

There are several ways for a user to personalize the displayed colour schemes. Hue and saturation controls are provided, but not contrast, as the readability of the text is heavily dependent on light–dark contrast.

The colours of the interface or web site can be changed holistically by rotation of the wireframe. As mentioned earlier, this rotation takes place in the CIELAB space to maintain the readability of text and distinguishability of interface items, and after mapping from CIELAB to sRGB, the interface is updated in real time. The user experience is thus very simple and immediate; the user adjusts a single slide control, and the hues of the entire interface update in real time. Figure 8 shows a series of images of a web page taken at 30° intervals as the user adjusts the hue control.

A saturation control is also provided. It allows a user to alter the colourfulness of all the elements at once and allows the whole scheme to be adjusted holistically and continuously, from one containing only black/white/grey to full saturation, without impacting readability. Figure 9

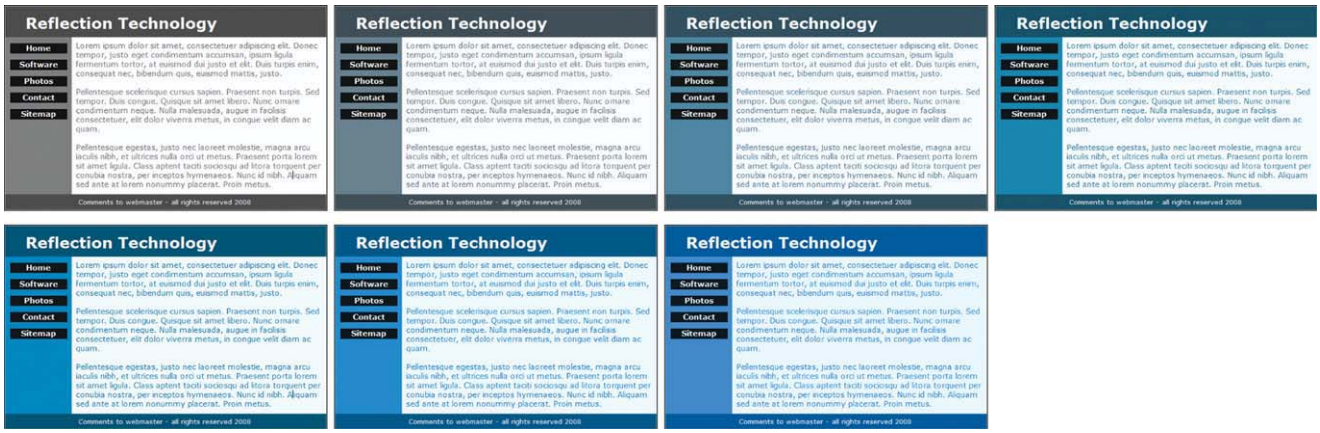

FIG. 9. Altering the saturation of a scheme in equal steps from 0 (black/white/grey) to full saturation.

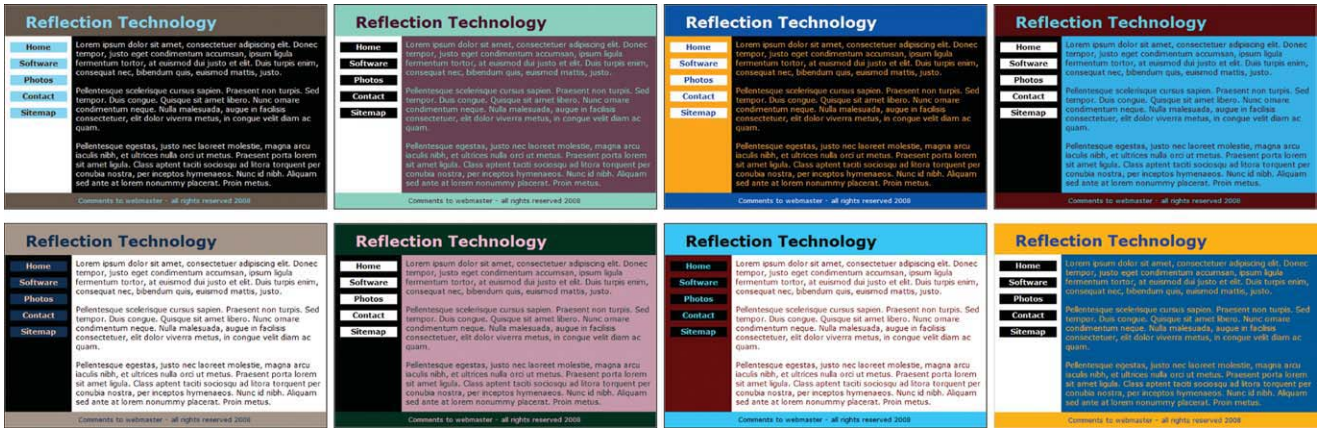

FIG. 10. Each of the images in the bottom row uses the same hues and saturations as the image above it, but light has been replaced by dark and dark by light.

shows the effect of a user varying the saturation of a colour scheme over its full range.

The final tweaking control allows for visual assymmetry in the appeal of certain colours; some colours look better when light. For example, yellow is naturally a light colour, whereas "dark yellow" is a khaki brown, and is much less used in colour schemes. Red, by contrast, is naturally dark, and its light version, pink, is difficult to harmonize. This asymmetry illustrates a weakness in the theory that colour harmony can be maintained by simple hue rotation of a wireframe. We have allowed for it by including a control that allows a designer to invert the wireframe in the lightness dimension. This allows the transformation of a scheme to one that appears quite different (although the colour relationships are identical) by simply clicking a control. The schemes in the top row of Fig. 10 have all undergone this conversion to produce the schemes directly below them in the bottom row of the figure.

The personalization controls update the complete colour scheme immediately while maintaining its colour harmony, its textual legibility, and the distinguishability of its components. These properties contrast with typical colour selection systems, which are modal, operate on a singled coloured element at a time, and often require the user to visualize the effect of colouring a large area from the appearance of a small swatch in the colour selector.

The Colour Harmoniser presents a rather different approach to colour selection, as the user is fine-tuning a scheme whose colour harmony has previously been determined by the optimization phase. However, although the personalization controls do constrain the types of modifi-

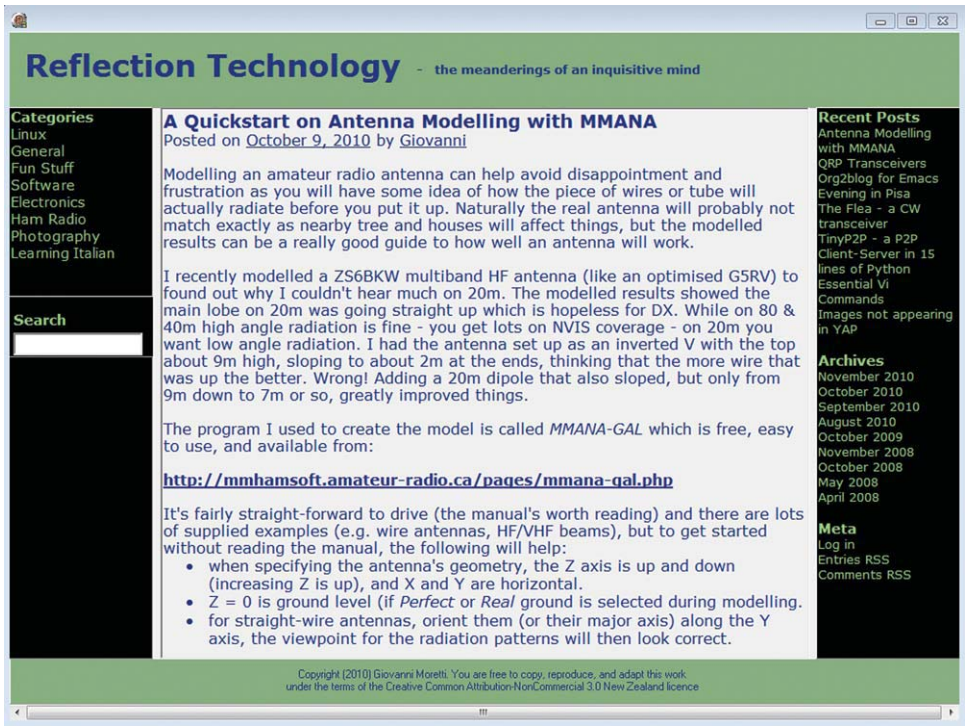

FIG. 11. Colour Harmoniser created colour scheme applied to a real web page.

cations that are possible, the changes the user can make to a scheme can be quite subtle or—as is clear from the amount of variation in Figs. 9 and 10—quite dramatic, leaving the user with a large degree of flexibility in final colouring.

A side effect of the holistic nature of the personalization controls is that they allow the serendipitous discovery of colour schemes. Because the controls alter all the colours in an interface colour scheme simultaneously, they allow the user to explore a very large number of colour scheme variations in a very short space of time, something that is quite impractical with the one-at-a-time method of colour selection. An example of a web page using a Colour Harmoniser-generated colour scheme and real content is shown Fig. 11.

CONCLUSIONS

The Colour Harmoniser architecture is based on replaceable models of colour harmony, user interface semantics, and a simple characterization of a user interface. By providing a means to holistically adjust the colours of complete schemes, it enables the rapid generation and exploration of a wide variety of user interface colour schemes.

In this article, we have described extensions to existing rules for colour harmony that make it possible to ensure text legibility and visual distinguishability between components, two important pragmatic issues for designers of computer interfaces and web pages.

We have introduced the concept of abstract colour schemes and the mapping of these to real colour schemes and advocated separating colour harmonization into two phases; generation of an abstract colour scheme whose colour harmony is independent of hue, and generation of a real colour scheme in which the colour harmony generated in the first phase is unchanged, but the designer's personal preferences for overall colouring, light–dark contrast, and saturation can be taken into account. This phase separation has the additional benefit that aspects of colour harmony that involve large numbers of tedious mathematical calculations can be performed automatically by the software, and aspects that involve the designer's aesthetic preferences can be performed by direct manipulation of simple, intuitively meaningful controls. We describe the fitness function that we have designed to assess candidate colour schemes for their conformity to the rules for colour harmony, and we present a number of images showing the results of the colour harmonization.

In a following article, we will describe experiments we have conducted to test the ideas behind the Colour Harmoniser.

1. Munsell AH. A Color Notation: A Measured Color System, Based on the Three Qualities Hue, Value and Chroma. Boston: George H. Ellis; 1907.
2. Judd DB, Wyszecki G. Color in Business, Science and Industry. New York: Wiley; 1963.
3. Ou L-C, Luo MR. A colour harmony model for two-colour combinations. Color Res Appl 2006;31:191–204.
4. Arnheim R. Progress in color composition. Leonardo 1987;20:165–168.
5. Polzella DJ, Montgomery DA. Dimensions of color harmony. Bull Psychonomic Soc 1993;31:423–425.
6. Burchett KE. Color harmony. Color Res Appl 2002;27:28–31.
7. Graves M. Color Fundamentals—With, 100 Color Schemes. New York: McGraw Hill; 1952.
8. Itten. In: Birren F, editor. The Elements of Color, translated by Enrnst van Hagen. New York: Van Nostrand-Reinhold; 1970.
9. Howlett V. Visual Interface Design for Windows. New York: Wiley Computer Publishing; 1996.
10. Foster V. Colour Matching Handbook. Rochester, Kent: Grange Books; 2004.
11. Kobayashi S. The aim and method of the color image scale. Color Res Appl 1981;6:93–107.
12. Lavendel L, Kohler T. The Story of a Color Advisor. In: Proc. of the Sixth IS&T/SID Color Imaging Conference, Scottsdale, Arizona, 1998. p 228–232.
13. Kobayashi S. A Book of Colors. New York: Kodansha International; 1987.
14. Ou L-C. Existing theories of colour harmony; 2007. Available at: http://colour-emotion.co.uk/harmony.html [accessed June 2011].
15. Westland S, Laycock K, Cheung V, Henry P, Mahyar F. Colour harmony. Colour: Design Creativity 2007;1:1–15.
16. Murch GM. Physiological principles for the effective use of color. IEEE Comput Graph Appl 1984;4:49–54.
17. Travis D. Effective Color Displays: Theory and practice. London, New York: Academic Press; 1991.
18. Wright P, Mosser-Wooley D, Wooley B. Techniques & tools for using color in computer interface design. Crossroads 1997;3:3–6.
19. Shneiderman B, Plaisant C, Cohen M, Jacobs S. Designing the User Interface: Strategies for Effective Human–Computer Interaction. Boston: Addison Wesley; 2009.
20. Granger GW. An experimental study of color preferences. J Gen Psychol 1955;52:3–20.
21. Morriss RH, Dunlap WP, Hammond SE. Influence of chroma on spatial balance of complementary hues. Am J Psychol 1982;95:323–332.
22. Rapoport A, Rapoport A. Color preferences, color harmony, and the quantitative use of colors. Empir Stud Arts 1984;2:95–112.
23. Morriss RH, Dunlap WP. Influence of value on spatial balance of color pairs. J Gen Psychol 1987;114:353–361.
24. Linnett CM, Morriss RH, Dunlap WP, Fritchie CJ. Differences in color balance depending upon mode of comparison. J Gen Psychol 1991;118:271–284.
25. Nakakoji K, Reeves BN, Aoki A, Suzuki H, Mizushima K. eMMaC: Knowledge-based color critiquing support for novice multimedia authors. In: Proceedings of the Third ACM International Conference on Multimedia. San Francisco, California: ACM Press; 1995. p 467–476.
26. Meier BJ. ACE: A color expert system for user interface design. In: Proceedings of the First Annual ACM SIGGRAPH symposium on User Interface Software, Alberta, Canada, 1988. p 117–128.
27. MacIntyre B. A constraint-based approach to dynamic colour management for windowing interfaces. Report nr CS-91–55, University of Waterloo; 1991.
28. Schlueter KG. Perceptual synchronization in window systems, Masters thesis, University of Waterloo; 1990.
29. Manniesing A. Creating harmonious and legible colour schemes in the automated generation of multimedia presentations: Centrum voor Wiskunde en Informatica; 2003. Available online at http://repository.cwi.nl/search/fullrecord.php?publnr=4088.
30. Nack F, Manniesing A, Hardman L. Colour picking: the pecking prder of form and function. In: Proceedings of the 11th ACM International Conference on Multimedia. Berkeley, CA: ACM; November 2003. p 279–282.
31. Rogowitz BE, Rabenhorst DA. CRAFT: A tool for customizing color and font selections guided by perceptual rules. 1993. p 140–143.
32. Kelly I. A two stage evolutionary model for the computer aided design of colour combinations. In: Sivik L, editor. Digital Creativity. Scandinavian Colour Institute, 1996; 8, 3–4, 106.

33. Birren F. The Elements of Color. New York: Van Nostrand Reinhold; 1970.
34. Nemcsics A. Experimental determination of laws of color harmony. Part 3: Harmony content of different hue pairs. Color Res Appl 2008;34:33–44.
35. Moretti GS. A calculation of colours: Towards the automatic creation of graphical user interface colour schemes. Ph.D. Thesis. Palmerston North: Massey University, Whitefish, MA: Kessinger Publishing; 2010. 280 p. Available online at http://hdl.handle.net/10179/1492.
36. Munsell AH, Farnum RB. A Color Notation: An Illustrated System Defining All Colors and Their Relations: Kessinger Publishing; 1941.
37. Fukuzumi S, Yamazaki T, Kamijo K, Hayashi Y. Physiological and psychological evaluation for visual display colour readability: A visual evoked potential study and a subjective evaluation study. Ergonomics 1998;41:89–108.
38. Lin C-C. Effects of contrast-ratio and text color on visual performance with TFT-LCD. Int J Ind Ergon 2003;31:65–72.
39. Lin C-C. Effects of screen luminance combination and text color on visual performance. Int J Ind Ergon 2005;35:229–235.
40. Ojanpaa H, Nasanen R. Effects of luminance and colour contrast on the search of information on display devices. Displays 2003;24:167–178.
41. Smith WJ. ISO and ANSI Ergonomic Standards for Computer Products: A Guide to Implementation and Compliance. Upper Saddle River, NJ: Prentice-Hall; 1996.
42. Bangor AW. Improving Access To Computer Displays: Readability for Visually Impaired Users. Virginia Polytechnic Institute and State University; 1998.
43. Zuffi S, Beretta G, Brambilla C. A color selection tool for the readability of textual information on web pages. In: Proceedings of IS&T/SPIE Internet Imaging VII, San Jose: SPIE, 2006.
44. Zuffi S, Scala P, Brambilla C, Beretta G. Web-based versus controlled environment psychophysics experiments. In: Cui LC, Miyake Y, editors. 19th Annual Symposium Electronic Images Sciences and Technology. San Jose: SPIE, 2007.
45. Schwartz B. The Paradox of Choice: Why More is Less. New York: Ecco; 2004.
46. Solano M, Joyner I. Performance analysis of evolutionary search with a dynamic restart policy. In: Proceedings of the 20th International Florida Artificial Intelligence Research Society Conference, Florida, 2007. p 186–187.
47. Shneiderman B. Creating creativity: User interfaces for supporting innovation. ACM Trans Comput Hum Interact 2000;7:114–138.