A Data-Driven Approach to Hue-Preserving Color-Blending

Lars Kühne, Joachim Giesen,
Zhiyuan Zhang, *Student Member, IEEE*, Sungsoo Ha, and Klaus Mueller, *Senior Member, IEEE*

Abstract—Color mapping and semitransparent layering play an important role in many visualization scenarios, such as information visualization and volume rendering. The combination of color and transparency is still dominated by standard alpha-compositing using the Porter-Duff over operator which can result in false colors with deceiving impact on the visualization. Other more advanced methods have also been proposed, but the problem is still far from being solved. Here we present an alternative to these existing methods specifically devised to avoid false colors and preserve visual depth ordering. Our approach is data driven and follows the recently formulated knowledge-assisted visualization (KAV) paradigm. Preference data, that have been gathered in web-based user surveys, are used to train a support-vector machine model for automatically predicting an optimized hue-preserving blending. We have applied the resulting model to both volume rendering and a specific information visualization technique, illustrative parallel coordinate plots. Comparative renderings show a significant improvement over previous approaches in the sense that false colors are completely removed and important properties such as depth ordering and blending vividness are better preserved. Due to the generality of the defined data-driven blending operator, it can be easily integrated also into other visualization frameworks.

Index Terms—Color blending, hue preservation, knowledge-assisted visualization, volume rendering, parallel coordinates.

1 Introduction

Color mapping and transparency are both frequently used in visualization. A color can be described by the values for luminance, hue and saturation. While luminance is mainly responsible for detection of shape and state of movement [1], the sensitivity to the chromatic part of a color stimulus, i.e., hue and saturation, allows humans to visually distinguish objects that are otherwise identical. Hence, color mapping, i.e. assigning a hue value to scalar data, is most often used for visual grouping and labeling of nominal data. In applications like illustrative parallel coordinates [2] and volume rendering, color mapping is combined with transparency in order to avoid occlusions. The most popular method for simulating transparency is the source-over operator from the collection of the Porter-Duff operators [3], which creates a new color C as a weighted sum of two input colors A and B,

$$C = A \cdot \alpha + B \cdot (1-\alpha),\ \alpha \in [0,1].$$

The over-operator is also referred to as alpha-compositing and illustrated in Figure 1.

A drawback of alpha-compositing is the creation of false colors, i.e., the hue of C is usually not one of A or B. This is problematic when hue is used to encode nominal data. For example, blending between a vivid red and green results in a slightly less saturated yellow. A hue-preserving color blending scheme would choose C having either the hue of A or B, but not necessarily the luminance and saturation values of A and B. In Figure 1 the hues are indicated by red line-segments. False colors can diminish the value of a visualization by creating artifacts when C has a hue that exists already in the given color palette, but also when the hue is not present in the given palette. False colors with a valid interpretation in the defined color palette can lead to misinterpretations, e.g., as a material property of a certain type rather than the overlapping of two different materials. False colors whose hues are not present in the given color palette may introduce visual clutter.

The goal of this work is to build a hue-preserving color blending operator, that, while eliminating false colors, perceptually preserves important information present in layered data, in particular depth information. Also, the blending result should be perceptually close to the result from alpha-compositing in terms of vividness. However, the goals of hue-preservation and perceptual closeness to the result of the Porter-Duff source-over operator are conflicting. Hue-preservation can by definition be achieved by using colors close to, or exactly on the gray axis as the blending result, i.e., by desaturation. But as colors get more and more desaturated, the perception of hue information is also diminished, and vanishes entirely on the neutral gray axis. The hue-preserving color blending problem deals with the trade-off between the two goals. Recent approaches [4, 5] tackle the problem with an a-priori solution, but have not yet found the optimal balance between the two objectives. Therefore, we propose a data driven approach, that strives for capturing important color aspects worth pertaining, while preserving the hue of the input colors. The proposed solution is a generic adaption of the standard Porter-Duff source-over operator, and therefore easy to integrate into any visualization application that uses alpha-blending, but requires hue preservation.

This paper is organized as follows. In the next section we discuss the state-of-the-art in hue-preserving color blending and work related to our data driven approach, especially the knowledge-assisted visualization paradigm. In Section 3 we briefly review the basics of color theory and also support vector machines, a machine learning tool that we have used for data analysis. In Section 4 we state the problem of hue-preserving color blending formally and describe the data driven approach and the visual interface that we have used to collect data from users. In Section 5 we provide some information on how we have collected data for generating an actual blending operator. The analysis of the data is then described in Section 6, i.e., it is described how to compute a blending operator from the data. We have applied the resulting blending operator in two applications, illustrative parallel coordinates and volume rendering. Both applications are described in Section 7. We conclude the paper with a discussion of our results in Section 8 and some concluding remarks in Section 9.

2 Related work

Wang et al. [4] explain the problem of false colors in the context of constructing a color palette for illustrative visualization. They point out that the problem of false colors can be circumvented by blending opposing colors on a color wheel. If it is not possible to choose the

- *Lars Kühne and Joachim Giesen are with the Institute of Computer Science at Friedrich-Schiller-Universität Jena, E-mail: Lars.Kuehne@uni-jena.de and Joachim.Giesen@uni-jena.de.*
- *Zhiyuan Zhang, Sungsoo Ha, and Klaus Mueller are with the Visual Analytics and Imaging Lab, Computer Science Department, Stony Brook University, E-mail: zhiyuzhang@cs.sunysb.edu, sunha@cs.sunysb.edu, mueller@cs.sunysb.edu.*

Manuscript received 31 March 2012; accepted 1 August 2012; posted online 14 October 2012; mailed on 5 October 2012.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

colors that have to be blended opposite of each other on a color wheel, then they suggest to reduce the saturation component of either the front or the back color, depending on which layer contains more important information. In a second implementation, the color is only changed within the region of overlap (see Figure 2). Wang et al. show the effectiveness of their approach on applications in medical volume rendering and illustrative parallel coordinates. However, false colors have not been eliminated entirely in these applications, since eventually the algorithm is still carrying out standard alpha-blending. Another drawback of the approach is that parameters have to be optimized by hand, which is a complex task even for a moderate number of transparent layers. Moreover, the perceived ordering of the layers has not been preserved in all cases. This can be mitigated though by manually tuning the opacity of the layers and the lightness of the colors.

In follow-up work Chuang et al. [5] have developed a generic blending operator that is also based on the opposing-color scheme, in order to eliminate false colors. Their parameter free algorithm automatically determines the less dominant color of the two colors that have to be blended, and rotates it such that it gets the opposite hue of the more dominant color, while preserving saturation and adapting lightness. The subsequent linear interpolation results in a color with one of the original hues (see Figure 2). While perfectly preserving the hues of the given color palette, this method often produces blending results near the gray axis, which is a problem for both labeling the layers and determining their depth order.

Urness et al. [6] have introduced an alternative to the standard Porter-Duff blending operators for conveying transparency. Their approach preserves the original color palette by displaying the colors of overlapping layers side-by-side instead of mixing them. The color of a pixel is chosen from the colors of the layers covering the pixel, while the choice follows a *weaving pattern*. A user study by Hagh-Shenas et al. [7] shows that observers can grasp multivariate information better when color weaving is used instead of standard color blending. But the performance gain decreases with an increasing number of overlapping layers. A drawback of the method is that so far the weaving pattern has to be selected manually. Recently, the weaving technique has been extended to scatter plots by Luboschik et al. [8].

Blending overlapping colored, semi-transparent layers is usually only a minor part of a visualization system. Hence, a solution to hue preserving color blending should not introduce many or any additional parameters. Ideally, parameter values for a particular rendering method and dataset are determined automatically. A relatively recent approach to avoid tedious parameter tuning is the knowledge-assisted visualization (KAV) paradigm. Chen et al. [9] propose a transition from interactive visualization and information-assisted visualization to knowledge-assisted visualization that decreases the effort to explore the visualization parameter space substantially. They envision a framework where domain expert knowledge is centrally stored to be shared in the visualization community. In addition, automated reasoning is proposed as a way to synthesize knowledge by observing visualization processes, and applying semantic computing and machine learning techniques to implement case-based reasoning. Visualization techniques that incorporate the knowledge-assisted visualization paradigm include [10, 11, 12].

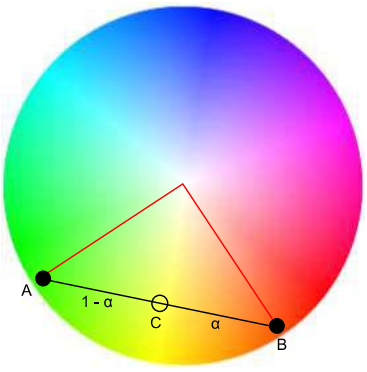

Fig. 1. Alpha-compositing. Two colors A and B create a false color C in a slice of the *HSV* representation of the *sRGB* color space of equal luminance (color circle). The hue denotes an angle in the color circle.

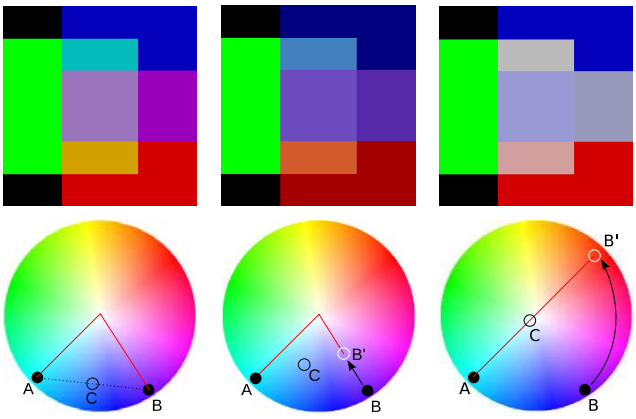

Fig. 2. Operator comparison. Shown are the Porter-Duff source-over operator (left, linear RGB), the blending operators by Wang et al. (middle, sRGB) and Chuang et al. (right, linear RGB). On top are shown three semitransparent squares on black background rendered with the respective method, while the images at the bottom illustrate each technique for blending colors A and B to create color C.

3 Background

3.1 Color space theory

A single electromagnetic color stimulus can be quantified as a vector in some color space. One of the most frequently used color spaces in digital image processing is the *sRGB* color space [13] that defines every color as a convex combination of three pre-defined primary colors $\mathfrak{R}, \mathfrak{G}$, and $\mathfrak{B}$. The set of all convex-combinations

$$\left\{ r \cdot \mathfrak{R} + g \cdot \mathfrak{G} + b \cdot \mathfrak{B} \;\middle|\; r+g+b = 1, r,g,b \geq 0 \right\}$$

is called the induced *color gamut* and contains all color stimuli representable for a particular choice of primaries. A color from the color gamut can be represented by its coordinate vector $[r,g,b]^T$. The *sRGB* coordinates c_{sRGB} of this point are in general not $[r,g,b]^T$, but still convex coordinates that are obtained after a non-linear transformation called gamma-correction. In 1931, the CIE (International Commission on illumination) proposed the *XYZ* color space, based on psychovisual experiments, with a set of primaries, such that the color gamut contains all colors that can be perceived by the human visual system.

More intuitive representations of colors are given by their cylindrical coordinates induced by a transformation from *sRGB* into *HSL* or *HSV* color space, respectively. The transformation aligns the grey axis within the *RGB*-cube either with the L (lightness) or the V (value) axis, to represent the achromatic part of a color stimulus independently. The hue (H) stands for an angle in the plane perpendicular to the grey axis, thus specifying a color circle for every particular L or V value, respectively. The exact definition of saturation (S) differs between *HSV* and *HSL* (see [14]).

The Euclidean distance in both the *sRGB* and *XYZ* space is an insufficient means to determine color similarities, since these spaces are not inherently equidistant according to the color distance function of the human visual system. Therefore, the CIE proposed the perceptual color space *CIELab* in 1976, that transforms *XYZ* coordinates non-linearly to account for non-linearities in the visual response of the human visual system. Hence, color distances Δ_E given as the Euclidean distance between two *CIELab* vectors conform better with the perceived distances. This conversion is parameterized by a reference white point in *XYZ* coordinates, where *D50* and *D65* are most frequently used in practice. Analogously to *HSV* for *sRGB*, *CIELCh*

stands for the representation of the *CIELab* space in the cylindrical coordinates lightness, chroma and hue. Subsequent improvements on color distance measures include the $\Delta_{E_{94}}$ and $\Delta_{E_{00}}$ color distance metrics, that introduce more complex distance formulas to compensate for errors in the Δ_E metric.

3.2 Support Vector machines

Support Vector Machines (SVMs) are a popular family of machine learning techniques for supervised classification (and regression) problems. Given a data matrix $X \in \mathbb{R}^{m \times n}$, i.e., m data points x_i in dimension n, together with a vector of labels $Y \in \mathfrak{L}^m$, where the i'th entry in Y corresponds to the sample in the i'th row in X, the task is to learn a predictor that associates to every point in $\mathbb{R}^n$ a label from the label space $\mathfrak{L}$. If $\mathfrak{L}$ is a finite set, then the learning problem is called a classification problem, and if $\mathfrak{L} = \mathbb{R}$, then it is called a regression problem. The special case when $\mathfrak{L}$ contains only two elements is called a binary classification problem.

Support vector machines are based on sound principles (see the book by Steinwart and Christmann [15]) and are phrased as optimization problems. A binary classifier, i.e., the case where $\mathfrak{L} = \{-1, 1\}$, can be computed from the solution of the convex quadratic program

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^{m} \xi_i$$
$$\text{s.t. } y_i(w^T x_i + b) \geq 1 - \xi_i,\ \xi \geq 0$$

as the following function

$$\mathbb{R}^n \ni x \mapsto \text{sign}(w^T x + b). \tag{1}$$

Extensions from binary classification to multi-class classification and regression are well known [16]. Support vector machines and similarly also support vector regression basically compute optimized hyperplanes as predictors or regressors, respectively. But they can be turned non-linear by applying the so called kernel trick (see for example the book by Schölkopf and Smola [16]). The kernel trick replaces the standard Euclidean scalar product by a positive kernel. Important positive kernels are the Gaussian kernel

$$k(x_i, x_j) = \exp\left(-\gamma \left\|x_i - x_j\right\|^2\right) \text{ with } \gamma > 0,$$

and inhomogenous polynomial kernels $k(x_i, x_j) = (x_i^T x_j + c)^d,\ c > 0$.

4 The Data Driven Approach

State of the art techniques for hue-preserving color blending, as discussed in Section 2, take a hypothesis based approach. In contrast to that we follow a data-driven approach inspired by the principles of knowledge-assisted visualization (KAV) [9]. One can think of a data-driven approach as a KAV-based solution, where knowledge (about optimal parameter values for a particular visualization technique) is gathered implicitly rather than explicitly. That is, we do not aim for explicit (externalized) domain expert knowledge, but elicit data from users on examples of blending problems. That also means that we pursue an active data collection strategy, i.e., we do not collect data by passively observing users while working with visualizations that involve color blending, but ask them actively about their preferences by using specifically designed interfaces to collect the data.

4.1 Problem specification

Before describing how we have collected data implicitly comprising the domain knowledge and preferences for hue-preserving color blending, we first want to specify the question that has to be answered from the data.

Generally, rendering a scene with multiple colored semi-transparent layers results in pixels that represent several layers and the question is how to integrate the information from the different layers at a given pixel. Hadwiger et al. [17] point out that when looking at a real scene, color information is accumulated iteratively at every physical layer before it reaches the observer's eye. Hence, also for rendering the case of multiple layers can be reduced to an iterative sequence of two-layer problems. This still leaves two options: the information can be accumulated either from the front to the back layer, or the other way around. The two options lead to two compositing schemes:

$$c_r = c_f \cdot \alpha_f + (1 - \alpha_f) \cdot c_b \tag{2}$$
$$c_r = c_f \cdot \alpha_f + c_b \cdot (1 - \alpha_f) \cdot \alpha_b, \quad \alpha_f := \alpha_f + (1 - \alpha_f) \cdot \alpha_b \tag{3}$$

, where Equation 2 states the back-to-front and Equation 3 the front-to-back compositing scheme. Where c_f, α_f and c_b, α_b are the colors and alpha-values of the front- and the back layer, respectively. Both compositing schemes are technically equivalent and in general not hue preserving. Hue-preserving color blending seeks to find $\hat{c}_r$ close to c_r whose hue (H value) is among or close to one of the hues of the different layers. Hence, the problem becomes to find a hue-preserving mapping

$$[c_f, c_b, \alpha_f] \quad \text{or} \quad [c_f, c_b, \alpha_f, \alpha_b] \mapsto \hat{c}_r.$$

Since any additional dimension of the problem space increases the data-gathering effort in a data driven approach, we specify every problem instance by only three parameters, i.e., by a vector $[c_f, c_b, \alpha_f]$.

4.2 Solution space exploration

In principle, there are at least two data driven strategies for solving the problem of false colors, i.e., to define or compute a mapping $[c_f, c_b, \alpha_f] \mapsto \hat{c}_r$, namely:

1. Transforming the input vector $[c_f, c_b, \alpha_f]$ into some vector $[\hat{c}_f, \hat{c}_a, \hat{\alpha}_f]$ based on user data and applying the standard blending formula 2 to the transformed vector. This approach is similar to the one taken Wang et al. and Chuang et al. (see Section 2).

2. Inferring the color c_r directly from the user data without the additional evaluation of a blending formula.

We have pursued both strategies and want to refer to the first strategy as *Problem Transformation* and to the second strategy as *Direct Solution*. In the following we describe how we have implemented the two strategies and why the second strategy works better for us.

4.2.1 Problem transformation

We have designed an interface (see Figure 3) for data gathering that gives the user full control over the solution space $[\hat{c}_f, \hat{c}_b, \hat{\alpha}_f]$. Here the solution space coincides with the problem space since we want to transform a given problem instance to another problem instance to which the standard blending formula is then applied. The interface has controls to manipulate the value (V) and the saturation (H) in the *HSV* space for both input colors c_f and c_b, and one control for modifying the alpha value α_f (since α_b can be assumed to be 1 in back-to-front compositing). Hence, the interface has five controls in total. Through

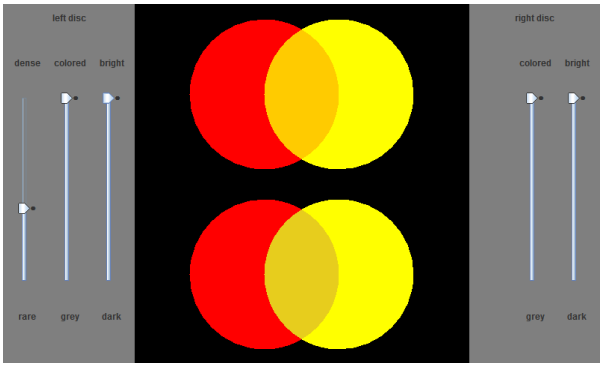

Fig. 3. A prototype interface. This version was not used for the final data collection.

the controls the user can transform the given problem instance, namely two overlapping disks with colors c_f and c_b, respectively, that are presented on top in the middle of the interface. The color in the overlap of the two disks in the problem instance is obtained from applying the standard blending formula for $[c_f, c_b, \alpha_f]$. The second pair of overlapping disks at the bottom in the middle of the interface shows the same disks, but now the color in the overlap region is computed by the standard formula for the transformed instance $[\hat{c}_f, \hat{c}_a, \hat{\alpha}_f]$ as specified by the user. The task for the user is to modify the original problem instance such that the overlap region for the lower pair of disks does not show a false color and stays perceptually close in terms of depth ordering and vividness to the original blending on top.

It turned out that the five-dimensional solution space is too large to be well explored even for users that are familiar with the topic of false colors or color science in general. Also, the impact of each parameter on the final blending is oftentimes not clear since the impact is only indirect through the evaluation of the blending equation. The high cognitive workload on the user results also in a long time to acquire just a single data point.

A possible way to reduce the search space is to reduce the number of dimensions, e.g., by fixing of the transparency channel α_f, or to subsample each dimension. Subsampling can turn the continuous solution space finite, and the problem of the solution space becomes a finite choice problem, i.e., picking the best solution from a finite choice set. The interface to choose from such a choice set can be simplified with techniques that we describe in Section 4.2.2, but it would still be more complex to be used effectively than the solution that we describe next.

4.2.2 Direct solution

For the direct solution we also use a finite choice approach, but now without evaluating the blending equation. Thus the choice set, i.e., a finite set of colors $\hat{c}_r$ for each problem instance $[c_f, c_b, \alpha_f]$, has been constructed differently than in Section 4.2.1. To construct the choice set we exploit the fact that only a "small" subspace of the color space qualifies as possible answer, namely the set of colors with either one of the hues of c_f or c_b. In color spaces with cylindrical coordinates, like *CIELCh*, all colors with the same hue lie on a plane spanned by the lightness and chromaticity axes.

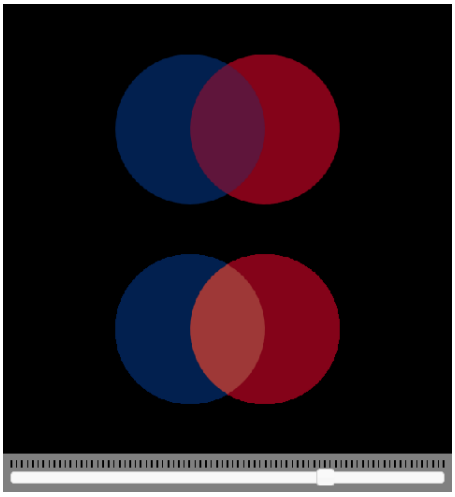

Fig. 4. Final user-interface. This version was used to collect the preference data.

We have designed an interface (see Figure 4) that allows to explore the choice set with lower cognitive effort than required for the problem transformation approach. As before, the overlapping disc pair at the top provides a reference to the standard alpha-compositing blending result. The lower disc pair renders the blending color in the overlap region computed directly from the user's selection. Therefore, the interface has now only one control that allows to browse along the choice set in a linear fashion, in order to keep the cognitive workload of the 2D exploration as low as possible. For that, the colors in the choice set have been ordered linearly, such that colors that are close with respect to the linear order are also perceptually close (using the Δ_E metric). We have determined that this allows the user to narrow the potential best choices from the choice set down quickly, by ruling out candidates that lie far from potential candidates. Hereby, the challenge is to order a sampling of a two-dimensional subspace of the color space linearly. We have addressed this by sampling the two-dimensional subspace along a continuous space filling curve, more specifically a Hilbert curve [18] (see Figure 5).

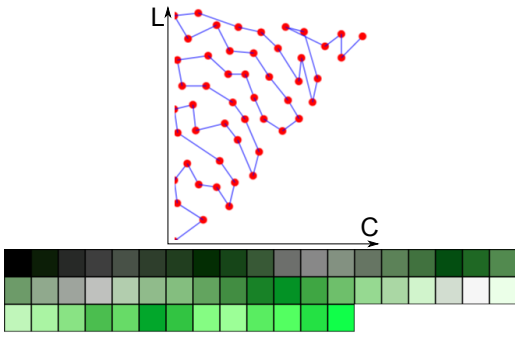

Fig. 5. Locality preserving linear ordering of samples from a hue plane (here green). Shown at the top are the samples and their ordering along a space filling curve from the lower-left to the upper-right corner. Shown on bottom is the resulting order of the colors laid out in rows from top to bottom.

Besides locality preserving exploration of the two-dimensional sample space along a path, the interface allows for directly jumping to distant colors. This specifically includes user marked colors, that were considered as potential best choices during the exploration, in order to get back to them quickly at any time, if necessary.

5 Data Collection

We have collected data points of the form $\left([c_f, c_b, \alpha_f], \hat{c}_r\right)$ over the Web using the interface shown in Figure 4. The data points will be used later to learn a good mapping $[c_f, c_b, \alpha_f] \mapsto \hat{c}_r$.

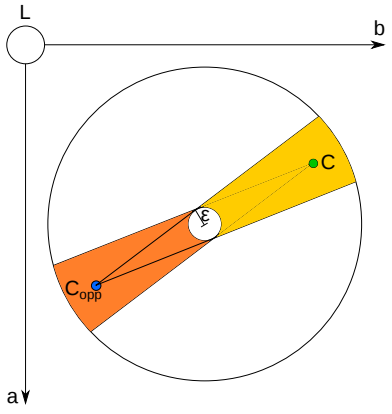

Fig. 6. A cross-section of *CIELab* space orthogonal to the lightness axis. The disk at the center with radius ε defines a color region that appears achromatic to the human visual system.

In the following we first describe a strategy that we have employed to reduce the number of necessary data points, before we discuss their collection. As has been discussed already by Wang et al. [4], colors that lie on opposite sides of the color circle do not result in false colors when they are blended together, because the two hue axes coincide on a common line that intersects with the gray axis. Although exact

opposition is rarely the case because of rounding errors or a random sampling, the idea of Wang et al. can still be used to reduce the number of sample points, i.e., problem instances. The idea is, that if the line segment that connects two colors in a color circle that need to be blended intersects a small disk centered at the center of the color circle (which lies on the gray axis), then the hue of the result of alpha-blending the two colors cannot be distinguished perceptually from one of the two original hues (see Figure 6).

Of course the radius ε of the disk needs to be small enough to guarantee this property, that represents a continuous extension to the discrete evaluation of the opposite color predicate. We chose the radius ε such that the colors in the disk seem achromatic to the human visual system. The intuition behind this approach is that all colors in the truncated cone whose apex is C and whose opening angle is determined by the disk with radius ε (see again Figure 6) can be considered perceptually opposite to C. The cone is truncated at the line that is passing through the center of the color circle and is orthogonal to the line segment from C to C_{opp}, where C_{opp} is an actual opposing color of C. The truncated cone is shown in orange in Figure 6, and the analogous truncated cone for C_{opp} is shown in yellow. The tolerance disk with radius ε allows to define predicates that given two colors, determines their relationship in terms of perceptual *opposition* and *equivalent hue-angle*. With these predicates it can be decided if the blending result $\hat{c}_r$ of a problem instance $[c_f, c_b, \alpha_f]$ can be computed by alpha-compositing (without creating precptually false colors), or has to be inferred from the collected data. The decision rules are as follows:

- If α_f is either 0 or 1, then the blending result is either c_b or c_f, respectively, and thus can be set directly
- If c_f and c_b are opposing colors; this can be decided by checking if the line-segment connecting the two colors intersects the tolerance disk with radius ε.
- If the colors c_f and c_b have the same hue; this can be decided by creating C_{opp} for c_f (see once more Figure 6), and performing the color opposition test for C_{opp} and c_b.

Note, that if one of the colors c_f or c_b is perceptually achromatic, then the opposite-color-check will always evaluate positive, and thus the blending can be computed directly. Blending with an achromatic color does not create any false colors, thus adding achromatic colors to the survey samples does not increase the complexity of the survey, however, this stabilizes the learned model along the lightness axis. Also, we did not exploit the symmetry between $[c_f, c_b, \alpha_f]$ and $[c_b, c_f, 1-\alpha_f]$ in the standard alpha-compositing setting, because it is not obvious, why such a symmetry should hold for hue-preserving color blending.

It remains to describe how we have determined the radius ε of the tolerance disk. For that we have conducted another web based survey. We have sampled *CIELCh* space close to the gray axis at six equidistant hue angles, and along each hue angle in steps of 4 chromaticity units from 4 to 20, and at six equidistant lightness levels from 0 to 100 units on the *CIELCh* lightness scale. Each of these 180 sample points was presented to participants in our survey together with sample points with similar lightness values that have been sampled exactly from the gray axis. The participants had to identify the unique chromatic sample point among 16 sample points that were presented in a grid of 4×4 colored patches. The radius ε of the tolerance disk has then been chosen such that the disk does not contain any sample point that has been identified correctly as the chromatic one in at least 80% of the answers for that sample point.

The data for this survey have been collected over the course of two weeks from more than 120 participants who contributed a total of 2457 answers, i.e., about 20 decisions per participant and 14 answers per sample point.

We observed that the chromaticity-sensitivity that has been probed in this survey is hue-angle independent but depends on lightness value. Resulting values for the radius ε are listed in Table 1.

Finally, we are ready to describe how we have sampled data points from the function $[c_f, c_b, \alpha_f] \mapsto \hat{c}_r$. The problem instances $[c_f, c_b, \alpha_f]$ have to be represented in a specific color space. Since the data have been collected over the Web we cannot assume that the monitors of the participants of our survey support a color gamut that is larger than the *sRGB* color space. Hence, the sample points have been sampled randomly from *sRGB* space such that any point in *sRGB* has a sample point at distance at most 60 Δ_E . This results in 16 *sRGB* samples (including five equidistant achromatic colors added afterwards). Together with a sampling of the alpha-channel α_f in increments of 0.25, this resulted in a total of 1280 sampled problem instances. Out of these 129 had to be evaluated by participants of our Web based survey using the interface from Figure 4. The remaining instances can be blended automatically. For each of the 11 chromatic *sRGB* samples, we sampled the corresponding hue-plane in *CIELCh* space such that any point has a sample point at distance at most 15 Δ_E, leading to an average of 50 alternative colors $\hat{c}_r$ on the interface slider.

Table 1. ε-Radius for the Tolerance Disk, depending on the lightness value in *CIELCh* space.

lightness level	chroma tolerance ε
< 20	15.0
20 to 80	10.0
> 80	20.0

Over the course of 4 weeks, we have collected 1851 choices from 133 participants, most of them students and people with an academic background, however not necessarily color science experts. Initially, every participant had to answer to at least 20 choices, however later we reduced this to 10 choices, in order to reduce the cognitive workload on the participants and to obtain answers of high quality in at most 20 minutes. The participants did not evaluate any problem instance twice. On average, each of the 129 problem instances was presented to 14 participants.

6 Data Analysis

The goal of the data analysis is to learn a function

$$\mathbb{R}^7 \ni [c_f, c_b, \alpha_f] \mapsto \hat{c}_r \in \mathbb{R}^3$$

from the user feedback that has been gathered via the Web using the interface shown in Figure 4. The perceptual non-uniformity of *sRGB* can cause non-linear dependencies between the input vector and the gathered label. To circumvent this problem the color coordinates of the input colors c_f and c_b have been transformed from *sRGB* to *CIELab* coordinates, i.e any problem instance can be represented as

$$[L_f, a_f, b_f, L_b, a_b, b_b, \alpha_f]$$

using the *CIELab* coordinates L, a and b. At a first glance the learning problem at hand looks like three regression problems, namely one regression problem for each coordinate of $\hat{c}_r$ (lightness, chroma and hue). However, the hue of $\hat{c}_r$ is by definition either the hue of c_f or c_b, and thus can be determined by a binary classification. Predicting the hue angle of the blending color by regression can produce false colors, even if the prediction is only a little bit off. Thus, binary classification is both more efficient to evaluate and more reliable in the sense of hue preservation.

As mentioned before, we had 1280 problem instances in total where a problem instance now is given as point in seven dimensional Euclidean space $\mathbb{R}^7$. The number of data points that we enter into the support machine or regression machinery is higher though, because the same problem instance has been evaluated by several participants in the survey. We handle this as follows: we add one constraint to the support vector/regression optimization problem (see Section 3.2) for every problem instance and chosen alternative $\hat{c}_r$ and weigh the slack variables x_i for the i'th constraint by the number of times w_i that participants have chosen the particular alternative $\hat{c}_r$, i.e., the objective function of the support vector machine is modified as follows,

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^{m} w_i \cdot \xi_i$$

Luckily, the modified support vector machine/regression instances can still be solved using the popular, publicly available LIBSVM solver [19]. We used this solver together with polynomial and Gaussian kernels. Parameters like the regularization parameter C in the objective function of the support vector machine and kernel hyperparameters have been set using cross-validation which is supported by LIBSVM. It turned out by using cross-validation that in our case the solutions of the kernelized optimization problems outperform their linear counterparts, and the polynomial kernel slightly outperforms the Gaussian kernel.

Finally, the solutions to the two regression and one classification problems provide us with the mapping from problem instances $[c_f, c_b, \alpha_f]$ to a blending color $\hat{c}_r = [\hat{L}_r, \hat{C}_r, \hat{h}_r]$. Pixel-wise evaluation of the three functions is computationally expensive. The SVM based blending operator on our dataset is about 20 times as expensive as alpha-compositing. Therefore, we used solution pools to avoid evaluating any problem instance twice. This has been sufficient for our applications where the images have been rendered off-line. Note though, that by construction the evaluation of the three functions is easily parallelizable. Hence, we recommend a GPU based function evaluation for achieving real-time, interactive frame rates, especially in volume rendering.

7 Applications

We have integrated our data driven blending operator in an illustrative parallel coordinates framework, and have also used it for volume rendering. In both applications we have blended two layers at a time in back-to-front order, where blending results are always fed into the next blending iteration as the back layer. But before we discuss both applications in more detail, we will first discuss the performance of our blending operator in a synthetic test case.

7.1 Synthetic test case

Figure 7 compares the blending operators defined by Wang et al., Chuang et al. and our approach on a synthetic test case with three semi-transparent squares on a black background. A rendering using alpha-compositing is also shown for reference. The test case is probably the most difficult color setup in *sRGB* color space since the hues of the three layers are the three primaries of *sRGB* and thus cover all hues (angles) when blended using alpha-compositing.

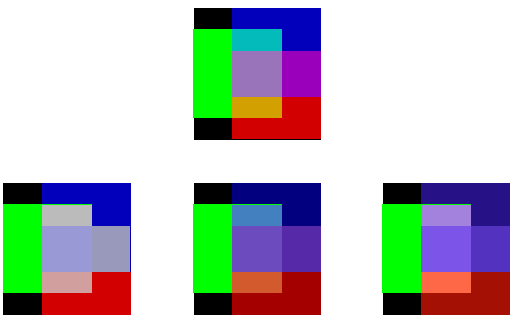

Fig. 7. Synthetic test case: (On top) default *sRGB* blending of three colored squares against a black background creating false colors; (At the bottom from left to right) same physical layout rendered using the blending operators defined by Chuang et al., by Wang et al., and our data driven blending operator.

The rendering produced by the method of Chuang et al. creates hardly distinguishable shades of gray in the overlap regions, making it difficult to determine any depth order. The method by Wang et al. performs much better in this respect, however, there not all false colors have been eliminated entirely.

Our blending result is hue-preserving by construction. However, in order to produce a consistent rendering, the classifier needs to be reliable on all the overlap regions. For example, it would be of little value if the blue hue would be predicted to lay over the red hue on the right, but below it in the central overlap region. But as can be observed this consistency property is satisfied in this test case. Also,the goal of preserving the depth-ordering has been achieved. Note that we have enforced this goal as a constraint during the Web based data collection, and this constraint carries over to the support vector models. Another notable difference to the other methods is that the lightness of the lower layers is higher. This has not been enforced as a constraint in the data collection phase, but surprisingly for us it supports the perception of the shape of the underlying layers nicely, without sacrificing the perception of the correct depth ordering.

7.2 Illustrative parallel coordinates

Figure 8 to 10 show results that have been obtained after integrating the data driven blending operator into the rendering software for illustrative parallel coordinates (IPC) by McDonnell et al. [2].

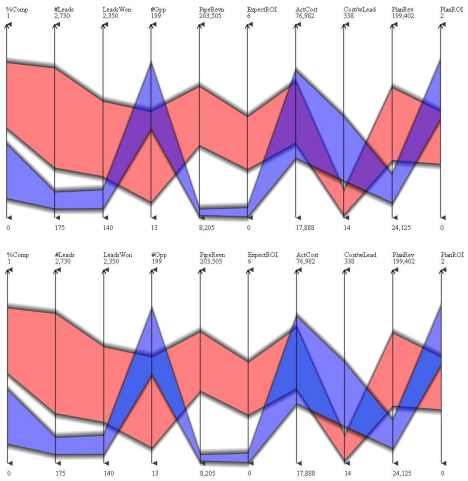

Fig. 8. A two-layer IPC with a blue layer on top and red layer in the background: (top) blending using alpha-compositing; (bottom) blending using the data-driven blending operator.

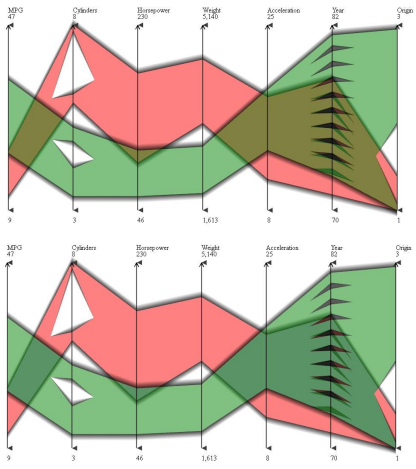

Fig. 9. A two-layer IPC with a green layer on top and red layer in the background: (top) blending using alpha-compositing; (bottom) blending using the data-driven blending operator.

Depth order preservation is crucial for the performance of IPCs, but avoiding false colors is also very important. IPCs can suffer from optical illusions when false colors are created, i.e., areas of overlap with false colors can appear as new categories that are not present in the data (geometric artefacts). Hue-preserving blending helps to reduce the perception of geometric artefacts. This is probably most notable in the three layer rendering in Figure 10.

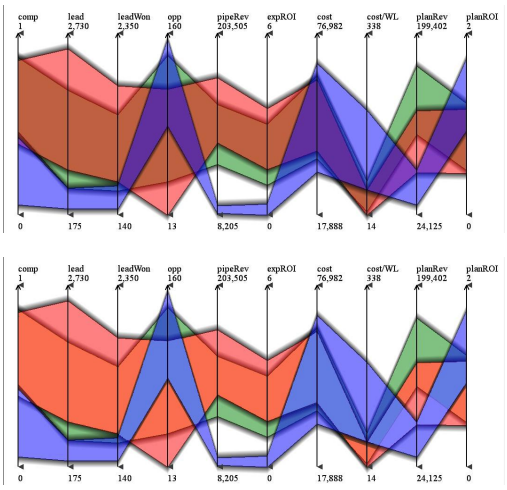

Fig. 10. A three-layer IPC with a blue layer on top, a red layer in the middle, and a green layer in the background: (top) blending using alpha-compositing; (bottom) blending using the data-driven blending operator.

7.3 Volume visualization

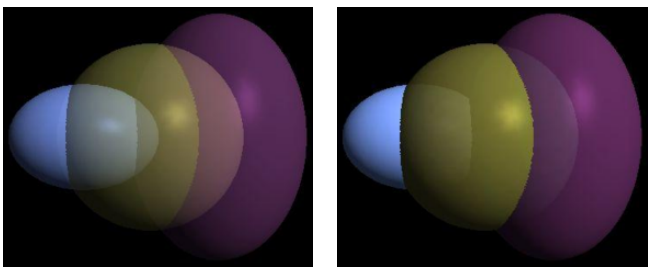

Fig. 11. Simulated dataset, that shows three ellipsoids partially including each other: (On the left) blending using alpha-compositing; (On the right) blending using the data-driven blending operator.

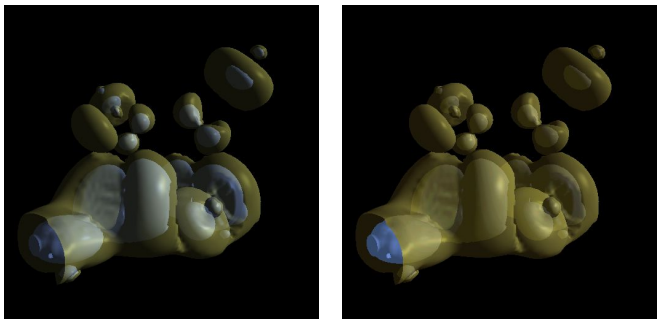

Fig. 12. Neghip dataset: (On the left) blending using alpha-compositing; (On the right) blending using the data-driven blending operator.

Finally, we have also integrated our blending operator into a volume rendering framework, and applied it to a simulated geometry, and the well known fuel and neghip data sets (see Figures 11 to 13). Also in volume rendering hue-preservation turns out to be beneficial in terms of depth perception and color vividness. But the most important observation here is that the data-driven model is able to pertain continuity along all the *CIELCh* axes on a large number of layers, in addition to the large variation in lightness between adjacent pixels which results from the lighting model that is employed by the volume renderer.

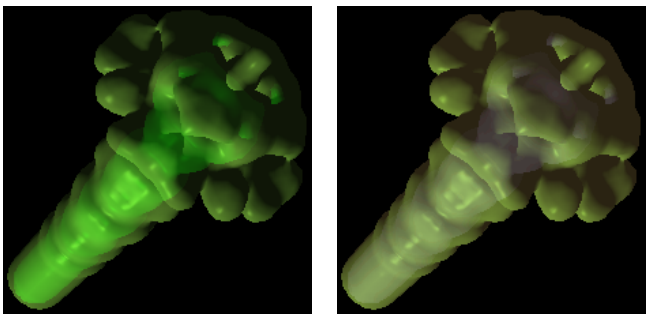

Fig. 13. Fuel dataset: (On the left) blending using alpha-compositing; (On the right) blending using the data-driven blending operator.

8 Discussion

8.1 Towards the goal

The main goal that we have stated in the introduction is a hue- and depth order preserving blending operator that is perceptually close to the standard Porter-Duff over-operator, e.g., in terms of vividness. The renderings produced by our blending operator in all the evaluated applications show that we have come closer to our goal by enforcing the preservation of the chosen color palette as a constraint in our Web-survey. These renderings allow a better understanding of the data, because they avoid the artefacts that can be introduced by default alpha-compositing. Although the constraints (hue and depth-order preservation) can be conflicting (see Section 2), our results show, especially in the case of illustrative parallel coordinate plots, that a data driven model can find a good balance between them. Also, the restriction to only a small number of layers as for the weaving approach (see the evaluation in [7]) does not apply to the data-driven operator as can be seen in the volume rendering examples.

8.2 Data collection

8.2.1 Study design

To design the web survey as a choice task proved to be an effective method to collect the required amount of data for developing a reliable blending operator. Although a single paired comparison test comes with the least cognitive burden for the user, see [20], it would not have been practical for us to capture a single sample of the blending function

$$\mathbb{R}^7 \ni [c_f, c_b, \alpha_f] \mapsto \hat{c}_r \in \mathbb{R}^3$$

in this way. In a paired comparison study, before any ranking information for the $\hat{c}_r$ can be retrieved (e.g. by Thurstone's Law of Comparative Judgement [21]), a frequency matrix F_{ij}, where each entry represents the number of times element i has been preferred over element j, would have to be filled, and therefore $n(n-1)/2$ comparisons would have to be made by every single participant in order to retrieve his preferred $\hat{c}_r$ among n alternatives. To reach statistical significance, a rule of thumb suggests at least 15 data points per free parameter, i.e., in this case $15n$ binary choices for a problem instance which in turn only provides a single function sample. Hence, the required number of comparisons would either be infeasibly large or the number of choice alternatives $\hat{c}_r$ would have to be very small. Our choice-based evaluation supported by a linear alignment of the possible choices allows for a more effective test procedure.

8.2.2 Study participants

Another important factor are the participants that take part in the survey. Usually, digital user studies/surveys are conducted in one of two ways: in a completely controlled lab environment, or over the Internet. Because of relatively high time and monetary costs of lab-based user studies, Web-based user studies have become increasingly popular. The main advantage of a Web-based user study is the large number of participants that can be recruited in relatively short time. Garg et al. [22] conducted two different online studies, where they collected

responses from 96 users in just one day, a number that is difficult to achieve in a lab-based study. However, without some sort of mediator, an online-study has several drawbacks. A small number of people can contribute a larger number of votes, and thus skew the results, without a definite way to detect this. Incentives to motivate the participants to deliver high quality results, are also more complicated to be distributed. Amazon provides a service to fill the gap of a mediator between test requesters and workers, namely Amazon Mechanical Turk (AMT). An insightful discussion about AMT for perceptual studies is given by Kosara and Ziemkiewicz [23]. Another efficient data collection method for color blending using gamification was presented by Ahmed et al. [24]. Our study for data-driven color blending aims at sampling an unknown function hidden in the human visual system. Therefore, the ground truth is not known, which would be necessary to evaluate the quality of a user's response in order to use the incentive and penalty system of AMT. That is why we decided to conduct a manually developed online-survey.

9 Conclusion

We have shown how to construct a data-driven blending operator that captures the human domain knowledge by means of support vector machine models and a back-to-front compositing scheme. Hereby, the Web-survey design was the most critical part, as it directly dictates the quality of the collected data. Main challenges when designing the survey was keeping the cognitive burden on the participants low, as well as optimizing participation in the survey. A careful selection of search-space reduction criteria and an intuitive survey-interface allowed the collection of enough data of sufficient quality. Still, the final blending operator would profit from an even larger amount of data on which the support vector machines can be trained. Equipped with a ground truth provided by our data, online-survey mediator services, like Amazon Mechanical Turk, could leverage this strategy and provide a larger data base. But already the blending operator that has been learned from the data that we have collected so far performs well in applications. We have applied the data-driven blending operator successfully to 2D data sets with few semi-transparent layers (illustrative parallel coordinates), as well as to 3D data sets (volume rendering). In both applications we have been able to improve significantly over default alpha-compositing and also over state-of-the-art techniques for hue-preserving color blending. Avoiding desaturated blending results and enforcing the preservation of the perceived depth-order already in the data collection phase, resulted in a blending operator that keeps these qualities next to being hue-preserving.

Finally, we have developed a software library that can be integrated into any visualization application that requires a hue-preserving blending operator. So far our solution is limited to offline-rendering applications, because of the computationally expensive per pixel evaluation. However, with GPU-based classifiers and regressors, interactive frame rates should be possible.

Acknowledgments

We would like to thank the participants of our user surveys for their dedication and stamina with which they supported us in our journey.

The work by Lars Kühne and Joachim Giesen has been supported by the DFG Priority Program 1335: Scalable Visual Analytics, and the CG Learning project. The project CG Learning acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 255827".

The work by Klaus Mueller, Zhiyuan Zhang, and Sungsoo Ha was supported in part by NSF grants 1050477, 0959979 and 1117132, by a Brookhaven National Lab LDRD grant, and by the Ministry of Korea Knowledge Economy, Republic of Korea.

References

[1] Colin Ware. *Information visualization: perception for design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[2] Kevin T. McDonnell and Klaus Mueller. Illustrative parallel coordinates. *Comput. Graph. Forum*, 27(3):1031–1038, 2008.

[3] Thomas Porter and Tom Duff. Compositing digital images. *SIGGRAPH Comput. Graph.*, 18:253–259, January 1984.

[4] Lujin Wang, Joachim Giesen, Kevin T. McDonnell, Peter Zolliker, and Klaus Mueller. Color design for illustrative visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14:1739–1754, November 2008.

[5] Johnson Chuang, Daniel Weiskopf, and Torsten Moller. Hue-preserving color blending. *IEEE Transactions on Visualization and Computer Graphics*, 15:1275–1282, November 2009.

[6] Timothy Urness, Victoria Interrante, Ivan Marusic, Ellen Longmire, and Bharathram Ganapathisubramani. Effectively visualizing multi-valued flow data using color and texture. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 16–, Washington, DC, USA, 2003. IEEE Computer Society.

[7] Haleh Hagh-Shenas, Victoria Interrante, Christopher Healey, and Sunghee Kim. Weaving versus blending: a quantitative assessment of the information carrying capacities of two alternative methods for conveying multivariate data with color. In *Proceedings of the 3rd symposium on Applied perception in graphics and visualization*, APGV '06, pages 164–164, New York, NY, USA, 2006. ACM.

[8] Martin Luboschik, Axel Radloff, and Heidrun Schumann. A new weaving technique for handling overlapping regions. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, pages 25–32, New York, NY, USA, 2010. ACM.

[9] Min Chen, David Ebert, Hans Hagen, Robert S. Laramee, Robert van Liere, Kwan-Liu Ma, William Ribarsky, Gerik Scheuermann, and Deborah Silver. Data, information, and knowledge in visualization. *IEEE Comput. Graph. Appl.*, 29:12–19, January 2009.

[10] Christopher Koehler and Thomas Wischgoll. Knowledge-assisted reconstruction of the human rib cage and lungs. *IEEE Comput. Graph. Appl.*, 30:17–29, January 2010.

[11] Benjamin J. Kadlec, Henry M. Tufo, and Geoffrey A. Dorn. Knowledge-assisted visualization and segmentation of geologic features. *IEEE Comput. Graph. Appl.*, 30:30–39, January 2010.

[12] Edward Swing. Prajna: Adding automated reasoning to the visual- analysis process. *IEEE Comput. Graph. Appl.*, 30:50–58, January 2010.

[13] Michael Stokes, Matthew Anderson, Srinivasan Chandrasekar, and Ricardo Motta. A standard default color space for the internet — srgb. http://www.color.org/contrib/sRGB.html, November 1996.

[14] George H. Joblove and Donald Greenberg. Color spaces for computer graphics. *SIGGRAPH Comput. Graph.*, 12:20–25, August 1978.

[15] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[16] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[17] Markus Hadwiger, Joe M. Kniss, Christof Rezk-salama, Daniel Weiskopf, and Klaus Engel. *Real-time Volume Graphics*. A. K. Peters, Ltd., Natick, MA, USA, 2006.

[18] David Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. *Mathematische Annalen*, 38:459–460, 1891.

[19] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[20] Zofia Barańczuk, Joachim Giesen, Klaus Simon, and Peter Zolliker. *Advances in Imaging and Electron Physics*, volume 160, chapter Gamut Mapping, pages 1–34. Elsevier Inc. Academic Press, 2010.

[21] Louis Leon Thurstone. *Psychological Review*, volume 34, chapter A law of comparative judgement, pages 273–286. 1927.

[22] Supriya Garg, Julia EunJu Nam, Kshitij Padalkar, Ming-Yuen Chan, Soren Laue, Waqar Saleem, Joachim Giesen, Huamin Qu, and Klaus Mueller. Kav-db: Towards a framework for the capture and retrieval of visualization knowledge over the web. In *Proceedings of the Schloss Dagstuhl Scientific Visualization Workshop 33(5) (SciVis)*, pages 607–615, 2010.

[23] R Kosara and C Ziemkiewicz. Do mechanical turks dream of square pie charts? *IEEE Transactions on Visualization and Computer Graphics*, 17(4):393 – 411, 2010.

[24] Nafees Ahmed, Ziyi Zheng, and Klaus Mueller. Human computation in visualization: Using purpose driven games for robust evaluation of visualization algorithms. *IEEE Transactions on Visualization and Computer Graphics (Special Issue IEEE Visualization)*, December 2012. *to appear*.