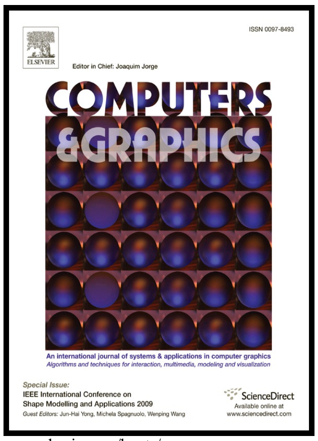


Personalized 2D color maps

Nicholas Waldin, Matthias Bernhard, Peter Rautek, Ivan Viola

PII: S0097-8493(16)30077-2
DOI: http://dx.doi.org/10.1016/j.cag.2016.06.004
Reference: CAG2725

To appear in: *Computers and Graphics*

Received date: 14 February 2016
Revised date: 17 May 2016
Accepted date: 10 June 2016

Personalized 2D Color Maps

Nicholas Waldin[a], Matthias Bernhard[a], Peter Rautek[b], Ivan Viola[a]

[a] *TU Wien*
[b] *KAUST*

Abstract

2D color maps are often used to visually encode complex data characteristics such as heat or height. The comprehension of color maps in visualization is affected by the display (e.g., a monitor) and the perceptual abilities of the viewer. In this paper we present a novel method to measure a user's ability to distinguish colors of a two-dimensional color map on a given monitor. We show how to adapt the color map to the user and display to optimally compensate for the measured deficiencies. Furthermore, we improve user acceptance of the calibration procedure by transforming the calibration into a game. The user has to sort colors along a line in a 3D color space in a competitive fashion. The errors the user makes in sorting these lines are used to adapt the color map to his perceptual capabilities.

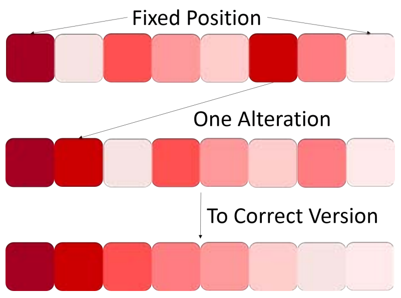
Keywords: Color, Perception, Color Vision Deficiency

1. Introduction

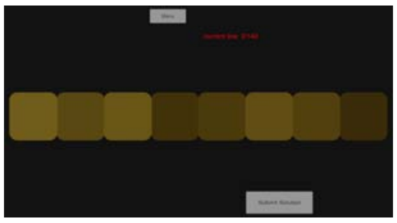
Color maps are commonly used to convey data properties in height or heat maps. In a one-dimensional case the data is mapped onto a line in a color space, for example, RGB, CIELab, or a single color that changes in intensity. In two dimensions, either a plane is taken in a color space, e.g. RGB, or each axis corresponds to a color and a data value's color is a linear combination of the axes. Comprehension of two-dimensional color maps can be curtailed if the perceptual distance between data values mapped onto the color map is not consistent with the distance between original data values [1]. Fortunately, for people with normal color vision, despite having large differences in cone ratios, color perception is fairly similar [2], so standard perceptually uniform color spaces can be used. However, for users with a form of color deficiency, such as anomalous trichromats or dichromats, conventional color maps can raise several issues. First, there is the obvious problem of being unable to distinguish certain colors, such as red and green in the case of red-green blindness. The second issue is a change in perceptual distance. Because the perceptual distance between colors that are near each other can be very short, and for people with color vision deficiencies even look the same, the user will have difficulty comprehending the data with these colors. As a result, he/she may overlook important features. This problem can be addressed by personalizing color maps. Our method extracts lines of color gradients from the color map and takes samples along it. These samples are displayed to the user as a sequence of squares, ordered randomly. The user must then sort these squares to achieve a smooth color gradient. An example of this task is shown in Figure 1b, and an example of the sorting in Figure 1a. Based on errors the user makes, we personalize the color space by contracting and expanding areas of the color map where the user performs worse or better. Through this we can personalize any continuous color space.

2. Related Work

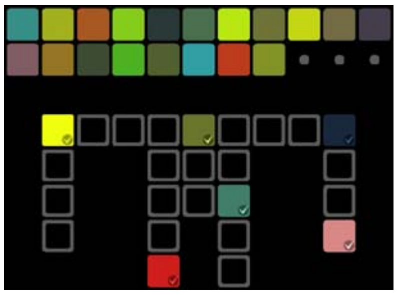
Two-dimensional color maps are used in a large variety of tasks [4], and are also used with higher dimensional data with data being projected onto a two-dimensional color map. In this case, tasks fall generally into one of two groups: either into identification and comparison of data points and clusters, or lookup of classes and clusters. For people without color vision deficiencies often perceptually uniform color maps [1] are used. The choice of color map depends on the task, and there is research on how to choose the appropriate color map [5]. An area where color map alterations have been demonstrated as useful is the remapping of colors for people with color vision deficiencies, for example encoding colors as patterns [6] and completely altering the colors of the image, both for stylistic reasons [7], [8], as well as scientific purposes [9], [10], and in real-time [11]. A longer list of examples can be found in a recent survey [12]. There are also methods for calibrating monitors to people with color deficits in order to recolor images, such as [13] and [14], which will be touched on in Section 7.3. However, the trend of personalizing visualization and color maps is gaining momentum. One approach uses human perception of faces to generate isoluminant color maps [15]. An image of a face is divided into two parts, one white, one black. The image is copied and the white and black areas reversed. The image and its copy are placed side by side and shown to the user, with the black areas set to a grey value, and the white areas to a color. Two images are used because, if the luminance is unequal, one of the faces will stand out more. The user alters the luminance of the colored area until he/she thinks the luminance of the colored area is the same as the luminance of the grey area. This is done repeatedly, each time with a different color. The settings are then used to create an isoluminant color bar. Another approach [16] shows a method whereby a one-dimensional color bar can be adapted to an in-



(a)



(b)



(c)

Figure 1: In (a) an illustration of a person sorting a line is shown. The top line is presented to the person. S/He moves individual squares in the line in an attempt to sort it. In the second line one square has been moved. In the last line all the squares have been correctly sorted. In (b) a screenshot of what is displayed in our method to the user in the center of the screen is shown. The black area between what is shown and the edge of the monitor has been cut away. The colors have been altered significantly for easier perception. (c) shows a screenshot of the game Blendoku [3] on medium difficulty. Menu, stage name, and similar have been removed.

dividual's color perception by searching for small changes in the color bar. This method can adapt a color bar in 10 minutes using 15 sample points. At each sample point the color of the sample point is taken and copied to another position along the bar. This creates a dot along the bar, which the user must search for and find within 5 seconds. While this method can be used to adapt two-dimensional color maps, it does not scale well. If we expand this method to two dimensions, this would mean about 15^2 (225) points of interest. Furthermore, working in two dimensions increases the number of directions. Because the number of directions in two dimensions is infinite, it is imperative to select the least number of directions that can still give a useful result. In the case of an arbitrary plane, all axes of the color map and their combinations are of interest, because the change in color and perceptual distances along a combination may be very different from the changes along an axis of the plane. Therefore, both the x-axis, y-axis and the diagonal directions $[x, y] = [1, \pm 1]$ should be investigated, leading to a total of 4 directions. As a result, in this approach the amount of work is first squared (number of points) and then multiplied by 4, leading to an approximate testing time of 400 minutes, or 6 hours 40 minutes. If we assume their approach uses as many points as we have areas in our experiment (20), then their approach would use $10 \times (20/15) \times 4 = 53$ minutes. However, the user will require a break. If we assume the user takes a 5 minute break after every 10 minutes, then it will take a total of 70-75 minutes. Our work has a similar goal. However, we developed a method that scales better into two dimensions, is less exhausting and more enjoyable. In contrast, our method can be completed in as little as 30 minutes, though some users take significantly longer. Furthermore, it is influenced by the game Blendoku [3], which has a huge player base, indicating that it is enjoyable. It is also not as exhausting. In the approach paper by Gresh [16], the user must constantly react within seconds. In our game the user progresses as fast as he/she wants.

3. Methodology

People dislike using software that adapts to the users because it takes time, effort, and is usually tedious and boring. Therefore we have drawn inspiration from the field of mobile gaming in order to overcome these issues. Our method is based on the game Blendoku, where the player must sort colors. In this game the player is given a figure consisting of squares. He/She must move squares with different colors onto the fields of the figure, such that the colors change from one hue to another between two different squares on the game figure. In our method, we do not include a puzzle aspect, as we only wanted to focus on the ability to distinguish colors. As such, we greatly reduced the difference in colors between tiles. A screenshot of our version can be seen in Figure 1b, and Blendoku in 1c. In our game, the user is asked to sort a sequence of eight squares, the first and last can not be moved. Using a pilot study with different color maps and methods of creating sequences, we analyzed what measurements could be used. We measured the number of incorrectly sorted squares, time spent comparing two colors, time needed to sort a sequence, and the number of times squares were moved. Only the number of incorrectly sorted squares was useful, as no other measurement correlated with it. Therefore, the method for adapting the color map uses only the number of errors made, and it attempts to alter the color map so that the areas of the color map with higher error are shrunk and areas with low error are expanded, increasing and decreasing the perceptual distance between data values. In this section, we will describe the probing and evaluation method. All RGB values are between 0 and 1 and in the format RGB = [R,G,B].

3.1. *Forced Choice Statistics*

Our method is essentially a yes-or-no forced choice method. In forced choice statistics, the user must choose between two options, in our case if a tile is to the right or left of another. The probability distribution of normal forced choice follows the curve in Figure 2. When the user can not determine which option to choose, he must guess. The threshold for noticing a difference is placed at 75%. See Mckee et al. [17] for details. In our case of a line, the function will not decline as quickly, because the user can use additional squares to compare. For example, he may not see the difference between squares 2 and 3 and squares 3 and 4, but may see a difference between 2 and 4.

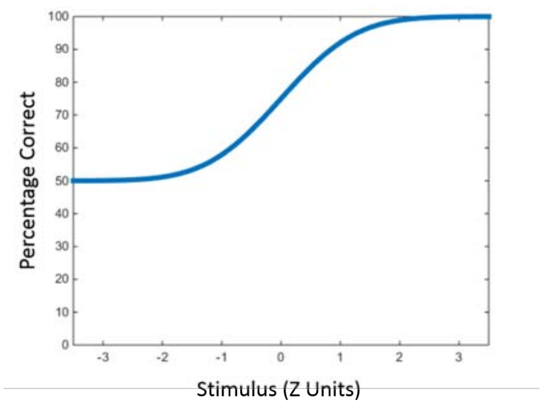


Figure 2: Illustration of the curve generated by forced choice statistics when choosing which of two items has more of some quality. Z units refer to the standard deviation, i.e., 2 means 2 standard deviations of the cumulative normal curve

It needs to be pointed out that if a person can solve a line without making a mistake, for example because it is very easy, then our method can not tell the degree of difficulty of perception in this area. Another aspect of this forced choice is the effect on an adaption of the color map based on measured user errors. If a color map is optimally adapted to a person, then the perceptual difficulty would be evenly distributed over the color map for this person, i.e., the probability of an error would be independent of the position on the color map. If we take the mean perceptual difficulty and alter all areas to have this difficulty, then we will have achieved an even distribution of perceptual difficulty. This is described in Section 3.5.

3.2. Game Structure

The user has to sort a sequence of 8 squares, which we found to be adequate through a pilot study. All squares have different colors but are equally large. The squares at the ends can not be moved, as they define the color gradient. The user can drag and move any square anywhere else in the sequence. This allows users to compare any two squares by overlapping one square with another. The sorting mechanism works in the same way as insertion sort. The user sorts the squares so the color changes smoothly from one end to the other. When he/she believes the sequence is sorted correctly he/she presses the submit button. This is repeated until all sequences are sorted.

3.3. Selection of Lines

The color map was divided up into a grid of rectangles. An equal number of lines, which have been generated at random across the color map, are randomly extracted from each rectangle. Points along the line are extracted to form a color gradient sequence. The change between squares in the color gradient sequence along the line is not necessarily the same for each step. This is due to line aliasing. Essentially, we rasterize the line in RGB space, as if the RGB space consisted of 256^3 voxels. However, the difference between two squares needs to be equivalent to two steps, e.g., [0.0039, 0, 0] + [0, 0.0039, 0] (0.0039 = 1/255). Anything smaller, i.e., only changing one value, can not be chosen because it leads to a loss of direction. The direction could change by 90 degrees from one step to the next (e.g., one step [0, 0, 0.0039], the next [0, 0.0039, 0]), which may confuse a user. This means a change along the line in RGB from one square to the next could be a change of [0.0039, 0, 0.0039] the first time, and [0, 0.0039, 0.0039] the next. Therefore, the error might not be simply along the line itself, but in the direction of a certain color. After 7 step combinations our line is created.

3.4. Evaluation of Lines

The evaluation is based on the number of errors the user makes when sorting the lines. An error occurs when a square is in the wrong position relative to another square. This means that the incorrect placement of one square can lead to multiple errors. For example, if the square that should be in position 2 is placed in position 4, there are two errors, one for the incorrect comparison with square 3 and one with square 4 (which is now in position 3). An example can be seen in Figure 3. The error consists of three components, the red, green and blue directions. For each line the errors in these three directions are added up. When a mistake is made, the error for this mistake is divided up into its RGB components and added to these three directions, similar to a histogram. This means, that if a mistake consists of two squares and the difference is [0.0039,0.0039,0], then 0.0039 will be added to R and G. There are two things to be considered when using the error for calculating the deformation: the effect of the error on the color map plane and the effect of the error on a specific direction. As described in Section 3.3, the line does not follow the plane precisely and this affects the measured error. Furthermore, our previously described binning of the errors creates a three-dimensional error vector for each line, which will not be parallel to the plane. Therefore, the error vector is projected onto the plane. The projection is orthogonal in order to take the angle between the error vector and plane into account, which alters the magnitude of the error. This projected vector can then be used in the second step. We use PCA on the projected errors of each rectangle. This will give us a major and minor axis, which is similar to fitting ellipses, which is used in evaluations of color spaces [18]. However, PCA is more stable. By taking the dot product between a direction and the PCA axes, while considering their magnitude, the effect of the error on a direction can be calculated. The center of the group is calculated as the geometric center of the lines. This approximation of the error can be used to adapt the color map for the machine the test is done on. The sorting method allows noise to creep into the results, especially with only 7 lines per rectangle, which we used in our user study. To reduce the effects of noise, smoothing is performed before the PCA calculation. The neighboring rectangles are included in the PCA calculation, but to a lesser degree. The rectangle for which the PCA is being calculated has a weight equal to the number of neighbors. The neighboring rectangles have a weight of one. This reduces the effect of the noise and allows for a smooth transition across the color map. An example of this PCA analysis can be seen in Figure 4, where the major and minor axes are displayed in the form of an ellipse for better visibility.

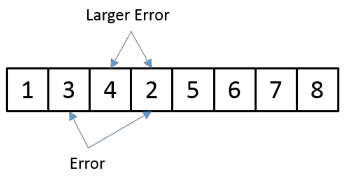


Figure 3: Illustration of occurring errors in an incorrectly sorted line. The number refers to the correct index.

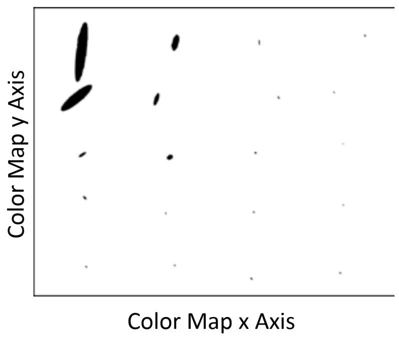


Figure 4: Image of the color map with measured errors of each line group. The ellipses represent the actual errors from a test subject after the PCA step in the different rectangles, but have been scaled to be more visible.

3.5. Morphing of Color Map

The color map is adapted to the measured errors. First the color map is divided up into a regular, rectangular grid for the purposes of deformation, in our user study 25×25 though finer grids can be used. The corners of the rectangles are defined as their position along the edges of the color map. If we take the bottom left as the origin, the right and up directions become the x and y coordinates. We use a mass-spring system in order to contract and expand areas across the color map to equalize the error, which is our goal as stated in Section 3.1. The vertices are the mass objects, and the edges are the springs. The mass-spring model can be described in the following manner. A spring h as a resting length L a constant k (here 1), and two ends. The spring applies a force equal to k times the dislocation from the resting length. The resting length corresponds inversely to the magnitude of the error. The greater the error, the shorter the resting length. We use squares instead of rectangles because we base the deformation on the size of the error and the resting length on the error relative to the mean, which allows us to compare the edges directly. This is simpler than incorporating the distance between vertices in RGB space, which in our case is taken care of when we project our deformation onto the color map because the ratio to the default state is kept. The springs are connected not only between the vertices along the x and y axes but also diagonally. A diagonal connection is required, otherwise artifacts can appear, such as neighboring springs along an axis forming a V shape. Furthermore, it allows us to cover more directions than just along the axes. Then, in order to morph the color map Algorithm 1 is applied.
The springs do not align exactly with the directions of error. For this reason, the weight of the errors depends not only on the distance from the spring, but also on the direction. The

Algorithm 1 Calculate Spring Resting Length

```
1:  $V_k \leftarrow error$ \\for the directions of all connected springs based on Equation 1 for every vertex $V_k$
2:  $V_k \leftarrow (V_k - V_{min})/(V_{max} - V_{min})$
3:  $V_k \leftarrow 1 - V_k$
4:  $V_{mean} \leftarrow Mean\ of\ all\ V_k$
5:  $S_k \leftarrow (V_{k1} + V_{k2})/2$ \\where $V_{k1}$ and $V_{k2}$ are the vertices at the ends of the spring with resting length $S_k$.
6:  if $S_k$ diagonal then
7:      $S_k = S_k \cdot \sqrt{2}$
8:  end if
9:  Set the magnitude of the distances between vertices (i.e., along the edges of the squares) to the mean value.
10: Run mass spring simulation
```

distance weight is based on a Gaussian kernel equivalent to a Gaussian with a standard deviation σ equal to the smallest distance between two line centers. A Gaussian kernel is chosen for the following reasons. First, a bilateral interpolation based on a quadrilateral mesh does not guarantee a C^1 continuity, i.e., a smooth derivative across an edge. A kernel can achieve this, which effectively smooths abrupt changes that could occur at such an edge. Furthermore, there is an issue regarding the border, as we do not have any samples on it. A vertex on the border would not be influenced by a vertex diagonally across, even though this point might have important information, especially if there are no samples at the border vertex. Therefore, constructing a quadrilateral mesh is circular: in order to create a mesh to interpolate values, we must first interpolate values. Therefore, a kernel is used instead. A Gaussian kernel is used because a linear kernel behaves poorly for points far away. A linear kernel that reaches 0 at a distance of 10 would give a point at a distance of 8 twice the weight of a point at 9. A Gaussian kernel treats points that are far away from the center similarly and emphasizes points that are near the center. The error magnitude at a spring of direction $\vec{d}$, position p, and Gaussian weights G is calculated using the σ mentioned above. The sum of the weights is set to one. Using I as the set of all line centers and J_i as the set of PCA axes for each $i \in I$ with corresponding latent magnitude $m(\vec{j})$, the formula is as follows:

$$\sum_{i \in I} \frac{G(i,p)}{\sum_{i \in I} G(i,p)} \sum_{\vec{j} \in J_i} m(\vec{j})\,(\vec{j} \cdot \vec{d}) \tag{1}$$

3.6. Boundary conditions

The vertices on the boundary are allowed to move along the boundary edges only. The corners are fixed, and no vertex may be displaced around a corner. This has an effect on the distortions inside the color map. Because the mesh cannot freely contract, certain areas will have the same size, even if the resting length is different. Assume we have two lines A and B parallel to the x axis from one border to the other, but at a different y value and different but constant error magnitudes. Because the springs are pulling equally along the line, no vertices along A or B move. Therefore, both A and B are equally long.

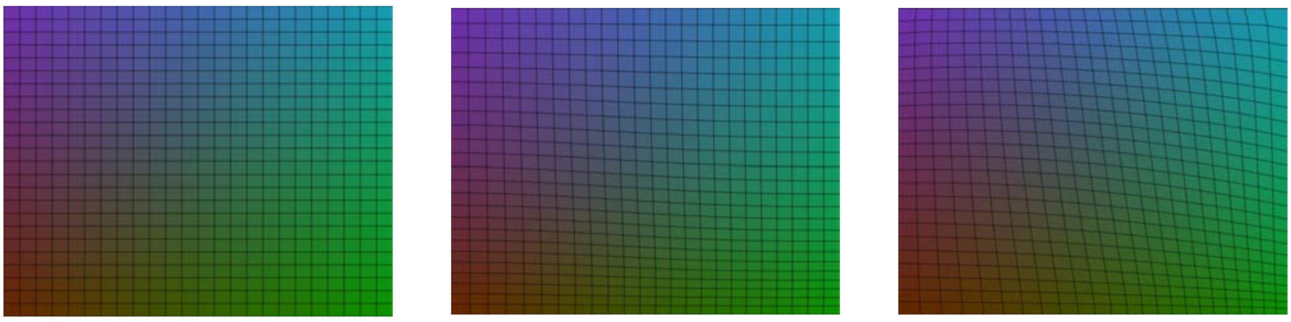

(a) (b) (c)

Figure 5: These three images show the color map in an undeformed state (a), deformed by the averaged results of users with normal color vision (b), and deformed using the distances in IPT [18] color space (c). Both (b) and (c) contract and expand in the same areas, though to different degrees.

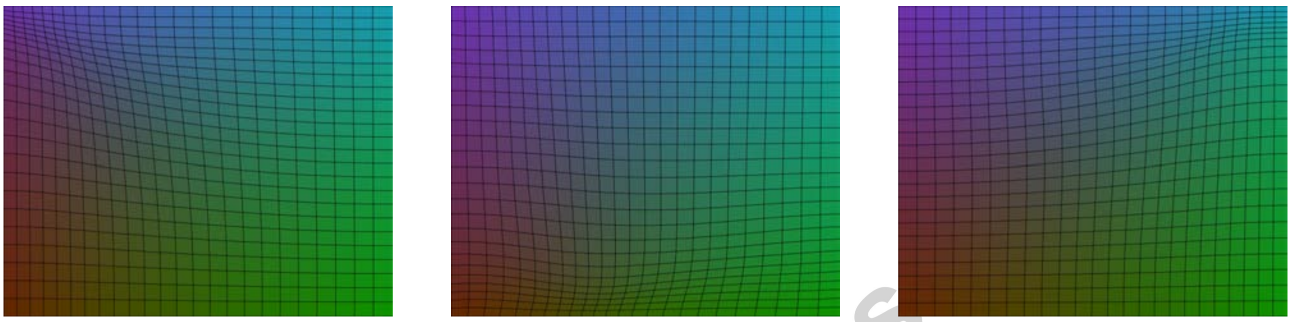

(a) (b) (c)

Figure 6: Deformations of the color map for three people with red-green weakness.

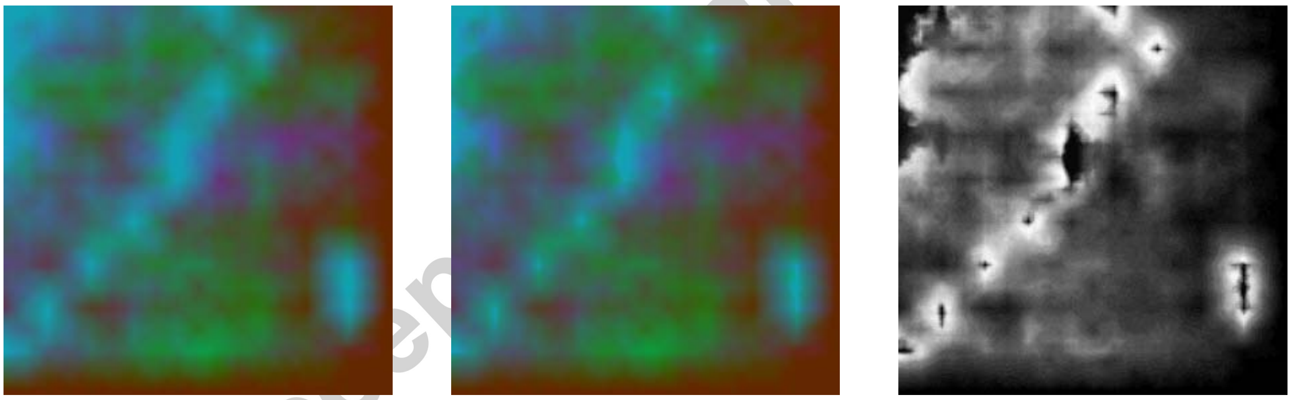

(a) (b) (c)

Figure 7: The images show a map where each pixel has 2 data values. 7a is colored by the default color map, 7b according to 6c. 7c shows the normalized difference in value. In terms of distance in RGB space, the maximum difference is 0.123 and the mean 0.04. As stated before, axes in RGB space have a maximum value of 1.

4. User Study

4.1. Users

Nine subjects with normal color vision and three subjects with a red-green color weakness participated; the particular type of weakness was not checked. Subjects were screened using Ishihara's tests for color blindness.

4.2. Calibration and Setup

The user study took place in a darkened room, i.e., no lamps and with covered windows. The monitor was a standard white LED monitor that was calibrated to D65 color temperature and sRGB. No restrictions were set on the user regarding the distance to the monitor. This was allowed in order to provide similarity to a standard work space. The user was presented with a screen that contained the following: 8 squares in a sequence in the center of the screen, a submit button at the bottom, and a menu button in the top left. The background was almost black. Users complained that a completely black background caused strain on their eyes, so it was changed to RGB = [0.071, 0.071, 0.071]. Users had to solve a total of 140 sequences. When the submit button is pressed after each solution, the user was asked to rate the difficulty of that line from 1 to 10, 10 being the hardest. The sectors were traversed in the same predetermined, ordered manner for all participants. There was no indication that participants became exhausted during the experiment. The users were informed that they could take as long as they needed and could take breaks if they wanted to, but were made aware that there were a total of 140 lines to solve.

4.3. Color Map

The color map used in the user study has the endpoints [0.392, 0.157, 0] (reddish brown), [0.039, 0.588, 0] (green), [0.431, 0.196, 0.706] (purple), and [0.078, 0.235, 0.706] (cyan). An example can be seen in Figure 5a. In all images in this paper containing the color map, these correspond to the bottom left, bottom right, top left, and top right corners respectively, and will be referred as such. These points were chosen to show that our method would work with varying luminance and color, and would not require a specially chosen color map. While a color map that only has variances in luminance or in hue can be chosen, the latter can cause difficulties, as it is impossible for the luminance to be equal for all points on the color map. If a sequence is extracted from such a color map, then the luminance along it varies. Essentially, it is aliasing. The computer would not be able to display each color at the same luminance due to the discrete nature of its color space. Therefore, the sequence would not appear ordered along a straight line to the user and can lead to confusion, as we noticed in a pilot study.

4.4. Color Map Division

As described more generally in 3.3, the color map was divided up into a grid of 20 rectangles, arrayed 5×4. The longer axis had 5 along it. The user had to solve a total of 140 lines extracted from the color map; 7 from each rectangle at random. The lines were generated randomly across the color map. These values were chosen so the color map would be well covered.

5. Results

The subjects with normal color vision took 56 minutes on average to complete the study. The minimum time was 28 and the maximum 90 minutes with a standard deviation of 21.7 minutes. The subjects with red-green weakness took 55, 51, and 39 minutes respectively. See Table 1 for more information. There was no correlation between time taken and number of errors. All subjects with normal color vision had a similar result. The averaged deformation of people with normal color vision and a deformation based on the perceptually uniform IPT color space [19] as a comparison can be seen in Figure 5. The distances in IPT space along the x and y axes can be seen in Figure 8a and 8b respectively. The deformation of both the color map based on the IPT color space and that based on the averaged users with normal color vision contract in the bottom right (green) area. This means our method expands and shrinks in the correct areas. However, the IPT version contracts less vertically

Table 1: Time to Solve (TtS) for different subjects

Weakness	Median TtS (seconds)	Total TtS (minutes)
None	13.4	35
None	24.3	75
None	12.1	34
None	20.3	57
None	32	90
None	15.1	38
None	27.1	68
None	11.7	28
None	30	83
Red-Green	18.8	55
Red-Green	20.6	51
Red-Green	14.9	39

but more horizontally. Unlike the deformation for users, the deformation based on the IPT color space alters the constants of the springs, rather than the resting lengths, which are set to 0. Because the distances in the color space are known, we can create a more exact deformation to try to preserve the perceptual distance. The constants are set to $1/distance_{IPT}$. This can be explained through a simple one-dimensional example. There are two springs. Along the first spring the perceptual distance is 1, along the second 2. If we set the constants to $1/distance_{IPT}$, then we have the constants $k_1 = 1$ and $k_2 = 0.5$. This means the spring with a perceptual distance of two will be twice as long. The three subjects with red-green color weakness all had substantially different results. While the exact type of color deficiency of the users was not determined, this indicates that personalized color maps for people with a color weakness can be fairly unique, and so personalized color maps may be needed. Their distortions can be seen in Figure 6. Figure 8c shows the difficulties reported by the users with normal color vision. The users considered the bottom right to be the most difficult, the same area where the most contraction occurs. In comparison, the distances in IPT space can be seen in Figures 8a and 8b.

6. Validation Study

A validation study under the same conditions was performed on the users with color deficiency. The users were shown an image of a curvy line on a monotone background. The colors were selected from default color map and the personalized color map.

6.1. Selected Color

The color of the line and background were chosen in the following manner. A point on the color map where the most contraction occurred was selected for each individual user as the background color. The color of the line was selected by taking a nearby point, i.e., the new point was offset from the original. The offsets were 0, 2, and 4 steps on the color map

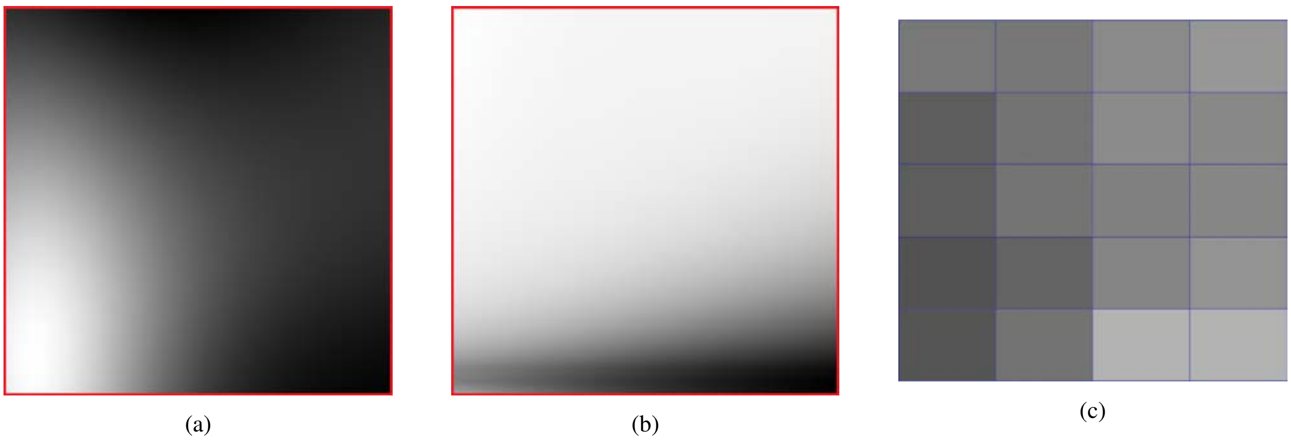
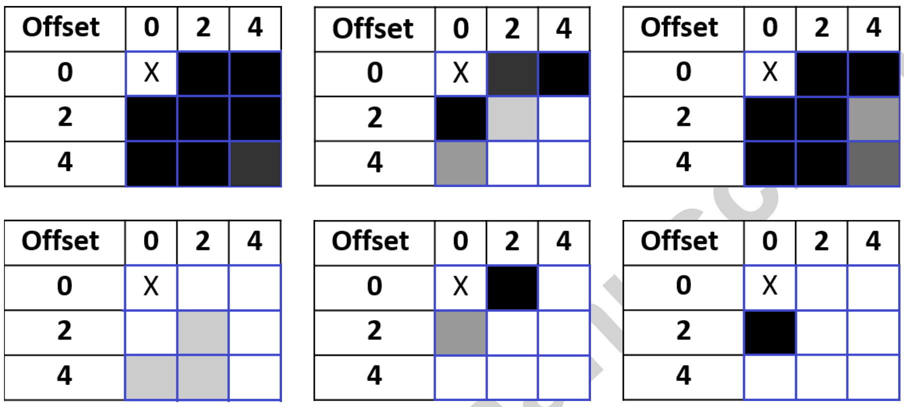


Figure 8: Normalized perceptual distances of color map in IPT color space along x axis (a) and y axis (b). Darker indicates shorter distance. (c) Averaged normalized difficulties reported by users. Areas of higher difficulty are lighter.



Offset	0	2	4
0	X	■	■
2	■	■	■
4	■	■	▓

Offset	0	2	4
0	X	▓	■
2	■	░	□
4	▒	□	□

Offset	0	2	4
0	X	■	■
2	■	■	▒
4	■	■	▓

Offset	0	2	4
0	X	□	□
2	□	░	□
4	░	░	□

Offset	0	2	4
0	X	■	□
2	▒	□	□
4	□	□	□

Offset	0	2	4
0	X	□	□
2	■	□	□
4	□	□	□

Figure 9: These tables show the capability of the individuals to detect curvy lines with offset color map values in an image. The top row (a-c) are with unaltered color maps. The bottom row (d-f) with personalized color maps. The offsets were 2 or 4 along the x (towards the right) and y (downwards) axes. The lightness indicates the number of detections, with a total of 5 lines per cell. The darker the cell, the less lines were detected; black none, white all.

along the x and y axes and their combination (i.e., one diagonal direction). If viewed on the color map, the x direction is to the right, the y direction is downwards, and the diagonal direction is towards the bottom right.

6.2. Line Variations

The line either went from the left to the right border of the image, or from the top to the bottom. The end points were not aligned and could be on independent, arbitrary points along the border. In order to avoid exhaustion, the number of line and color variations was minimized. Five lines were generated, 3 horizontal and 2 vertical. While there were a limited number of variants, the line had a random horizontal or vertical shift, depending on the endpoints. If the endpoints were on the right and left borders, the line was shifted randomly up or down, otherwise the line was shifted arbitrarily to the right or left. However, the shift was the same for each offset combination. For a given offset and line, the only difference would be the colors used.

6.3. Execution and Results

These variations result in 80 images. The image was displayed for 3 seconds, after which the user had to say the direction of the line, or if he/she did not see a line. The number of cases where the user could not detect the line or mistook its direction were counted. Two users mistook the direction of the line only once, and one user three times. Because these numbers are so low, these cases were counted as not seeing the line. The results can be seen in Figure 9. As expected, the users all perform better with the adapted color map, though to varying degrees. If we take the colors from the adapted color map, then their distance on the original color map is 2 to 3 times higher, depending on the user.

7. Conclusion

The method of generating perceptually uniform color maps we have described can show where a person has difficulty differentiating color, and it does so in a reasonable amount of time. It also has the benefit of being more enjoyable. The variation among persons with red-green color weakness indicates a need for personalized color spaces in such cases. Another application could be a grading of color maps based on the magnitude of errors. Furthermore, while it has not been tested, our method should be usable on a non-calibrated monitor under arbitrary lighting, similar to previous work [16].

7.1. Comparison to Previous Work

As mentioned before, a similar method was created by Gresh [16], but can be expected to require 75 minutes to complete. Our method can be completed in 30 to 40 minutes, though some users take significantly longer. Furthermore, it is influenced by the game Blendoku [3], which has a huge player base, indicating that it is enjoyable. It is also not as exhausting. In the paper by Gresh [16], the user must constantly react within seconds. In our game, he progresses as fast as he/she wants.

7.2. Limitations

The method still has several limitations. The color map must be fairly fine-grained, otherwise the steps between the individual squares become too large and the errors will be simply noise. Also, it is still a lengthy process. The fastest users took about 30 minutes. Furthermore, the samples are semi-random. As can be seen in the case for the users with color weakness, most of the contraction is in one area, wasting a lot of time. Furthermore, the method can not detect if a user can not distinguish two colors that are not next to each other, such as red and green.

7.3. Future Work

Some of the limitations can be overcome. For example, adaptive sampling could reduce time and increase accuracy by increasing and decreasing sampling in areas with many or few errors. This would also help detect smaller areas where large contractions or expansions are required. Further research could reduce the number of squares in a line. The number affects the users ability to sort them — more squares allow for more comparisons — as well as the degree of difficulty that can be measured. There are also some aspects that have not been researched. The study was done in a dark room. This makes it easier for the user to distinguish colors. It is unknown how well the method will perform in a normal working environment, as daylight can affect color perception [20]. Another question is consistency between people with color deficiencies. In our study, the users with a red-green color deficiency all had different results. It is unknown if this is due to them having different deficiencies, or such differences occur for the same deficiency. Lastly, it might be possible to tie in the monitor calibration methods shown in [13] and [14]. First, similar to the method in [13], confusion axes – axes along which the user has difficulty seeing colors – could be used to speed up measurement and accuracy. Second, these methods try to measure a persons color vision across the monitor in order to use recoloring methods. They try to measure the distance at which a person can differentiate colors in certain areas of the color space. This measurement can be used, similar to the distortion of the color map using IPT space in Section 5 using the spring constants, to adapt any color map. However, as shown by the variation in areas where the color map contracted for different users, the accuracy and individualism is unknown.

Acknowledgements

This project has been funded by the Vienna Science and Technology Fund (WWTF) through project VRG11-010 and also supported by EC Marie Curie Career Integration Grant through project PCIG13-GA-2013-618680.

[1] S. Mittelstädt, J. Bernard, T. Schreck, M. Steiger, J. Kohlhammer, D. A. Keim, Revisiting Perceptually Optimized Color Mapping for High-Dimensional Data Analysis, in: EuroVis - Short Papers, 2014, pp. 91–95.

[2] H. Hofer, J. Carroll, J. Neitz, M. Neitz, D. R. Williams, Organization of the human trichromatic cone mosaic, The Journal of Neuroscience : The Official Journal of the Society for Neuroscience 25 (42) (2005) 9669–9679.

[3] LonelyFew, Blendoku (2015).
URL `http://www.blendoku.com/`

[4] J. Bernard, M. Steiger, S. Mittelstädt, S. Thum, D. Keim, J. Kohlhammer, A survey and task-based quality assessment of static 2d colormaps, in: Proceedings of SPIE, 2015, pp. 93970M–93970M–16.

[5] M. Steiger, J. Bernard, S. Mittelstädt, M. Hutter, D. Keim, S. Thum, J. Kohlhammer, Explorative analysis of 2d color maps, in: Proceedings of WSCG, 2015, pp. 151–160.

[6] B. Sajadi, A. Majumder, M. Oliveira, R. Schneider, R. Raskar, Using patterns to encode color information for dichromats, Visualization and Computer Graphics, IEEE Transactions on 19 (1) (2013) 118–129.

[7] D. R. Flatla, K. Reinecke, C. Gutwin, K. Z. Gajos, Sprweb: Preserving subjective responses to website colour schemes through automatic recolouring, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2013, pp. 2069–2078.

[8] G. Kuhn, M. Oliveira, L. A. Fernandes, An efficient naturalness-preserving image-recoloring method for dichromats, Visualization and Computer Graphics, IEEE Transactions on 14 (6) (2008) 1747–1754.

[9] W. Chen, W. Chen, H. Bao, An efficient direct volume rendering approach for dichromats, Visualization and Computer Graphics, IEEE Transactions on 17 (12) (2011) 2144–2152.

[10] K. Rasche, R. Geist, J. Westall, Re-coloring images for gamuts of lower dimension, Computer Graphics Forum 24 (3) (2005) 423–432.

[11] G. M. Machado, M. M. Oliveira, Real-time temporal-coherent color contrast enhancement for dichromats, in: Proceedings of the 12th Eurographics / IEEE - VGTC Conference on Visualization, 2010, pp. 933–942.

[12] M. Oliveira, Towards more accessible visualizations for color-vision-deficient individuals, Computing in Science Engineering 15 (5) (2013) 80–87.

[13] D. R. Flatla, C. Gutwin, Improving calibration time and accuracy for situation-specific models of color differentiation, in: The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility, 2011, pp. 195–202.

[14] D. R. Flatla, C. Gutwin, Situation-specific models of color differentiation, ACM Trans. Access. Comput. 4 (3) (2012) 13:1–13:44.

[15] G. Kindlmann, E. Reinhard, S. Creem, Face-based luminance matching for perceptual colormap generation, in: Proceedings of the Conference on Visualization '02, 2002, pp. 299–306.

[16] D. L. Gresh, Self-corrected perceptual colormaps, [Online] (2010).
URL `http://www.research.ibm.com/people/g/donnagresh/colormaps.pdf`

[17] S. McKee, S. Klein, D. Teller, Statistical properties of forced-choice psychometric functions: Implications of probit analysis, Perception & Psychophysics 37 (4) (1985) 286–298.

[18] M. Mahy, L. Van Eycken, A. Oosterlinck, Evaluation of uniform color spaces developed after the adoption of cielab and cieluv, Color Research and Application 19 (2) (1994) 105–121.

[19] F. Ebner, M. D. Fairchild, Development and testing of a color space (ipt) with improved hue uniformity, Color and Imaging Conference 1998 (1) (2005) 8–13.

[20] D. L. MacAdam, Visual sensitivities to color differences in daylight∗, J. Opt. Soc. Am. 32 (5) (1942) 247–274.