Color Lens: Adaptive Color Scale Optimization for Visual Exploration

Niklas Elmqvist, Pierre Dragicevic, Jean-Daniel Fekete

Color Lens: Adaptive Color Scale Optimization for Visual Exploration

Niklas Elmqvist, *Member, IEEE,* Pierre Dragicevic, and Jean-Daniel Fekete, *Member, IEEE*

Abstract—Visualization applications routinely map quantitative attributes to color using color scales. Although color is an effective visualization channel, it is limited by both display hardware and the human visual system. We propose a new interaction technique that overcomes these limitations by dynamically optimizing color scales based on a set of sampling lenses. The technique inspects the lens contents in data space, optimizes the initial color scale, and then renders the contents of the lens to the screen using the modified color scale. We present two prototype implementations of this pipeline and describe several case studies involving both information visualization and image inspection applications. We validate our approach with two mutually linked and complementary user studies comparing the Color Lens with explicit contrast control for visual search.

Index Terms—Color scales, visualization, interaction technique, Magic Lens.

1 Introduction

Color is one of the most basic visual attributes used by the human perceptual system [1]. For this reason, mapping quantitative data values to color is standard practice in visualization, and has been utilized for a wide array of purposes such as education, science, and medicine. Consider a doctor studying an X-Ray image to look for fractures where tissue and bone density are mapped to luminance using gray scale colors (Figure 1), or an oceanographer studying seasonal changes in the world's oceans where temperature is represented using a heat map.

Despite the universal use of color in visualization, color mapping can introduce information loss, due both to the limited color depth of conventional computer screens (where, for example, a gray scale typically only has 256 potential values) [2], as well as the limited color acuity of human vision [3]. For high-resolution data—data with important large-scale and small-scale variations, such as the X-Ray image in Figure 1—these limitations may lead to a loss of visual features (e.g. not finding an existing fracture).

This article introduces the Color Lens, a new interaction technique that addresses the limited color resolution problem by interactively optimizing color scales according to a sampling region controlled by the user (Figure 1). To validate our design, we performed two controlled experiments where we asked participants to search for visual features hidden in procedural noise and in photographs. We measured search time when using a Color Lens and a contrast slider. Our results show a significant time improvement for the Color Lens over the slider.

- *N. Elmqvist is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN. E-mail: elm@purdue.edu*
- *P. Dragicevic and J.-D. Fekete are with INRIA, Paris, France. E-mail: dragice@lri.fr and jean-daniel.fekete@inria.fr*

2 Background

2.1 Motivation

Visualization uses interactive graphical representations of data to aid the user's cognition when viewing, interacting with, and reasoning about the data. In general, this high-level cognitive process is known as *visual exploration*: the use of visualization to form, test, and validate hypotheses about data, where the goals and tasks for the exploration may not be known in advance [4]. Visual exploration is a high-level process consisting of numerous lower-level cognitive, perceptual, and motor operations, such as visual feature search [5], estimation, comparison, and interaction [6].

In this article, we mainly focus on perceptual issues of visual exploration involving the use of color. Clearly, visual exploration cannot be carried out effectively if the above lower-level tasks are not properly supported. For example, if colors cannot be discriminated in a particular area of a visualization, one cannot search for features or get any insight about the data. More specifically, we focus on *visual search*, a ubiquitous low-level user task in visual exploration.

Visual search is defined as the search for a particular feature with a known appearance in a visual space. We consider both guided and unguided search, i.e., whether or not the user has a priori knowledge about the spatial location of the feature in the image.

Color can convey both categorical or quantitative data—we focus on the latter here. In particular, our work is mostly concerned with *high-resolution quantitative data*: data with large-scale as well as small-scale features that are important to the viewer. The basic problem with this data is that the large data range may cause small-scale details to be lost due to both limited color scale resolution in the display hardware, as well as limited color perception in the user.

High-resolution quantitative data of this nature is

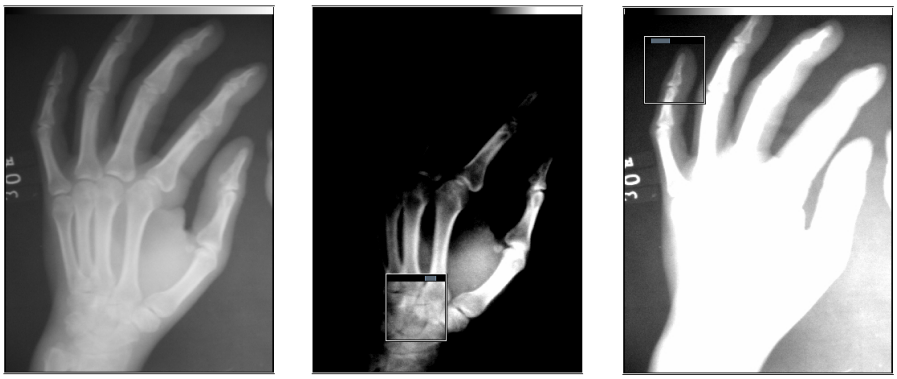

Fig. 1. A global Color Lens applied to an X-Ray image. The color scale dynamically adapts to the lens contents.

common in many domains. HDR photographs and scientific simulation images are often encoded in 32 bits or more [7], [8], meaning that meaningful information is contained in color variations whose magnitude is less than 1/1,000,000 of the whole data range. In the visualization domain, examples are numerous and include the power law in social networks (such as co-citation networks, where a small set of authors receive most citations), economic data (where the Pareto rule says that 80% of sales often come from 20% of the clients), or oceanographic temperature readings (where the overall temperature range is large, but it is the small variations that often matter the most).

2.2 Requirements

The performance of the visual search task depends on the relative visibility of the features in the display [5]. Because we do not know which particular features are of interest to the viewer, our work attempts to optimize the visibility of *all* features through exploratory interaction. Based on this discussion, and on the nature of the visual exploration process [4], we impose the following requirements on our new technique:

- *General:* The technique should work with any visualization, and it should be informed by the data and not be biased towards certain features.
- *Interactive:* The technique needs to be interactive to support visual exploration and should not be the result of a static query or off-line algorithm.
- *Faithful:* The visual presentation should be faithful to the data and should not mislead users [9].

2.3 Visual Search

Supporting visual search is a common research theme within cognitive psychology; Wolfe [5] gives a survey. The theme has also been extensively investigated in the HCI community, perhaps most prominently by Halverson et al. [10], [11]. Although Halverson and Hornof investigate how text color affects visual search [12], no existing technique optimize color and contrast to make features stand out better. They thus do not support many low-level tasks [6].

2.4 Color Display and Perception

The amount of information that can be conveyed through the color channel partly depends on the capabilities of the display hardware. Current displays have a limited color resolution, regardless of the resolution of the data space. Modern LCD displays use 8 bits per color component (or 6 bits with dithering). This only allows for conveying 64 to 256 different values using a gray scale visualization. In practice, this means that data features whose magnitude is less than 1/256 of the total data range are invisible. Some specialized medical displays have 10-bit color per channel, allowing for 1024 different shades of gray.

It has been argued that a color depth of 8 bits is enough, because the human visual system cannot distinguish more colors [13]. At the same time, it has been shown that the human contrast sensitivity for sinusoidal gratings increases with the mean luminance level [3], [14]. This partly justifies the recent efforts in building high dynamic range (HDR) displays, i.e., displays that have a much wider range of luminance and a higher color depth [13], [15].

However, because of the eye adaptation mechanisms [3], the number of shades of gray that can be distinguished does not increase linearly with average luminance. Current models predict only about 1,200 just noticeable difference (JND) steps for a monitor 300 times brighter than a conventional computer display [13], which is still far from enough for displaying high-resolution data. Additionally, at a high luminance, light scattering in the cornea (glare) can negatively contribute to the legibility of visualizations [16].

2.5 Color Scale Design and Optimization

Quantitative data is typically visualized using *color scales*, i.e., functions that map scalar values to colors, usually an interpolation between a discrete set of colors. The most basic color scale is a linear transition from black to white, i.e., a gray scale. Gray scales are widely used because luminance is especially efficient at faithfully conveying quantitative data [9]. However, as discussed above, the amount of information that

can be communicated through the luminance channel is limited. One common approach to enhance a color visualization is hence to vary hue and/or saturation in addition to luminance, e.g., using pseudo-coloring. However, pseudo-colors tend to produce unwanted artifacts, such as artificial contours [9].

Some research has focused on designing color scales that maximize the number of discriminable colors while preserving perceptual uniformity and ordering [17]. ColorBrewer [18] provides a set of optimized color scales commonly used for displaying quantitative values, such as the LOCS [19] color scale. Compared to gray scales, perceptually-optimized color scales better show small differences while remaining faithful to the data, but the gain is marginal.

2.6 Adaptive Color Scales and Tone Mapping

It is common practice to adapt color scales to the data distribution (or post-process the data) in order to convey more information. For example, logarithmic color scales are often used when the data follows a logarithmic distribution. General algorithms have been proposed, such as density function mapping and histogram equalization [20], as well as techniques for enhancing the contrast of natural images [21].

A related theme within the computer graphics community has been to faithfully reproduce high-dynamic range (HDR) images, i.e., natural images whose pixel intensity spans over several orders of magnitude [7]. These images contain both large-scale features (changes in illumination) and small-scale features (changes in albedo) which are difficult to reproduce on a regular computer screen. Solutions are *tone mapping* or *tone reproduction* operators, that can be either spatially uniform or locally adaptive [3], [7], [22], [23]. Tone mapping has also recently been applied to scientific visualization [8].

Although adapting color scales to the data distribution often conveys more information, it does not solve all problems. Spatially uniform techniques such as histogram equalization can make better use of color gamut and can suppress the impact of outliers on contrast. However, they can misrepresent the data distribution, and the gains are small when the distribution is close to uniform (such as in X-Ray images).

Tone mapping techniques are more concerned with perceptual faithfulness, but their goal is to produce natural-looking images rather than conveying actual quantities. In particular, locally adaptive techniques filter out large-scale changes in illumination because the eye is less sensitive to them. Even if the result looks realistic, it is not equivalent to watching the original scene [24]. In visualization, large scale color variations can convey important information and losing them can hide information such as global trends.

2.7 Interactive Control of Color Scales

Some of the limitations of color scales have been addressed by adding interactivity. Many visualization tools allow the user to try different color scales or to build new ones [18], [25], [26], which is useful because the appropriate color scale often depends on the nature of the data and the task at hand [9]. Color scales can also be parameterized by direct manipulation. When the visualization is updated in real-time, such techniques allow for searching for features in the data [25], [27]. Similarly, commercial applications such as Adobe Photoshop provide color-domain filters that can be tuned interactively to reveal near-invisible details in images, especially within under- or overexposed regions of photographs.

The interactive manipulation of color scales adds a new dimension to data visualization by allowing the user to navigate into the color space. However, explicit color scale control techniques usually require the user to haphazardly manipulate one or several degrees of freedom—e.g., brightness and contrast—until the desired result is obtained. Such a process can be tedious, especially when one just wants to enhance a region of interest. In contrast, some visualizations couple 2D or 3D navigation techniques with an automatic adaptation of the color scale to the viewport [17], [28]. Although highly natural, this implicit control of color scales tightly links the color adaptation to the current viewport, and is thus not flexible enough to decouple the interactive visual exploration in space and color.

2.8 Focus+Context Techniques

Among existing navigation techniques for visualization, focus+context [29] techniques have been widely advocated in a variety of domains because they prevent users from getting lost in data by providing both local detail and global context. However, surprisingly little work has been devoted to the use of focus+context techniques for color optimization. Among the exceptions are an HDR painting application that combines a global tone-mapped view of the image with a local linearly-mapped view [30], and a screen magnifier that can perform contrast adjustment [31].

The concept of Magic Lenses [32]—2D shapes that can be moved over a visualization and *locally* affect its representation—-is of direct relevance to our work. We will describe this concept in more detail later.

3 The Color Lens Technique

The basic idea of the Color Lens is to adaptively change the mapping from data space to color space (i.e., the color scale) based on an analysis of the contents of a subset of the dataset so that a maximum amount of the available color precision can be devoted to that subset. The subset is specified using one or several lenses that the user can interactively move

and resize on the visualization canvas to visually explore the dataset. In this section we provide a brief overview of the Color Lens technique from the user's perspective in order to illustrate its benefits and to give an idea about the new possibilities it opens.

3.1 Basic Motivating Example

Suppose a user needs to study an X-Ray image. Using standard tools, the user can zoom and pan the image, and manipulate colors using brightness and contrast controls. However, it is challenging to use these tools to focus on specific parts of the image. Even when the color scale is automatically adapted to the viewport, the user has to switch back and forth between different zoom levels and easily loses context.

In contrast, the Color Lens allows for easily examining different parts of the image. Figure 1 illustrates the use of a Color Lens that automatically adapts the contrast and brightness of an image given a region of interest. By default, the image is shown in its true colors. When the user drags a Color Lens on the image, it adapts the contrast and brightness of the display so that the image area inside the lens takes up the whole display gamut: when the area inside is bright, the whole image looks darker; when it is dark, the whole image looks brighter. Deactivating the Color Lens restores the image to its original color.

This mechanism obeys a "camera exposure" metaphor similar to how digital cameras dynamically adapt their exposure according to a small region in the view finder. Image exposure will be optimized for that region while the rest of the picture may be partly underexposed or overexposed. Underexposed and overexposed regions are visible in Figure 1, and although data is temporarily lost as a result of this process, it does give an indication that some regions have higher values than those in the lens (saturated white) and others have lower values (completely black) without affecting the stability of the display.

Interestingly, clipping the brightness (see the rightmost image in Figure 1) is precisely what allows the interior of the lens to be clearly visible, because otherwise—for example, on an HDR display—legibility would be severely impeded by the extremely bright surrounding light. The Color Lens smoothly animates the changes in the color scale, which further enforces the camera automatic exposure metaphor.

The key feature of the Color Lens is that it ensures that the maximum amount of the available gamut is devoted to displaying the data currently in focus. This enables the user to detect even small features. In a way, the technique allows users to navigate into color space similar to the way zoomable user interfaces [33] allow users to navigate in geometric space. However, the Color Lens is easier to use than traditional methods of interactive color scale manipulation that require explicit parameterization of color scales [25],

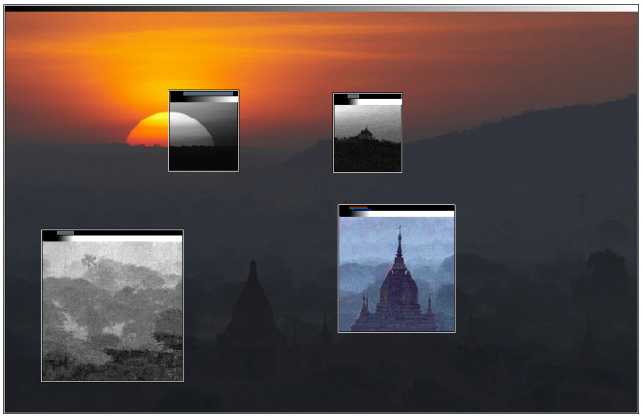

Fig. 2. Multiple local lenses used on a photograph.

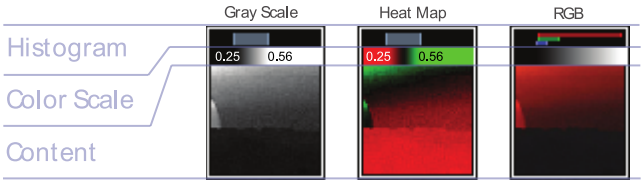

Fig. 3. Decorations showing histogram and color scale.

[27] because it maps better to the data exploration task and reduces the level of spatial indirection [34]. It also provides higher resolution than sophisticated color maps [19] while using a simpler visual form.

Magic Lenses [32] use interactive lenses to change visual representation, but our work is different:

- Magic Lenses have so far only been used to manipulate images at the *semantic* and *geometric* levels, not at the *color scale* level;
- Magic Lenses indicate the user's interest and are primarily designed for looking through, whereas our approach is to use the lens as a sampling region that may or may not be the user's focus;
- No existing work on Magic Lenses discusses the use of nested hierarchies of lenses which can be used to zoom in space in order to explicitly support a multiscale visual exploration process.

No Magic Lens automatically adapts contrast to the lens focus, instead requiring manual tuning. They hence share the drawbacks of direct manipulation approaches discussed earlier. Furthermore, none of these tools has been empirically evaluated, so their actual benefit to visualization is unknown.

3.2 Interacting with Color Lenses

The Color Lens supports more sophisticated interactions such as resizing lenses or creating multiple lenses. Lenses can also have local or global effects on the display. *Global lenses*, used in the previous scenario, affect the whole image, so the lens itself is mainly used for specifying a sampling area. *Local lenses*, on the other hand, affect the image only inside their bounds, like a Magic Lens [32].

Fig. 4. Multiscale navigation in US Census data, involving successive zooms into both space and color domains.

It is possible to use several lenses in the same visual space, but we currently define no method to compose lenses together. Instead, when several **global** lenses are present, the content of all lenses are used to compute a color scale for the whole visual space. In other words, the regions covered by each global lens is combined using a union operation of the screen space covered by each lens. The whole combiend space is then used for adapting the display gamut. When using multiple **local** lenses, they are independent of each other, allowing different regions of the visual space to be shown in different ways (Figure 2).

The above applies to the analysis component of the Color Lens, not the visual representation. We also maintain a configurable *stacking order* (Section 4.3). If lenses overlap on the display, the color scale of the topmost lens will be used for the overlapped region.

Color Lenses have different types that are controlled by the user. Figure 3 shows three types. The first lens uses a gray scale that is linearly optimized according to the extrema of the data inside. Data extrema are also shown on the top of the lens and dynamically updated as the lens is dragged or resized. The color scale is updated accordingly and is also shown on the top of the lens (Figure 3), or on the top of the parent lens when the lens is in global mode (Figure 1). We can show the actual data values for the extrema on top of the color scale, as a kind of graphical legend.

The second lens in Figure 3 is similar to the gray scale lens, except it uses a different color scale. The third lens enhances the contrast of color images by analyzing the three color components separately, extracting their extrema and rescaling the color components uniformly up to the saturation point.

3.3 Multiscale Color Navigation

Color Lenses are especially powerful when combined with 2D navigation techniques. We have experimented with the technique in a zoomable scatterplot visualization of US Census data (Figure 4). In addition to the continuous navigation tools, double-clicking inside a lens automatically adjusts the viewport to the lens through a smooth transition. Once a lens is in focus, additional lenses can be recursively added inside the parent, creating a nested hierarchy of lenses.

This navigation strategy is based on zooming in color and geometry that is consistent with the Visual Information Seeking mantra, "*overview first, zoom and filter, then details on demand*." [35] This is particularly useful when searching for features in visualizations that are dense both in space and in color.

Lens hierarchies also provide navigation history. While navigating in high-resolution data, users can backtrack by clicking outside the current lens to move back to the parent lens. This allows users to progressively build search trees for multiple lines of inquiry.

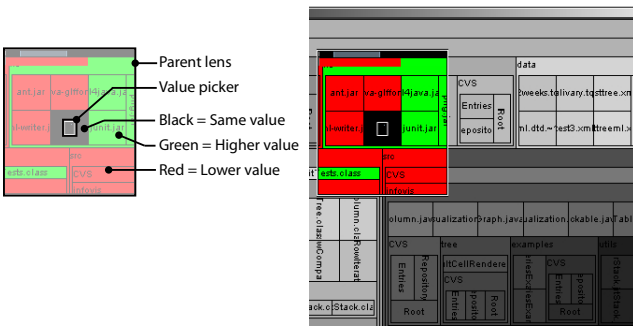

Fig. 5. A compound lens used to compare adjacent cells in a treemap. The lens consists of a large container lens with a red-black-green color scale and a nested global lens that acts as a sampling region.

Finally, lens hierarchies can also be used to build compound lenses. For example, lenses of different types can be put side by side into the same compound lens container to form a toolglass [32].

Another example, illustrated in Figure 5, involves a value comparator. The treemap in the example uses

color to encode the file size, but it can be difficult to compare the color values, especially when the two files are not side by side. The comparator lens is built by putting a small global lens (no color scale of its own) inside a bigger local lens that has been assigned a ternary red-black-green color scale (i.e., three colors only). The large lens acts as a container and uses the region defined by the small global child lens as a sampling region. This now means that every object in the display that has the same quantity (i.e., file size) as the object underneath the sampling region will appear in black. In other words, black cells are of the exact same file size as the sampled one. Lower-valued objects (i.e., smaller files) appear in red and higher-valued ones in green (i.e., larger files).

4 Basic Framework

Color Lens techniques can be specified in terms of an abstract graphics pipeline where the data contents of each lens is initially rendered to off-screen geometric buffers (or G-buffers [36]), i.e., images whose pixels hold abstract data instead of colors. This step is followed by inspecting the distribution of the data, adapting the color scale to optimize this distribution, and then rendering the actual pixels of the lens to the screen using the modified color scale.

4.1 Model

Consider a visualization that maps data to an image on the screen [37]. Part of the data is mapped to geometrical features, and the resulting spatial arrangement of the data on the screen is called the *visual substrate*. Another part of the data is mapped to colors. Let us assume a mapping from n-dimensional quantitative data—i.e., $D \subseteq \mathbb{R}^n$—to a standard RGB color space—i.e., $C \subseteq I^3$ (or I^4 for alpha channel), where I denotes an intensity coordinate in RGB space, usually $I = [0, 1]$. The transformation from data to color is performed using a color scale $cs : D \rightarrow C$ that is typically continuous and obeys some perceptual ordering, such as monotonically increasing brightness.

In the Color Lens, a subset of the data $l \subseteq D$ is analyzed before the color transformation and the color scale cs is modified to optimize the color precision for that subset. The analysis typically consists in deriving histograms for the data distribution of the subset l.

4.2 Abstract Rendering Pipeline

The rendering pipeline uses an initial rendering pass to a geometrical buffer (G-buffer), followed by analysis, adaptive modification of the color scale, and then rendering to the screen. Only at the last stage do we transform to the potentially limited color space (e.g., 32 bits on standard computer screens), ensuring that data precision is maintained during the analysis. The last step also involves rendering metadata, such as lens decorations, text annotations and labeling, and user interface elements.

A G-buffer contains data that has been partially rendered, i.e., pixels containing quantitative data values instead of colors. This is necessary in order to be able to analyze complex visual substrates consisting of multiple geometrical entities that may potentially interfere with each other. The G-buffer is a subset of the dataset sampled in the current viewport, and can be analyzed before being rendered.

4.3 Lens Hierarchy

Color Lenses are organized in hierarchies, with a single static Color Lens at the root containing the whole visual substrate. This root lens is consistent with the initial fitting of a color scale to a data domain (typically by calibrating the endpoints of the color scale to the extents of the data range) that is performed for virtually all visualizations that make use of color scales. Lenses may contain child lenses that are specified in terms of the local coordinate system defined by the parent and clipped to its bounds.

Lenses have flags for *rendering* and *analysis*. Rendering controls whether the lens is drawing its own representation, as opposed to delegating this to its parent. Similarly, analyzing lenses use their own contents when computing data histograms, whereas non-analyzing ones use their children for this.

We define the following lens types:

- **Analyzing lens.** The contents of this lens are taken into account for building histograms, and the results are propagated upwards in the hierarchy (a global mode lens, see Figure 1).
- **Rendering lens.** This lens renders its own representation using a color scale based on its analyzing children (see the root lens on Figure 1). The color scale used can be different from the parent.
- **Analyzing renderer lens.** This lens defines its own color scale and analyze its own contents, forming a local lens (Figure 2).
- **Null lenses.** This lens performs no analysis and no rendering, but may be used as containers (cf. toolglass in Section 3.3).

5 Implementations

We have implemented two prototypes of the Color Lens technique: one using OpenGL and programmable graphical processing units (GPUs), and the other as part of the InfoVis Toolkit [38]. Both are built in Java and achieve real-time performance.

5.1 OpenGL Implementation

Our standalone implementation of the Color Lens method—ColorLensGL—is built in Java using the JOGL[1] bindings for OpenGL 2.0 and utilizes GLSL

1. http://jogl.dev.java.net/

(the OpenGL shading language) and the OpenGL ARB framebuffer object (FBO) extension for realizing the G-buffers required by the abstract Color Lens piepline. ColorLensGL is designed as a graphical backend that can be used by any visualization application that wants Color Lens support. It is based on the GPU shader framework for information visualization proposed by McDonnel and Elmqvist [37].

5.1.1 Data Precision

Modern GPUs use floating-point arithmetic in all computations, but depending on the actual hardware the precision may be 16-bit (half-precision) or standard IEEE 754 32-bit (single-precision) floating point values. By performing all computations using floating-point arithmetic, ColorLensGL preserves the full precision of the data space (up to 32 bits) and minimizes data loss until the actual rendering of data values to 8-bit color components on the screen.

5.1.2 Visual Substrate

The ColorLensGL visual substrate is represented by a 2D scene graph consisting of simple 2D primitives such as points, lines, triangles, rectangles, and convex polygons. Instead of having a color or material associated with each node, as is common for standard scene graphs, this scene graph represents a mapping from the data domain to the spatial domain, independent of the color mapping. Therefore, each node has an associated *data tuple* in n-dimensional space instead.

To allow for GPU processing, data tuples are realized as floating-point color components in RGBA space. Additional data dimensions over four thus require the use of multiple render targets using the framebuffer object OpenGL extension. Most visualizations use only one or two dimensions, however.

To simplify the use of grid-based, space-filling visualizations, or volume visualizations [8], we also support arrays of multidimensional data (*data tensors*) that are mapped to scene graph shapes, the same way image textures are mapped to 2D or 3D geometry. Data tensors are implemented using 2D and 3D RGBA floating-point textures, so each additional multiple of four in data dimensions require the use of another texture through OpenGL's multitexturing mechanism.

5.1.3 Rendering

ColorLensGL uses a two-pass rendering pipeline that relies on data being sampled in the current screen resolution into off-screen G-buffers, implemented using the OpenGL framebuffer object (FBO) extension. Instead of rendering color components to the screen, we render data values to the off-screen buffer. G-buffers are written and stored in the video card memory, speeding up the rendering process.

The rendering algorithm proceeds by first rendering the visual substrate to the G-buffer, and then reading back the resulting data from video into system memory. This step only needs to be performed when the viewport changes. After that, whenever the lens is moved or resized, the contents of the lens will be inspected. Given the results, we adapt the color scale for the lens contents using a fragment shader.

5.1.4 Interaction

ColorLensGL has a zoomable user interface [33] supporting basic pan and zoom operations for navigation. Lens operations include creating, moving, resizing, and deleting color lenses. Users can also change color scale and toggle lens decorations.

5.1.5 Application: US Census Visualization

We have implemented the multiscale US Census visualization application from Section 3.3 using the ColorLensGL framework. The application reads a subset of the US Census 2000 dataset and draws a multiscale geographical map of the United States showing population density. The user can navigate in the dataset by panning and zooming, and the representation can be changed to show the data at the different levels of aggregation stored in the dataset. By iteratively building nested hierarchies of Color Lenses, the user is able to visually explore the dataset and to discover small-scale differences in population density that would otherwise be indistinguishable to the naked eye.

5.1.6 Application: Image Inspection

We have implemented the image viewer from Section 3.2 using the ColorLensGL framework. The user can interactively change the global or local contrast of a loaded image using a hierarchy of Color Lenses, and can save the final output to a standard PNG or JPEG file when satisfied with the result. The final output image will only contain the influence of the lenses, not their interface decorations.

5.2 Software-rendered Java2D Implementation

Color Lenses can be used in any visualization supported by the InfoVis Toolkit [38]. They have been implemented using the multi-layering mechanism of the toolkit, allowing for stacking lenses as visualizations on top of other visualizations without requiring changes to existing visualization implementations.

The implementation uses two layers: one is the lens layer, which manages the lens representation, and the other one is a background layer located behind the visualization layers, which is used to compute the range of attribute values inside the lens. This background layer renders visualizations using a special mode to compute the range of values of the attribute assigned to the color using a software G-buffer.

Since the InfoVis Toolkit references object by index, our G-buffer is actually an object buffer that uses a standard Java RGBA image to encode integer values. Reusing the standard Java image object saves us from

re-implementing the Java Graphics2D class for rendering in a G-buffer. In the special mode triggered by the background layer, we render using object indices instead of colors. When the rendering is finished, we compute the minimum and maximum attribute values for the indices contained in the G-buffer. The only subtlety is undefined values: when the attribute value of an object is undefined, the index of that object should not be written to the G-buffer. Therefore, when the attribute used to display colors is changed, the G-buffer should be recomputed unless the attributes used before and after contain no undefined values.

Compared with the OpenGL prototype, the software pipeline offers more flexibility and an unlimited precision but is slower for large lenses. The flexibility allows us to compute several color ranges for different layers. For example, in a node-link diagram, a Color Lens can control the color of the link layer and the node layer independently (Figure 6). Performance degradation is noticeable with large lenses (more than 300×300 pixels) when the G-buffer needs to be computed or when more than about 1000 objects are visible. This is not a major issue with small lenses (< 200 pixels), but for other situations, a hardware-accelerated implementation may be required.

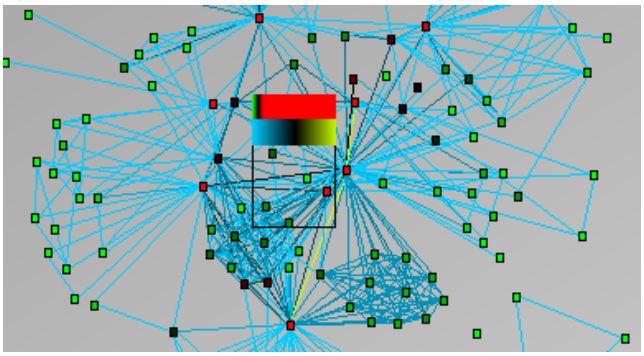

Fig. 6. Color Lens in a node-link graph visualization implemented in the InfoVis Toolkit showing two color scales, nodes and links. The lens magnifies the difference in the color mapping for nodes and edges.

6 Evaluation

In order to assess the benefits of the Color Lens technique to users, we compared Color Lens to direct contrast control in two types of visual search tasks. A first study involved searching for features in randomly-generated noise, whereas the second involved searching for hidden objects in photographs. These two tasks were simple enough to be quickly explained to our subjects, and they did not involve value comparisons but only shape detection under various noise conditions.

The explicit color scale manipulation tool was a range slider (Figure 8) modeled after the histogram manipulation tool provided in imaging applications such as Adobe Photoshop. Shrinking the slider's thumb increases contrast, and dragging it allows for quickly exploring dark or bright regions of the image. This design is more efficient than having separate controls for brightness and contrast.

Because we focus on interactive approaches, we opted to not include any advanced color optimization technique such as pseudo-colors, histogram equalization, or tone mapping. Our two techniques only employ linear rescaling of the color space, and are hence faithful to the underlying data (see R3 in Section 2.2). Automatic and interactive color optimization techniques are however not incompatible, and can be easily combined: since no color optimization technique is able to convey arbitrarily small color variations, it is still arguably useful for the analyst to be able to manually specify a range of interest in the data using either a Color Lens or a contrast slider.

The two studies were conducted using the same participants and within the same session. However, because their design differs in some ways, we present them separately. Common aspects are presented here.

6.1 Hypotheses

Our general hypotheses were the following:

H1: *Color Lens will be faster than explicit contrast control.*

H2: *Color Lens will be faster when the user has knowledge of feature location.* The lens will be particularly efficient when users know where to search.

H3: *Color Lens will be faster when searching in low-contrast regions.* Low-contrast image regions can be easily enhanced using a Color Lens (high-contrast regions tend to "saturate" the lens).

6.2 Participants

We recruited 16 unpaid participants from the university student pool (3 female, 13 male), all of them experienced computer users and screened to have normal or corrected-to-normal vision with full color perception. Ages ranged from 23 to 50 (mean 28.7, median 27). Demographics were self-reported.

6.3 Apparatus

The experiment was conducted on a Dell laptop computer. The computer had a 17-inch WUXGA LCD display with a native 1920×1200 pixel resolution and a color depth of 24 bits. Gamma correction was performed prior to the first experiment session and was maintained at 1.2. The experiment was conducted in a room with standard artificial lighting (no natural light) that was kept constant for all participants.

6.4 Task

In both studies, a trial consisted of a visual search for for a feature in a gray scale image. Participants

were first informed of which target to find and invited to hit the space bar to display the image. They then had a limited amount of time to identify the feature in the image using the tool provided (Color Lens or contrast slider). Participants were instructed to hit the space bar as as soon as they spotted the feature and then mark its approximate location on the screen. When marking the position, the image was reset to its original contrast settings and the tool was deactivated. Participants were instructed to be as fast as possible.

The test platform measured completion time, correctness, and tool use. Completion time was defined as the time difference between the two space bar presses. Trials were either correct or incorrect. Tool use was recorded as false when the participant completed the trial without using the tool, typically because the participant had already spotted the feature using the naked eye, otherwise as true. When the time cap was exceeded, the software aborted the trial and marked it as incomplete. We derived the time cap from a pilot study, striving to give enough time to detect the feature while at the same time minimizing frustration in case the task was too difficult for a particular trial.

6.5 Procedure

Both experiments consisted of a sequence of search trials in a randomized order, grouped by the Method factor. Individual trials were interleaved with instruction screens during which the participant could rest from the previous trial and prepare for the next.

Prior to performing the studies, participants were given an instruction sheet describing the tasks and explaining the two techniques. Before each study, they received training for both techniques using images different from the ones used in the experiment.

Sessions typically lasted 45 minutes: 5 minutes of training, 30 for the first study, and 10 for the second.

7 Study 1: Search in Noise

The motivation for studying artificial images was to provide a canonical example of the use of color scale optimization, independent of specific data. Being in control of the image generation allowed us to better control the characteristics of the search task.

The images were generated using simplex noise, an improved version of the classic Perlin noise [39]. This type of procedural noise exhibits the combination of large-scale and small-scale variations that is characteristic of the high-resolution data that we study here.

7.1 Task

The task was to search for small circles buried in procedural noise. The images were generated by linearly mapping a 10-octave simplex noise to a gray scale image of resolution 800×800 pixels and color depth of 8 bits (256 luminance values). The mapping was normalized so that each image took up the whole display gamut minus the luminance of the brightest feature. The scale of the first octave was a function of the Noise Complexity factor (see top of Figure 7).

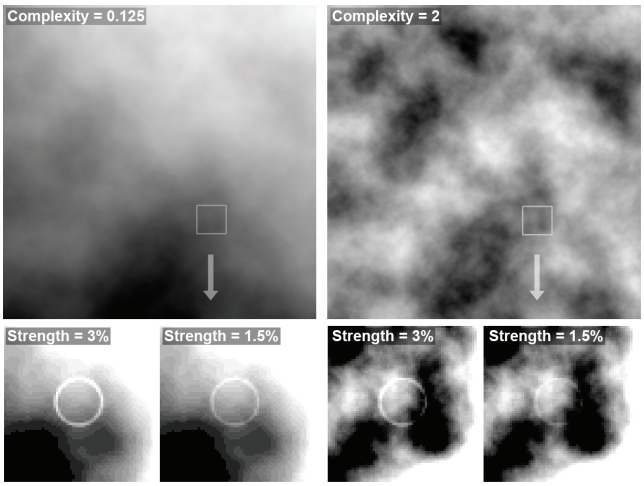

Fig. 7. Above: a randomly generated image with two different noise complexities (NC). Below: a magnified and contrast-enhanced view of the feature's neighborhood, with two different feature strengths (FS).

The features were white, anti-aliased circles of 20 pixel radius and 2 pixel thickness blended into the image at a random location. This was done using alpha blending, with the opacity being a function of the Feature Strength (see bottom of Figure 7).

Trials had a time cap of 20 seconds. In some trials, a rectangle (spatial hint) indicated which quadrant contained the feature. When found, the subject specified the feature position by clicking a cell on a 8×8 grid. Features were placed to not intersect grid lines.

7.2 Experimental Conditions

Method. In order to test H1, we included a Method (M) factor with the following two conditions:

- *Contrast slider.* Explicit control of contrast using a 800×25 pixel range slider located below the image (see Figure 8).
- *Color Lens.* Control of contrast using a 50×50 pixel global Color Lens (see Figure 1). Left drags were used to move the lens, and the +/- keys for resizing it. Other functions were disabled.

The same type of linear color-space transformation was applied to the images in both method conditions, i.e., brightness B_{min} was mapped to black and brightness B_{max} was mapped to white. The difference was in the way B_{min} and B_{max} were obtained: in the contrast slider condition, they were obtained from the location of the handles, whereas in the Color Lens condition, they were sampled inside the lens.

Noise Complexity. In order to test H3, we added a Noise Complexity (NC) factor. Noise complexity was defined as the ratio between image size and the scale

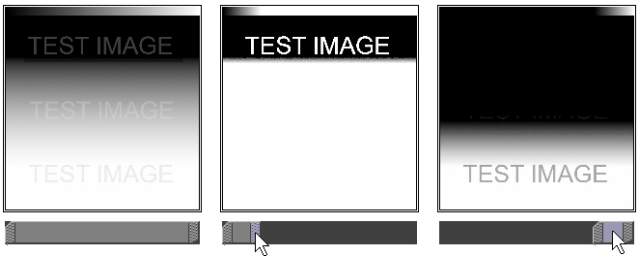

Fig. 8. Using the contrast slider to enhance an image.

(wavelength) of the first simplex noise octave [39]. In other words, this ratio represents the normalized wavelength of the noise. The larger this value, the stronger the high-spatial frequencies of the images relative to low-spatial frequencies. This in turn affects local image contrast, and allows us to approximate cases that are likely problematic for the color lens, such as the presence of low-contrast details near high-contrast boundaries. We included two different values for noise complexity: 0.125 and 2.00 (see Figure 7).

Feature Strength. The degree to which the visual feature stands out from the background image is likely to have an effect on search tasks. We used two values: 1.5% and 3% of the full brightness range of the images (Figure 7). The feature strength thus governs the amplitude of the feature in the image. With 256 levels of brightness in a standard 8-bit gray scale, this corresponded to about 4 and 8 brightness steps between the feature and the background.

Spatial Hint. To test H2, we introduced a Spatial Hint (SH) factor that controlled the display of a rectangle indicating which image quadrant contained the feature. This reduced the search space by 75%.

7.3 Study Design

We included the following factors (within-subjects):

- Method (M): *Contrast slider, Color Lens*
- Noise Complexity (NC): *0.125, 2*
- Feature Strength (FS): *1.5%, 3%*
- Spatial Hint (SH): *no, yes*

We used a full factorial design for the experiment based on the above factors, yielding $M \times NC \times FS \times SH = 2 \times 2 \times 2 \times 2 = 16$ conditions.

Each of these 16 conditions was assigned 8 search problems, a total of 128 trials. A problem consisted of a random seed for producing the noise image and a feature position. To control for task difficulty, we randomly picked 8 different problems and used the same 8 problems for all the 16 conditions.

A pilot study revealed that with only 8 different problems, participants quickly learned the feature positions. To avoid this, we applied 90-degree rotations and symmetries (8 transformations in total) to the problems. This preserves the difficulty while preventing learning. Each $NC \times FS \times SH$ condition was hence assigned a unique transformation, and this assignment was counterbalanced across subjects.

The 128 trials were grouped by method and their order randomized within each method. The same ordering was applied so that both two methods saw exactly the same series of trials. The order of presentation of methods was counterbalanced across participants.

With 16 participants conducting 128 trials, we collected time, correctness, and tool use measurements for a total of 2048 individual trials.

8 Study 2: Search in Photographs

In order to ground our work in a more realistic search task, we complemented the previous study with a shorter study involving finding visual features in actual photographs (see Figure 9 for example task).

8.1 Task

The photographs were all collected from the online photo sharing site Flickr.[2] They were selected on the criteria of containing over-exposed, under-exposed or low-contrast regions (e.g., night scenes, back-lit shots, fog, snow, underwater shots). A total of 32 "good" pictures were selected from a collection of about 150 images on the basis of containing hidden visual features that could be unambiguously described with a short sentence. Features were annotated with invisible bounding boxes. With the exception of being converted to gray scale, images were not altered from their original appearance on Flickr.

Prior to starting each trial, a sentence told the participants what to look for, e.g., "find a framed photo" or "find the guy on the bicycle." Participants were instructed to ask the experimenter in case they did not understand the sentence. Upon hitting the space bar, they had a maximum of 60 seconds to find the feature. The trial was counted as correct if the participant clicked inside the feature's bounding box.

8.2 Experimental Conditions

Because of the limited scope of this study, we only included the Method factor from the previous study.

8.3 Study Design

Given the existence of a dataset of 32 annotated photographs and the single Method factor, we constructed the study as a one-factor within-subjects design. The trials were balanced across participants so that half of the participants first used the Color Lens for half of the trials and the contrast slider for the remainder, and the other half of the participants did the opposite. This mechanism was designed to counter against certain affinities between a technique and a specific image.

With 16 participants conducting 32 trials each, we collected measurements from a total of 512 trials.

2. We do not have permission to reproduce all of the images we collected from Flickr in this paper. Figure 9 gives a representative portrayal of the style of the tasks, however.

Fig. 9. Example task for Study 2: "find the sign with Asian characters."

9 Results

We analyzed the results using a repeated-measures analysis of variance (RM-ANOVA) for Study 1, and a paired (i.e., dependent samples) t-test for Study 2. For Study 1, residuals were normal and a Mauchly test showed that the sphericity assumption was not violated. For Study 2, the data was checked for violations of assumptions for the t-test—residuals were normal, and variances equal (Bartlett's test, $p = .479$).

Table 1 gives the significant effects on search time for Study 1. Figure 10 shows time for each condition.

Search time: The Color Lens was significantly faster in both studies. In Study 1, the average search time with the contrast slider was 8.57 seconds, versus 7.73 for the Color Lens, a 10% improvement over the baseline. In Study 2, the average search time was 7.71 seconds with the contrast slider as opposed to 4.72 seconds for the Color Lens, a 43% improvement for our new technique (significant, $t = 5.605, p < .001$).

As Figure 10 shows, the Color Lens was reliably faster than the contrast slider. The spatial hint had a significant impact on search time for the more difficult conditions (high noise complexity and low feature strength). For Study 2, the speedup gained from the Color Lens technique was also significant.

Correctness: We found no significant main effect of Method on correctness in any of the two studies (Wilcoxon signed rank test, $p = .090$, and $p = .060$, respectively). The average correctness for both Methods was high, 96% for Study 1 and 92% for Study 2,

Factors	F	df, den	p
Method (M)	4.111	1, 15	*
Noise Complexity (NC)	339.825	1, 15	**
Feature Strength (FS)	215.220	1, 15	**
Spatial Hint (SH)	84.886	1, 15	**
NC * FS	63.620	1, 15	**
NC * SH	5.731	1, 15	*

* = $p \leq 0.05$, ** = $p \leq 0.001$.

TABLE 1
Significant effects of search time on factors for Study 1 (ANOVA). All other effects were non-significant.

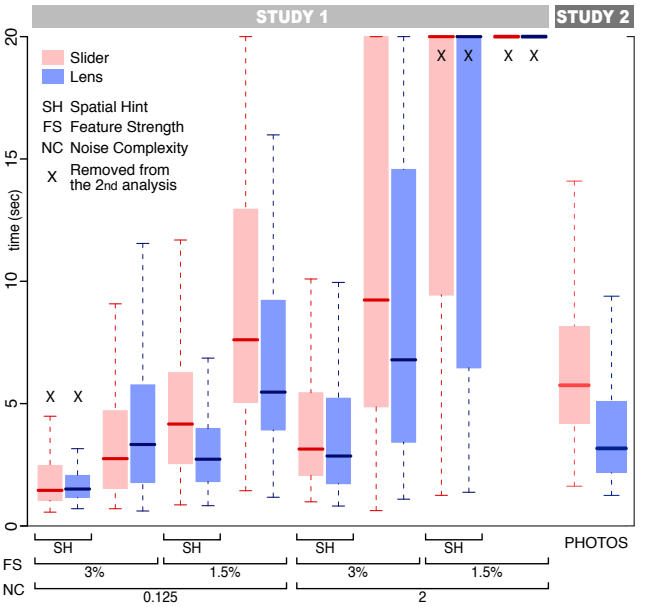

Fig. 10. Search time for each condition (both studies).

so we did not analyze this further.

Tool use: In Study 1, the contrast slider was used in 78.5% of the trials and the Color Lens in 81.4% of the trials—the difference was only marginally significant (Wilcoxon test, $p = .085$). In Study 2, the percentages were 97.7% for the contrast slider and 99.6% for the Color Lens, also only marginally significant (Wilcoxon test, $p = .073$).

Capped trials: The percentage of capped trials in Study 1 was 22.2% for the contrast slider and 20.0% for the Color Lens. In Study 2, the percentage was 0.8% for the slider and 0.0% for the Color Lens.

It might be argued that in Study 1, the ceiling effects caused by the capped trials affected the search time analysis. Furthermore, some trials were clearly trivial in that most participants could solve them without the tool. For Study 2, no such effects were found on the data and very few (16) trials were capped.

We re-analyzed Study 1 after removing conditions where more than 50% of the trials were capped and conditions where the tool was not used in more than

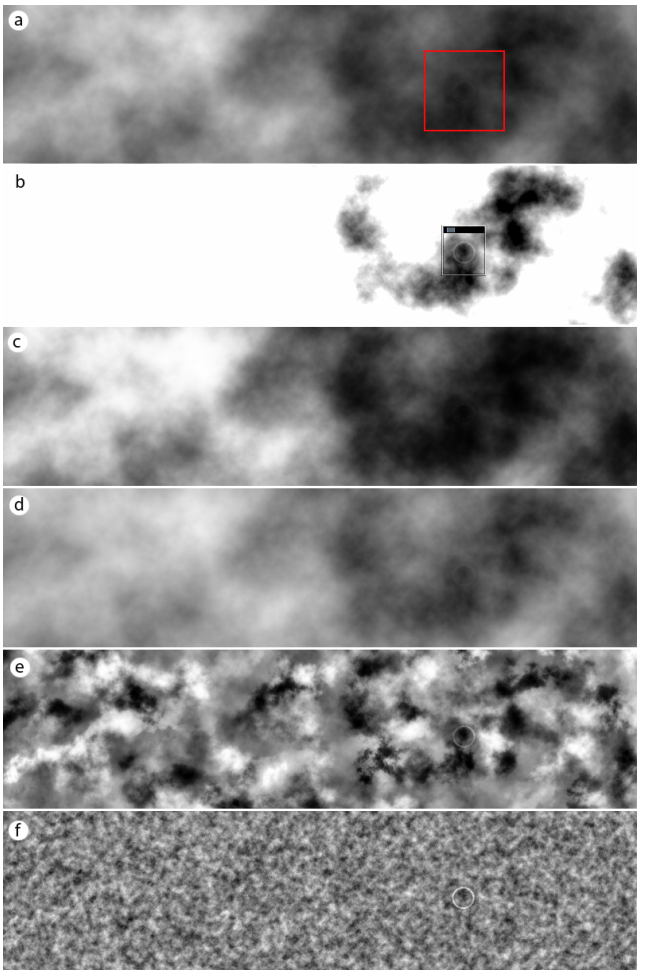

Fig. 11. Comparison between Color Lens and alternative image manipulation techniques (one per row) for an image from Study 1. a) original image, b) Color Lens, c) histogram equalization, d) tone mapping, e) local histogram equalization, and f) low pass filter.

50% of the trials. This led us to remove the four difficult conditions $NC = 2.00$, $FS = 1.5\%$ and the two trivial conditions $NC = 0.125$, $FS = 3\%$, $SH = yes$ (see bars marked X in Figure 10).

We performed our analysis on the remaining eight conditions, with unchanged main effects. There was a significant main effect of Method on search time ($p < .01$): 6.78 seconds for the contrast slider and 5.63 seconds for the Color Lens, a 17% speedup.

10 Discussion

Our experiment yielded the following main findings:

- The Color Lens technique was significantly faster overall than the contrast slider for visual search (confirming **H1**);
- Better speed came at no extra cost in accuracy;
- We found no significant interaction between hint and method in Study 1 (**H2** inconclusive); and
- We found no significant interaction between complexity and method in Study 1 (**H3** inconclusive).

10.1 Explaining the Results

Our results confirmed that Color Lenses allow faster detection of visual features without loss of accuracy.

One explanation behind this result is the lower level of spatial indirection [34] of the Color Lens. The slider *(a)* requires splitting one's attention between controlling contrast and observing the image, and *(b)* does not allow for immediately interacting with features of interest once they have been identified. Although we tried not to measure *(b)* in our study (we asked participants to press the space bar as soon as they saw the feature), it might have played a role.

Furthermore, a special case where the Color Lens is superior to the contrast slider is when one wants to enhance a specific region of the image. This is because the Color Lens has a more direct and thus more predictable mode of operation—the user controls the lens directly on the visual space to enhance. Using the slider, one has to first estimate the color range of the region, then set the range slider to that range. The Color Lens only requires clicking on the image region.

One reason for this is that the contrast slider has two degrees of freedom (1D position and width of the contrast band), whereas the Color Lens has three (2D position and size of the lens). This is perhaps yet another testament to the power of direct manipulation [40], where the general guideline is to minimize indirection in the interface. In this light, it is perhaps not so surprising that the Color Lens achieves better performance than the indirect contrast slider.

Despite the lower level of spatial indirection of the Color Lens, we did not confirm our hypothesis that the Color Lens would be especially faster in guided search, i.e., when the participant has prior knowledge of feature location. In fact, the first study did not show a significant interaction between Method and Spatial hint. On the other hand, Color Lens outperformed the slider by as much as 43% in Study 2, where a large majority of tasks involved semantically guided search.

One explanation for us not being able to find a main effect for the spatial hint in Study 1 might be that the hint—a bounding box 1/4 the surface of the image—did not restrict the search space enough to produce observable effects. The other possibility is that the type of semantic guidance given in the Study 2 is better suited to the Color Lens because it informs the search more efficiently than the spatial guidance, playing to the strengths of the Color Lens.

Our third hypothesis was also not confirmed by Study 1: we observed no significant interaction between Method and Noise Complexity. On the other hand, it is reasonable to assume that Color Lens was so successful in Study 2 partly because features were systematically hidden into low-contrast regions. Many participants did seem to feel that Color Lens was much more helpful when applied on smooth noise or photographs than on high-density noise.

A plausible explanation for the lack of significance for high-density noise might be that raising the magnitude of high frequencies increased the difficulty of the search task across the board, and thus both tools might have been perceived as being less helpful.

10.2 Generalizing the Results

In generalizing these results, we should note that the contrast slider is actually not only a baseline condition, but an improvement of the baseline that exists in most visualization applications. Adobe Photoshop and similar tools use two or more independent control points and thus do not allow for sweeping the contrast range like in our work. Many medical imaging tools do use window and level controls, however.

Furthermore, the acquisition time of the slider should also be taken into account (in the study, participants acquired the slider prior to starting trials), whereas there is no such step needed for the Color Lens. The Color Lens also supports more rapidly acquiring a feature after it has been discovered.

Another important point is the ecological validity of the tasks we used in our studies. We claim that visual search—i.e., finding hidden features in data—is a common task in visual exploration, and thus that the tasks evaluated in our studies are indeed representative. However, it would certainly be worthwhile to study images other than procedural noise and photographs in the future, such as X-Ray images or actual information visualization applications.

We should also note that the current evaluation only studies the main functionality of the Color Lens, but that many other features of the technique remain to be investigated, such as comparing data values on a visualization, finding value extrema, or performing multiscale visual exploration. Furthermore, many search tasks may involve a priori knowledge of not the location but instead the approximate data range of the targets to find (i.e., an intensity hint instead of a spatial hint). We have also not studied the use of local or multiple lenses. However, this article was intended to introduce the Color Lens technique, and we look to future work to expand these issues in greater detail.

We should note that when optimizing for color perception, interactive versus static techniques are two different ways of approaching the problem, and they are not necessarily incompatible. Our approach for specifying ranges of interest in color space is independent from the optimization technique—our prototype currently uses a simple linear rescaling of the color space. Although we advocate the use of interactivity rather than automatic optimization, our approach can be combined with pseudo-coloring and tone mapping for more aggressive color discrimination.optimization.

10.3 Alternatives to Color Lenses

As alternatives to the Color Lens approach, image enhancement techniques such as histogram equalization or tone mapping can be applied to the visualization. Figure 11 shows a comparison of some of these alternatives to our Color Lens implementation, for an example image from the first study. [3] As can be seen from the images, some methods fail to enhance the feature of interest, while other methods clearly reveal the feature but the manipulation process has introduced severe visual artifacts that highly distort the original image, hence reducing the context awareness and violating our requirement R3. For the Color Lens, the image is undistorted within and around the focus of interest, while the context is rendered as under- and overexposed regions of the image, providing a better connection to the original image.

Given these images, we draw two conclusions: that (1) the Color Lens technique certainly can hold its own against static image enhancement techniques, and that (2) our simple extents mapping adaptation seems sufficient. Nevertheless, looking into more advanced implementations of the actual color scale adaptation is certainly interesting for future work. Again, the general Color Lens technique does not specify the method for **how** to adapt the color scale to the lens contents, and state-of-the art image manipulation techniques can certainly be integrated into the Color Lens technique, such as frequency filters, histogram equalization, and even tone mapping.

11 Conclusion

We have presented the Color Lens, a method for dynamically optimizing color scales according to user-specified regions of interest. This method addresses a well-known bottleneck in visualization: the low bandwidth and resolution that are intrinsic to the color channel. The Color Lens technique allows for a color perception gain of several orders of magnitude in the local area indicated by the user.

Compared to previous interactive approaches, the Color Lens does not require users to explicitly manipulate image parameters. Instead, users zoom into color data simply by dragging over regions of interest. Our two user studies confirmed the superiority of the Color Lens when searching into noise or photographs. We are considering investigating other tasks involving color visualizations in the future.

Acknowledgments

Thanks to Emmanuel Pietriga, Olivier Chapuis and members of the AVIZ INRIA team for their helpful comments. This work was supported in part by a grant from Microsoft Research.

3. Image b) uses a Color Lens with a simple linear rescaling of the color space; Image c) uses KamLex's PhotoShop plugin for image equalization www.kamlex.com; d) uses HDRSoft's PhotoShop plugin for tone mapping with default settings www.hdrsoft.com; e) uses Alexander Belousov's PhotoShop plugin for multiple histogram equalization with a tile size of 64 pixels www.unicontel.com; f) uses Photoshop's high-pass filter with a radius of 4 pixels followed by automatic levels.

REFERENCES

[1] G. Wyszecki and W. S. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae*, 2nd ed. John Wiley and Sons, 2000.

[2] J. Tumblin and H. E. Rushmeier, "Tone reproduction for realistic images," *IEEE Computer Graphics and Applications*, vol. 13, no. 6, pp. 42–48, Nov. 1993.

[3] S. N. Pattanaik, J. A. Ferwerda, D. A. Greenberg, and M. D. Fairchild, "A multiscale model of adaptation and spatial vision for realistic imaging," in *Computer Graphics (ACM SIGGRAPH '98 Proceedings)*, 1998, pp. 287–298.

[4] D. A. Keim, "Information visualization and visual data mining," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 1–8, 2002.

[5] J. M. Wolfe, *Attention*. Psychology Press, 1998, ch. Visual Search, h. Pashler (ed.).

[6] R. Amar, J. Eagan, and J. Stasko, "Low-level components of analytic activity in information visualization," in *Proceedings of the IEEE Symposium on Information Visualization*, 2005, pp. 111–117.

[7] K. Devlin, "A review of tone reproduction techniques," Department of Computer Science, University of Bristol, Tech. Rep. CSTR-02-005, November 2002. [Online]. Available: http://www.cs.bris.ac.uk/Publications/Papers/1000680.pdf

[8] X. Yuan, M. X. Nguyen, B. Chen, and D. H. Porter, "HDR VolVis: High dynamic range volume visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 433–445, 2006.

[9] B. E. Rogowitz, L. A. Treinish, and S. Bryson, "How not to lie with visualization," *Computational Physics*, vol. 10, no. 3, pp. 268–273, 1996.

[10] T. Halverson, "Integrating models of human-computer visual interaction," in *Extended Abstracts of the ACM CHI 2006 Conference on Human Factors in Computing Systems*, 2006, pp. 1747–1750.

[11] T. Halverson and A. J. Hornof, "A minimal model for predicting visual search in human-computer interaction," in *Proceedings of the ACM CHI 2007 Conference on Human Factors in Computing Systems*, 2007, pp. 431–434.

[12] ——, "Link colors guide a search," in *Extended Abstracts of the ACM CHI 2004 Conference on Human Factors in Computing Systems*, 2004, pp. 1367–1370.

[13] H. Seetzen, W. Heidrich, W. Stuerzlinger, G. Ward, L. Whitehead, M. Trentacoste, A. Ghosh, and A. Vorozcovs, "High dynamic range display systems," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 760–768, Aug. 2004.

[14] J. J. Vos, "Disability glare—a state of the art report," *CIE Journal*, vol. 3, no. 2, pp. 39–53, 1984.

[15] P. Ledda, G. Ward, and A. Chalmers, "A wide field, high dynamic range, stereographic viewer," in *Proceedings of GRAPHITE*, 2003, pp. 237–244.

[16] G. Spencer, P. Shirley, K. Zimmerman, and D. P. Greenberg, "Physically-based glare effects for digital images," in *Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques*, 1995, pp. 325–334.

[17] A. Scheel, M. Stamminger, and H.-P. Seidel, "Tone reproduction for interactive walkthroughs," *Computer Graphics Forum*, vol. 19, no. 3, pp. 301–311, 2000.

[18] M. A. Harrower and C. A. Brewer, "ColorBrewer.org: An online tool for selecting color schemes for maps," *The Cartographic Journal*, vol. 40, no. 1, pp. 27–37, 2003.

[19] H. Levkowitz and G. T. Herman, "Color scales for image data," *IEEE Computer Graphics and Applications*, vol. 12, no. 1, pp. 72–80, Jan. 1992.

[20] E. Bertini, A. D. Girolamo, and G. Santucci, "See what you know: Analyzing data distribution to improve density map visualization," in *Proceedings of the IEEE VGTC/Eurographics Symposium on Visualization*, 2007, pp. 163–170.

[21] A. Majumder and S. Irani, "Contrast enhancement of images using human contrast sensitivity," in *Proceedings of the ACM Symposium on Applied Perception in Graphics and Visualization*, 2006, pp. 69–76.

[22] J. Kuang, G. M. Johnson, and M. D. Fairchild, "iCAM06: A refined image appearance model for HDR image rendering," *Journal of Visual Communication and Image Representation*, vol. 18, no. 5, pp. 406–414, 2007.

[23] P. Ledda, A. Chalmers, T. Troscianko, and H. Seetzen, "Evaluation of tone mapping operators using a High Dynamic Range display," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 640–648, Jul. 2005.

[24] P. Ledda, A. Chalmers, and H. Seetzen, "HDR displays: a validation against reality," in *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, 2004, pp. 2777–2782.

[25] P. Rheingans and B. Tebbs, "A tool for dynamic explorations of color mappings," in *Proceedings of the ACM Symposium on Interactive 3D Graphics*, 1990, pp. 145–146.

[26] P. K. Robertson, "Visualizing color gamuts: a user interface for the effective use of perceptual color spaces in data displays," *IEEE Computer Graphics and Applications*, vol. 8, no. 5, pp. 50–64, Sep. 1988.

[27] P. Rheingans, "Are we there yet? Exploring with dynamic visualization," *IEEE Computer Graphics and Applications*, vol. 22, no. 1, pp. 6–10, 2002.

[28] M. Tobiasz, A. Henderson, S. Carpendale, A. Dunning, and P. Woodrow, "Developing colour sequences for high dynamic range data," in *Poster Proceedings of the IEEE Symposium on Information Visualization*, 2007, pp. 120–121.

[29] G. W. Furnas, "Generalized fisheye views," in *Proceedings of the ACM CHI'86 Conference on Human Factors in Computing Systems*, 1986, pp. 16–23.

[30] M. Colbert, E. Reinhard, and C. E. Hughes, "Painting in high dynamic range," *Journal of Visual Communication and Image Representation*, vol. 18, no. 5, pp. 387–396, 2007.

[31] V. Software, "Magic Lens Max," http://www.visionsuit.com/, 2007.

[32] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. DeRose, "Toolglass and Magic Lenses: The see-through interface," in *Computer Graphics (ACM SIGGRAPH '93 Proceedings)*, vol. 27, Aug. 1993, pp. 73–80.

[33] K. Perlin and D. Fox, "Pad: An alternative approach to the computer interface," in *Computer Graphics (ACM SIGGRAPH '93 Proceedings)*, 1993, pp. 57–64.

[34] M. Beaudouin-Lafon, "Instrumental interaction: an interaction model for designing post-WIMP user interfaces," in *Proceedings of the ACM CHI 2000 Conference on Human Factors in Computing Systems*, 2000, pp. 446–453.

[35] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Proceedings of the IEEE Symposium on Visual Languages*, 1996, pp. 336–343.

[36] T. Saito and T. Takahashi, "Comprehensible rendering of 3-D shapes," *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, vol. 24, no. 4, pp. 197–206, 1990.

[37] B. McDonnel and N. Elmqvist, "Towards utilizing GPUs in information visualization: A model and implementation of image-space operations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1105–1112, 2009.

[38] J.-D. Fekete, "The InfoVis Toolkit," in *Proceedings of the IEEE Symposium on Information Visualization*, 2004, pp. 167–174.

[39] K. Perlin, "Improving noise," in *Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques*, 2002, pp. 681–682.

[40] B. Shneiderman, "Direct manipulation: A step beyond programming languages," *Computer*, vol. 16, no. 8, pp. 57–69, 1983.

Niklas Elmqvist is an assistant professor in the School of Electrical and Computer Engineering at Purdue University in West Lafayette, IN, USA. Having joined Purdue in Fall 2008, he was previously a Microsoft Research postdoctoral researcher in the AVIZ team of INRIA Saclay - Île-de-France located at Université Paris-Sud in Paris, France. He received his Ph.D. in December 2006 from the Department of Computer Science and Engineering at Chalmers University of Technology in Göteborg, Sweden. His research specialization is in information visualization, human-computer interaction, and visual analytics. He is a member of the IEEE and the IEEE Computer Society.

Pierre Dragicevic received the Ph.D. degree in 2004 from Université de Nantes. He is a Research Scientist (CR1) at INRIA Saclay - Île-de-France, in Orsay, south of Paris, and is a member of the AVIZ team, which focuses on data analysis and visualization research.

Jean-Daniel Fekete is a Senior Research Scientist (DR2) at INRIA Saclay - Île-de-France in Orsay, south of Paris. He leads the AVIZ team since 2007, which focuses on data analysis and visualization research. The AVIZ team is located in and collaborates with the Computer Science Department (LRI) at Université Paris-Sud. Jean-Daniel's research topics include network visualization, evaluation of information visualization systems, and toolkits for user interfaces and information visualization. His research is applied in fields such as biology, history, sociology, digital libraries, and business intelligence. He is a member of the IEEE.