Haifa University

Computer Science Department

Imaging Science & Technology Course, 2005

Hue Cancellation

Authors:

Shai Erera (serera@gmail.com)

Hagay Pollak (hagaypollak@gmail.com)

Hue Cancellation

Project Description

The purpose of the project is to demonstrate the Opponent Colors theory via a Java applet. The applet is kind of a test in which the user is given (or selects) a starting color, an axis (Red-Green / Yellow-Blue) and *Plus* and *Minus* buttons to move along the axis. The goal of the test is to reach an achromatic color. When the user believes he has reached an achromatic color (i.e. a balanced color – with no influence of the 2 colors selected on the scale, r-g or y-b), he can click on *Accept* which will display the RGB values of the chosen color. The user or test analyzer can then compare the results to the CIE diagram and view which colors the user believed to be of no hue to either of the 2 colors from the axis selected (red-green or blue-yellow). Should the test be successful, the analyzer of the results should see selections surrounding the 2 main lines connecting the unique colors (as these are the points in which the hue is actually canceled).

This project is done as part of a course under the guidance of Dr. Hagit Hel-Or, Computer Science department, Haifa University.

Hue Cancellation Introduction

Hurvich and Jameson reasoned that when red and green are mixed together, they produce yellow, not reddish green. Further when yellow and blue are mixed together they produce white and not yellowish blue. Red and green cancel each other as do yellow and blue. They further reasoned that if one started with a color such as bluish green it should be possible to mix this color with a unique yellow to cancel out the blue content leaving only green. This is the basic idea behind the hue cancellation technique.

Applet Description

Main Tab

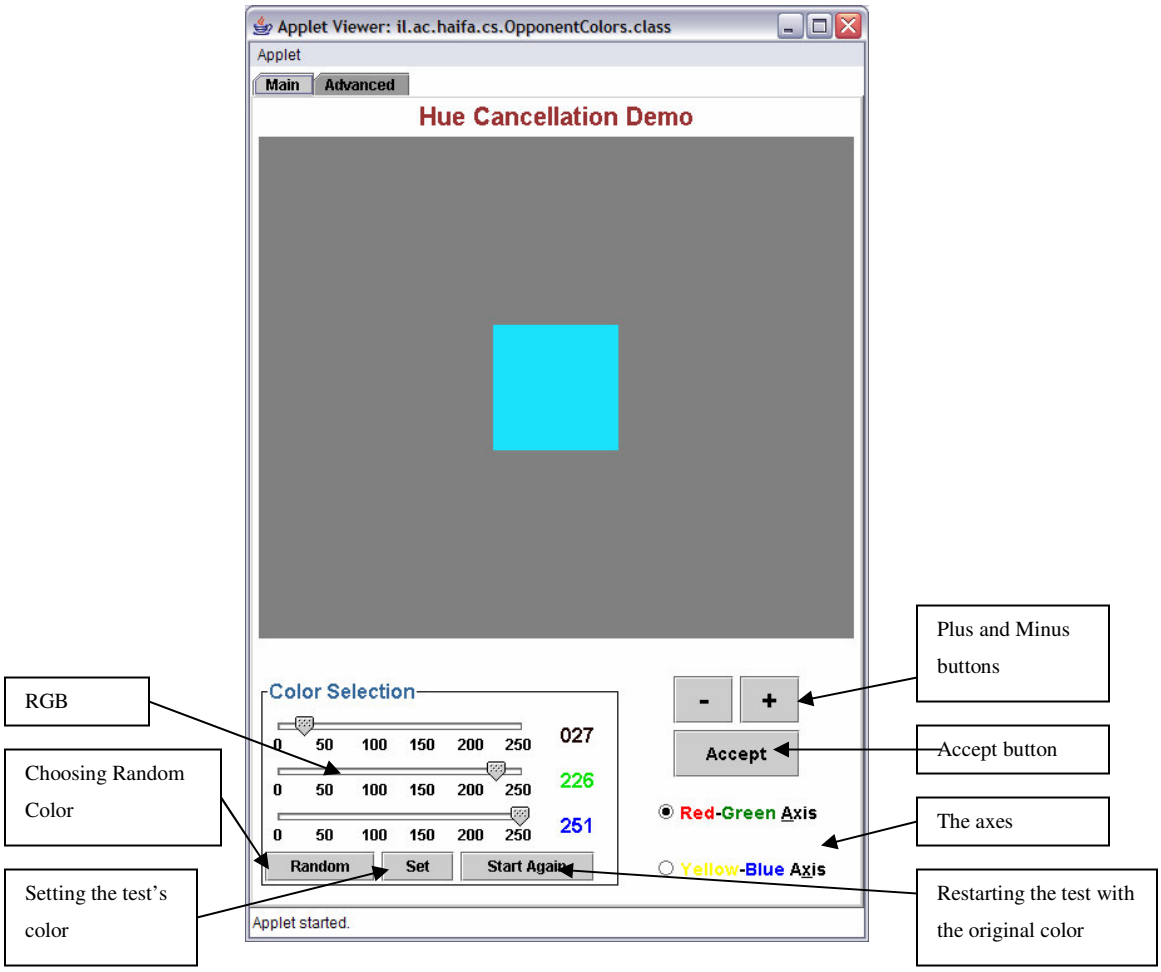

Figure 1: Applet's Main Tab

The *Main* tab contains the experiment section. It has two radio buttons that control the axis the user is moving along when clicking the *Plus* and *Minus* buttons. It also has an *Accept* button which the user can click and view the RGB values of the chosen color.

The *Color Selection* area contains 3 sliders which can be used to determine the RGB value of the starting color. The labels near the sliders are painted according to the value of the slider. The *Random* button draws a random color, the *Set* button sets the starting color and the *Start Again* button allows the user to re-start the experiment.

Advanced Tab

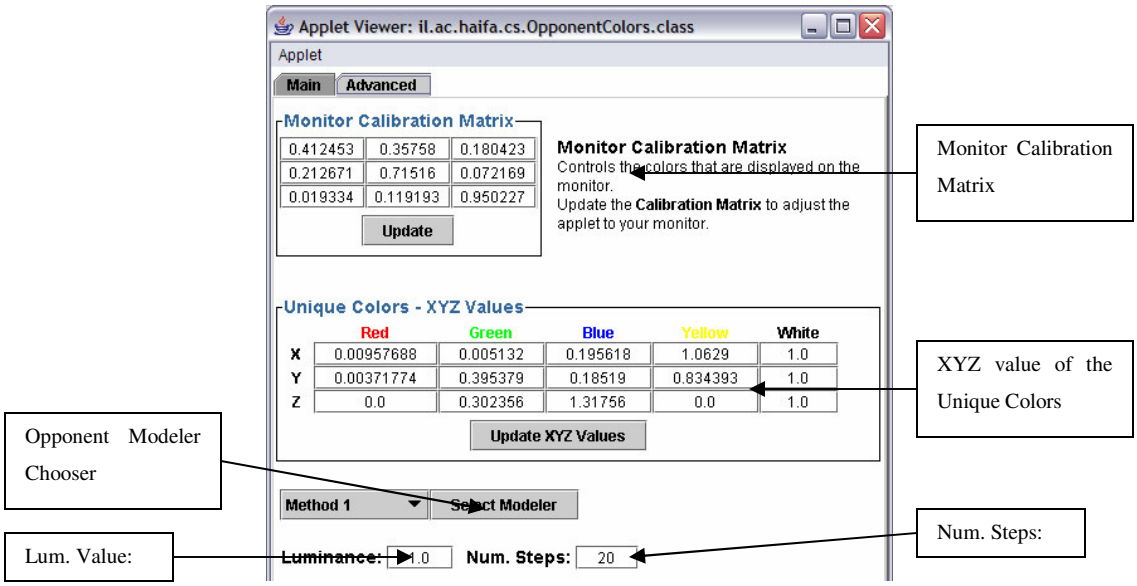

Figure 2: The Advanced Tab

The *Advanced* tab contains advanced settings of the experiment. The *Monitor Calibration Matrix* is used by the applet to control the actual colors that are displayed on the screen. One can use a photometer to calibrate the applet to match his monitor.

The *Opponent Colors* model can be based on using 2 pairs of opponent colors (also known as unique colors): *Red*, *Green*, *Yellow* and *Blue*. The tab displays the XYZ values of these 4 unique colors, as well as the *Reference White* point. By updating this matrix, the user controls the 4 unique colors and can change the output of the experiment.

The applet contains 3 implementations that model the *Opponent Colors* (each is described in detail later). The user can choose the modeler the applet works with.

Methods Used to Model the Opponent Colors

We offer 3 methods to model the Opponent Colors. The first two methods translate the CIE diagram to an (rg, yb) coordinate system. Our methods are based on *Opponent Colors* modeling which is presented in 1.

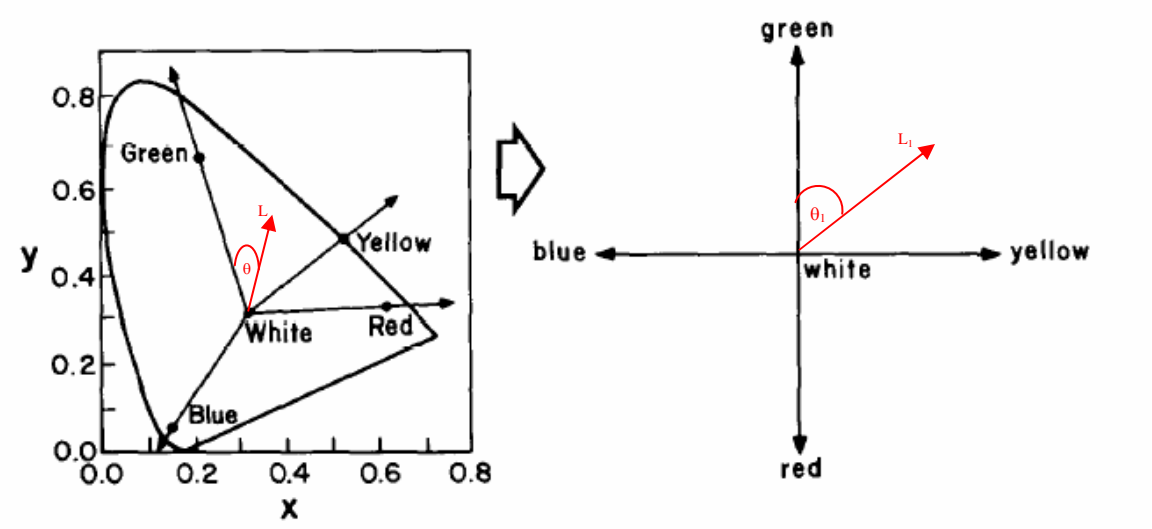

Figure 3: Transformation of the Opponent Colors axes

Method 1

This method uses 471 Color Matching Functions (CMF) samples to determine the "horse shoe" boundaries. Each quarter in the rg / yb coordinate system matches a section in the "horse shoe". For example, the section that is bounded by the Yellow and Green chromatic axes matches the first quarter in the rg / yb coordinate system (the quarter which is between the "yellow" and "green" labels). The boundary of that section is mapped to the circle boundary of the first quarter. If a color is found in that section, the following calculations are performed:

- Its relative angle between the Yellow and Green vectors. This angle is then used to determine its angle in the rg / yb coordinate system.
- The relative angle also bounds the color between two of the CMF.
- Then its distance from the *Reference White* is calculated. Also the distance of the two CMF is computed and is considered '1' in the rg / yb plane.
- The position in the rg / yb coordinate system is determined by calculating the relative length from the two CMF and using the angle that is calculated before.

When the user clicks the *Plus* or *Minus* buttons he moves along the chosen axis in the rg / yb plane. These coordinates are then transformed to the appropriate (x, y) coordinates in the CIE diagram and from that they are transformed to XYZ and RGB values. The color is then displayed on the screen.

Method 2:

This method is very similar to *Method 1* only it doesn't use CMF for the calculations. After calculating the relative angle, the length of the point is computed relative to the vector that connects the two unique colors (for example Yellow-Green for the first quarter). All the points on that vector and above it are considered as length '1' in the rg / yb plane.

The inverse calculation is done in the same way as in *Method 1*.

Method 3:

This method models the *Opponent Colors* in the following way: for each color it computes the (x, y) values in the CIE diagram. Also, *delta-x* and *delta-y* are calculated for each of the unique colors from the *Reference White* (the actual delta is calculated as the difference of Xs or Ys divided by the number of steps [20 in the current version] it should take to reach from the unique color to the *Reference White*). When the user clicks the *Plus* or *Minus* buttons, the modeler updates the x, y values according to their current position. For example, if the color is left to the *Reference White* and the user is moving along the Red-Green axis, then the x, y values will be *x += deltax_green* and *y += deltay_green*. When the color is right than the *Reference White*, the calculations will be the same, only with the delta-x and delta-y of the Red color.

These 3 modelers give different results and the user can use them to compare.

Experiment Results

We tried to cancel the Red/Green and Yellow/Blue hues using the applet. We set the luminance to 1.0 and the number of steps to 30. We had 10 runs and for each we recorded the rg and yb values. In each run we started with a random color and moved along the Red-Green axis to achieve the Hue Cancellation. Plotting them clearly shows an almost "flat" line around the 0, which proves that the Yellow-Blue axis contains colors which have no Red or Green hues.

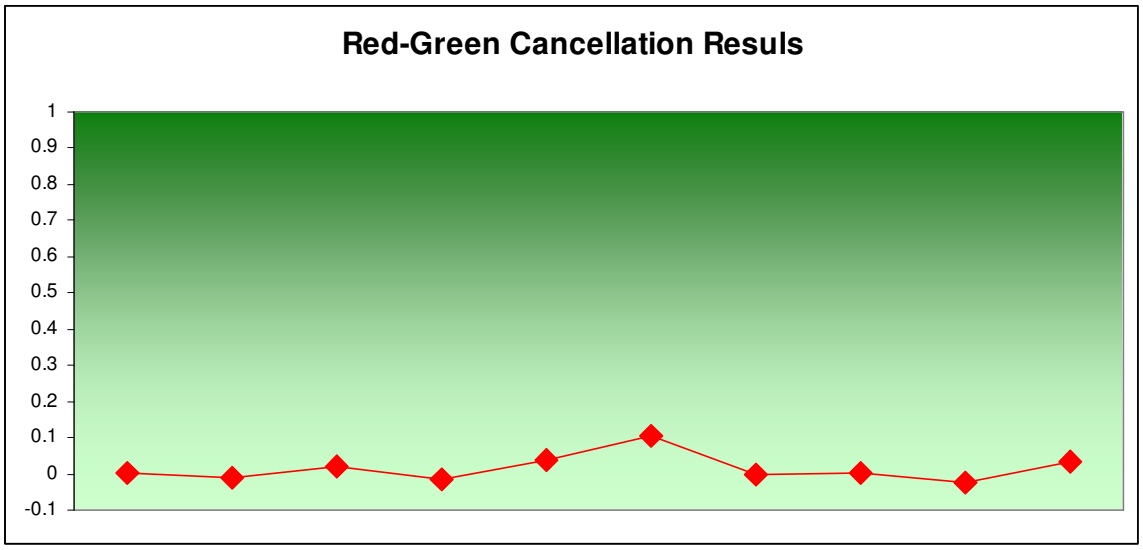

Figure 4: Red-Green Experiment Results

	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
RG	0.00138	-0.0102	0.01995	-0.01577	0.03729	0.10622	-0.00217	0.003	-0.02329	0.03142
YB	-0.02557	-0.19889	0.15165	-0.19864	0.40655	0.31912	0.03468	0.02721	-0.35012	-0.0774
R	**255**	**255**	**255**	**255**	**255**	**255**	**255**	**255**	**235**	**255**
G	**242**	**240**	**233**	**237**	**216**	**242**	**238**	**240**	**235**	**251**
B	**245**	**255**	**177**	**255**	**101**	**113**	**219**	**221**	**255**	**255**

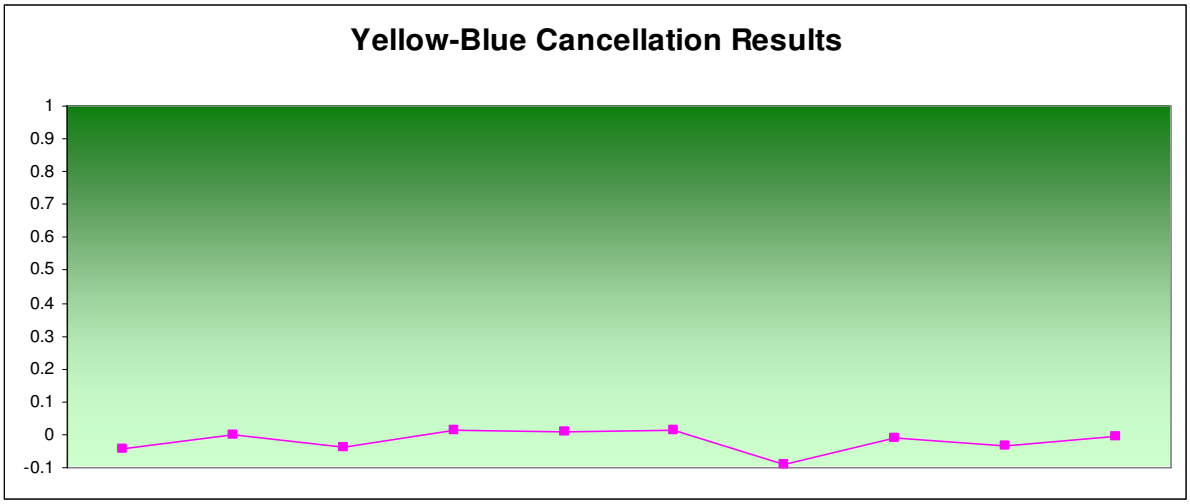

Figure 5: Yellow-Blue Experiment Results

	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
RG	-0.00352	-0.56369	-0.12926	0.12211	-0.85608	0.37246	-0.12392	0.58236	0.04298	-0.14494
YB	-0.04203	-4.70E-04	-0.03873	0.01316	0.00861	0.01154	-0.09048	-0.01175	-0.03246	-0.00751
R	**255**	**255**	**255**	**144**	**255**	**0**	**255**	**0**	**255**	**255**
G	**240**	**33**	**204**	**255**	**0**	**255**	**205**	**255**	**254**	**194**
B	**255**	**103**	**239**	**202**	**25**	**183**	**255**	**182**	**242**	**210**

Code Description

The code is written in Java and contains the following classes:

- *OpponentColors* – this is the main class which creates the applet and places all the controls on it. It handles all of the controls actions.
- *OpponentModeler* – this is the interface for all the opponent modelers in the system. If one wishes to develop a new model, it should implement this interface.
- *OpponentModeler1, 2, 3* – these are the 3 opponent models we developed in the application. Additional information on each can be found under the *Method 1, 2, 3* described above.
- *XYZConverter* – this class is used by each model to convert from RGB to XYZ and xyY and vice-versa.
- *CMFValues* – this class contains the 471 Color Matching Functions. Each entry written in the array contains the: *nm*, *X* value, *Y* value and *Z* value. To use different CMFs, the *cmfValues* matrix needs to be changed.
- *NMWrapper* – this class wraps a nm and all its XYZ values. It is used by the XYZConverter class.
- *HueCancellation.html* – this HTML displays the applet. In order for it to work, the code must be enclosed in a JAR file and placed in the same directory as the HTML file.

References

1. Michael W. S. "*An Experimental Comparison of RGB, YIQ, LAB, HSV, and Opponent Color Models"*. University of Waterloo.