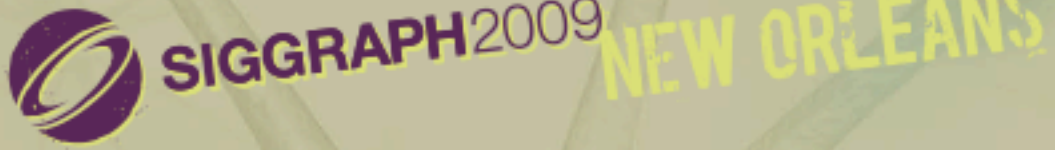


Color Imaging

Erik Reinhard
Greg Ward
Garrett Johnson

Color in Nature

Color in Art

Photograph Courtesy of Kirt Witte

Color in Science and Engineering

- Image Processing
- Computer Graphics
- Computer Vision

Schedule

- Before the break: Fundamentals
  - Introduction to Color Imaging (Reinhard)
  - Radiometry/Photometry/Colorimetry (Johnson)
  - Color Spaces (Reinhard)
  - Color Appearance (Johnson)

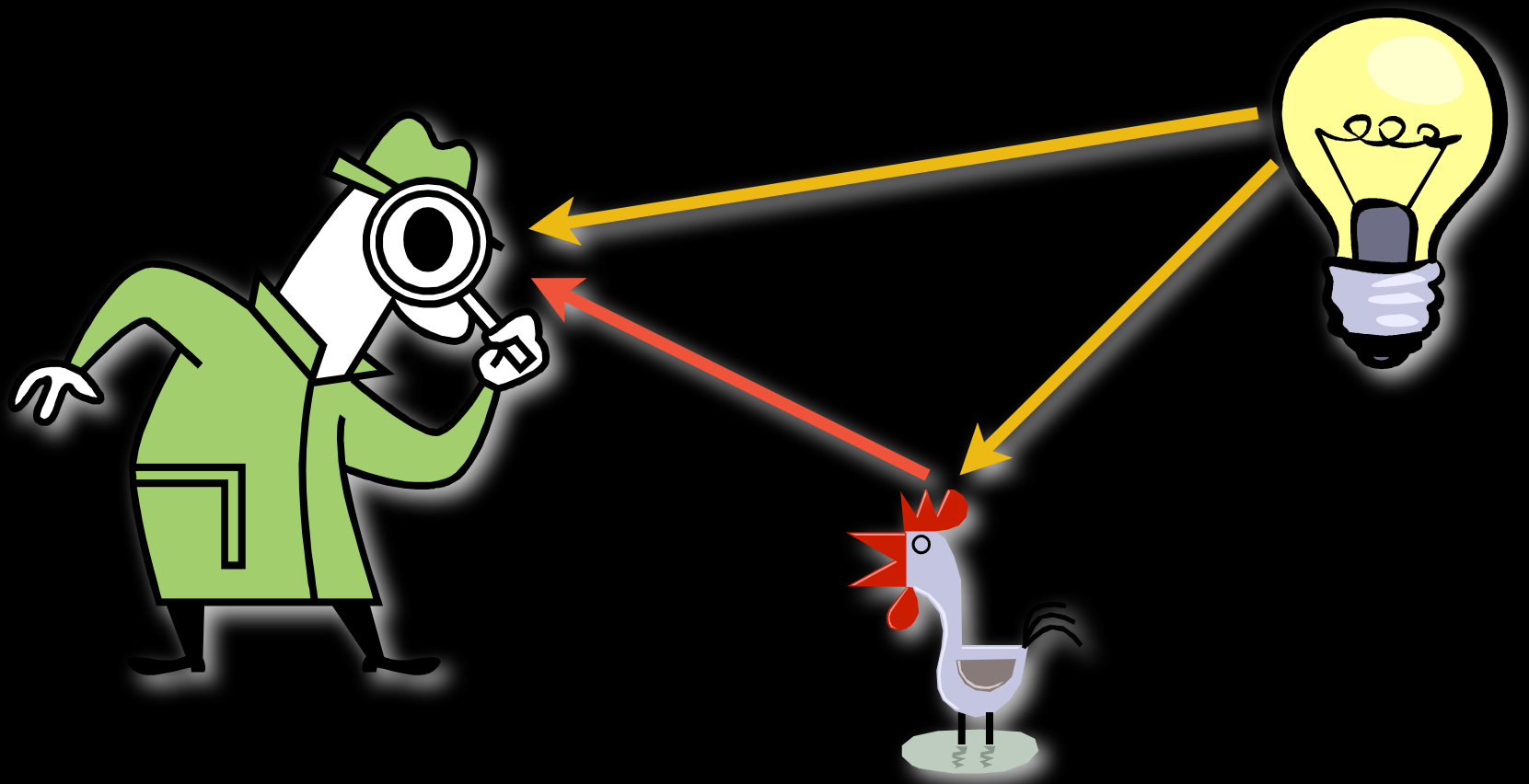
Schedule

- After the break: Applications
  - Application: Rendering (Ward)
    - Spectral prefiltering and sharp color primaries
  - Application: High Dynamic Range Imaging (Ward)
    - Image encoding
  - Application: Image Processing (Reinhard)
    - Color transfer between images
    - A simplified color appearance model
  - Discussion (All)

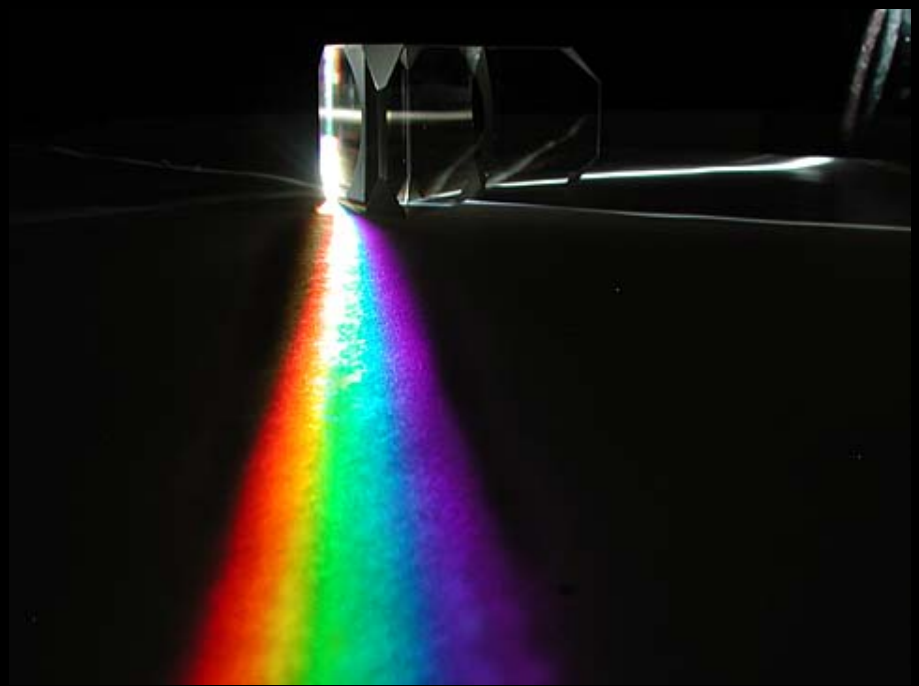
The Metrys: Radiometry, Photometry, and Colorimetry

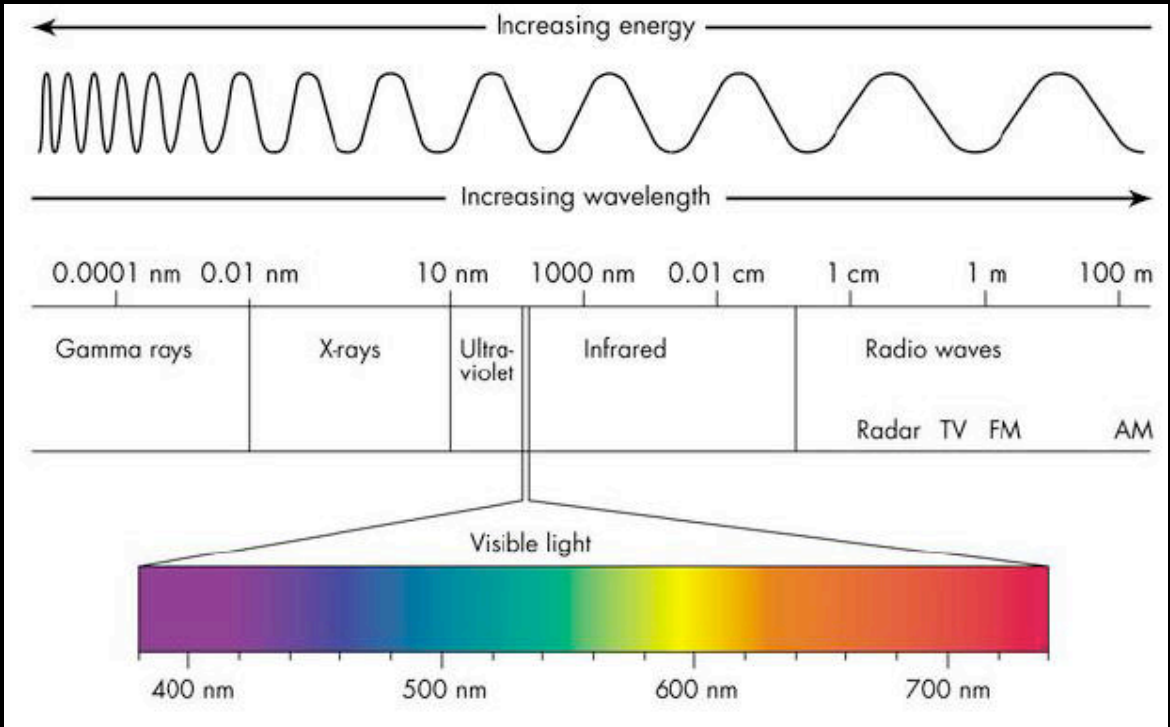
Garrett M. Johnson

Color is a visual perception...that requires three[ish] things!



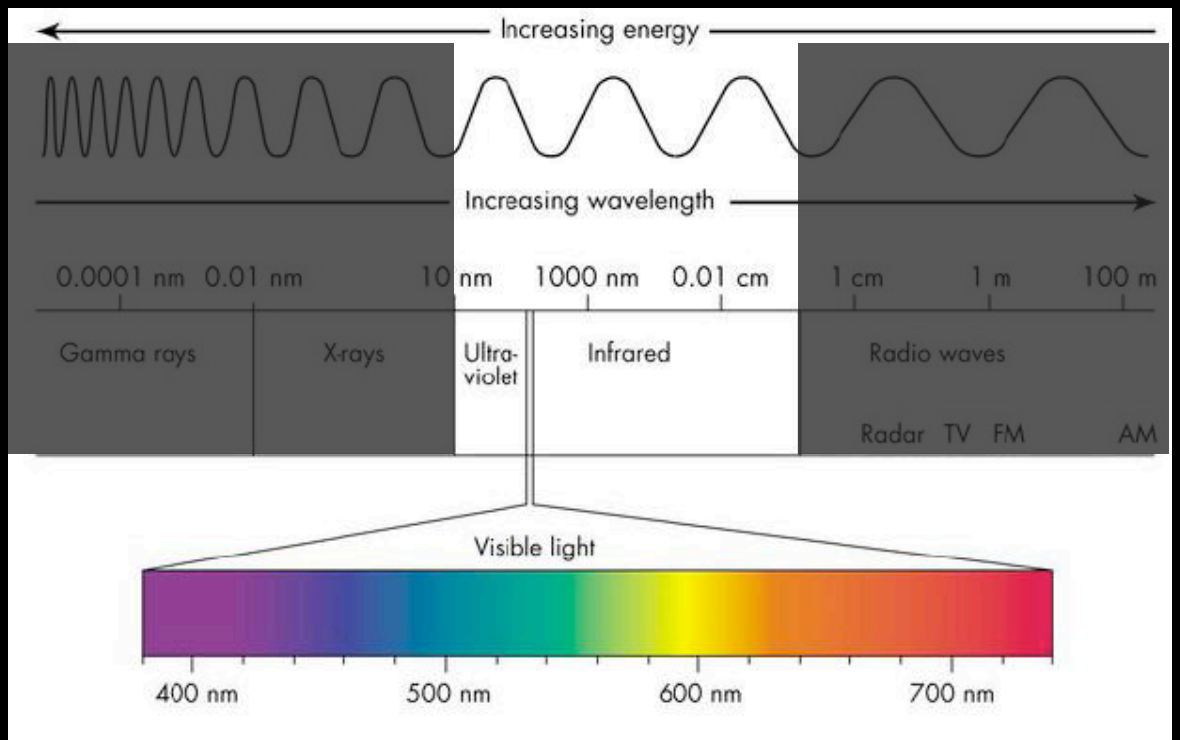
Ultimately Color Requires Light





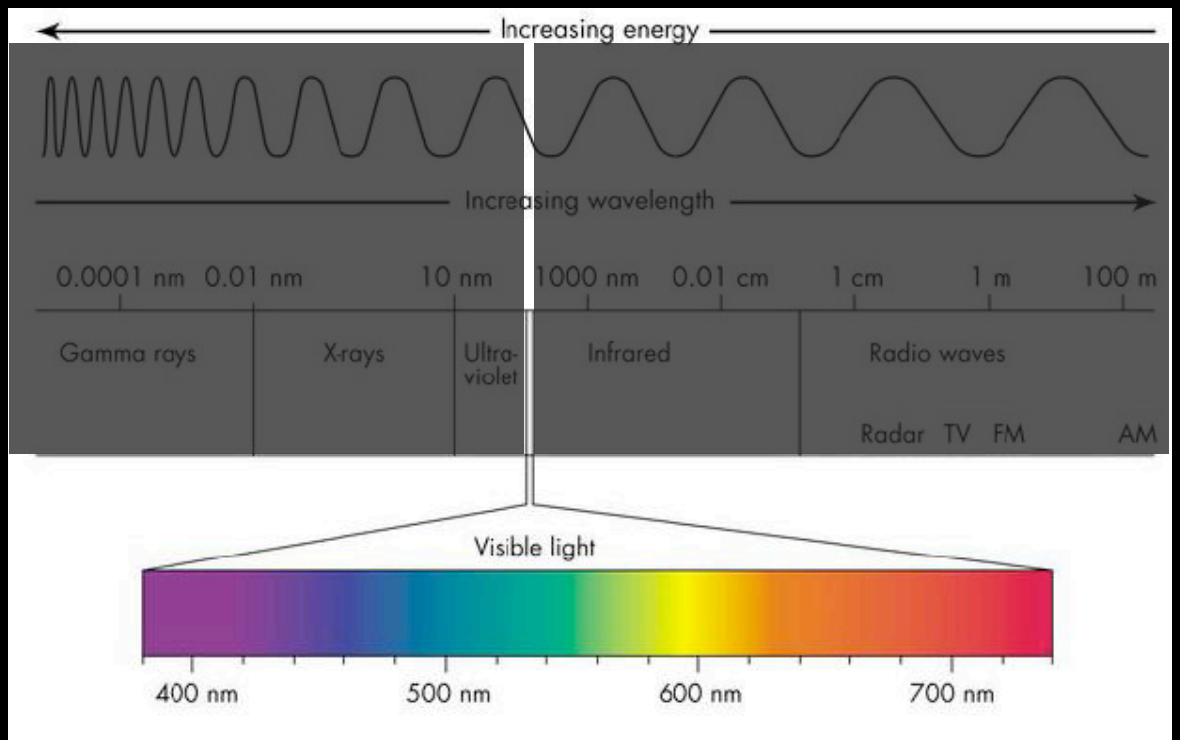
380-720 rendered in sRGB

Radiometry



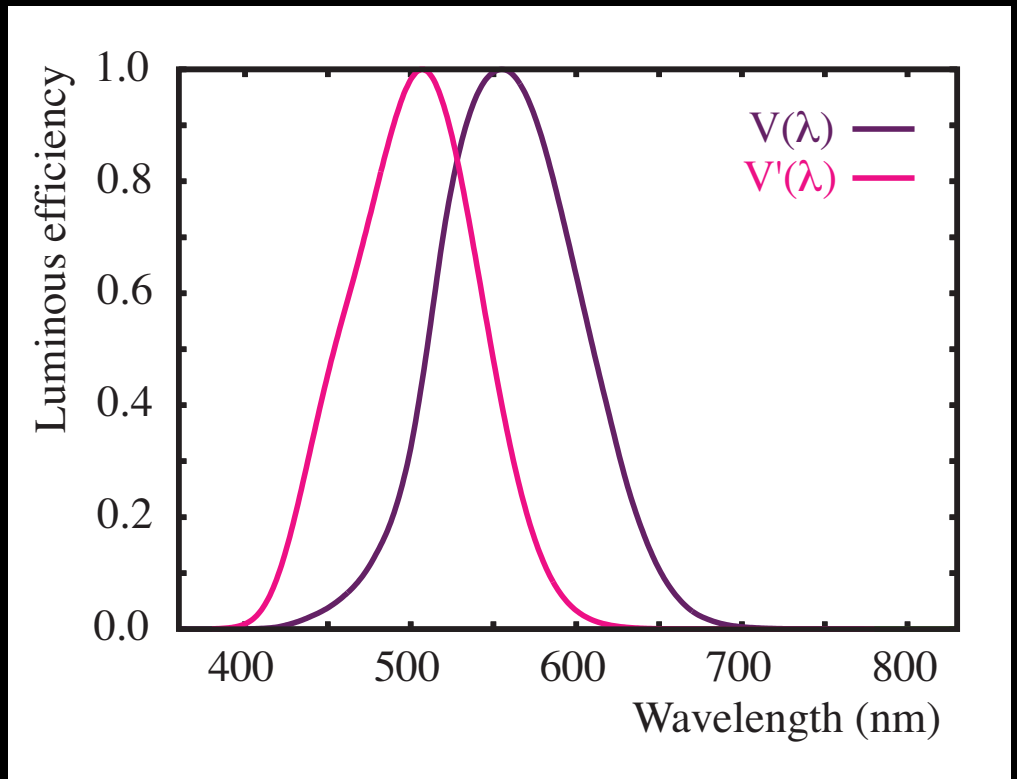
Radiometry is simply the measurement or quantification of optical radiation, and can include radiation outside the visual range (e.g. UV or infrared).

Photometry



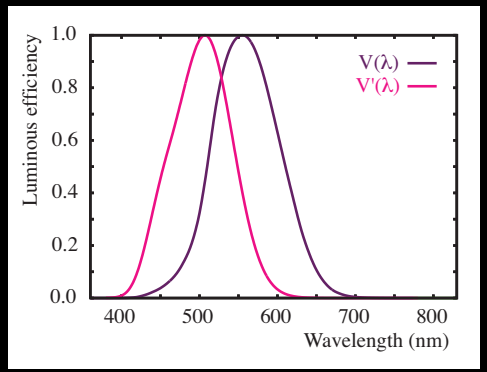
Photometry is the measurement of light, as weighted by visual perception. Think of it as a subset of radiometry.

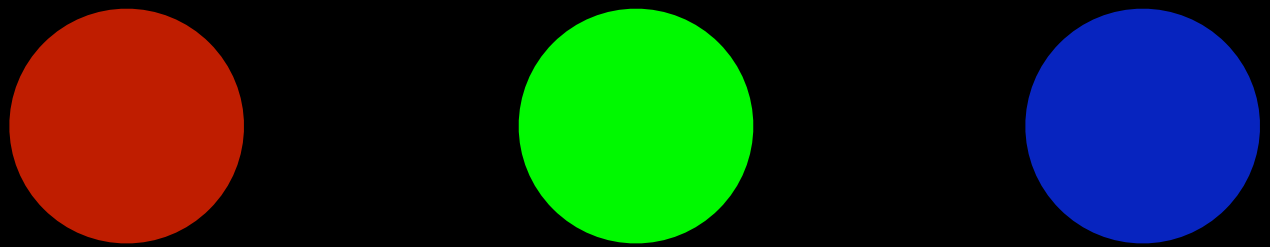
Sensitivity of the Eye



V(λ) and V'(λ) are the photopic and scotopic luminous efficiency curves, e.g. how well we see wavelengths in normal and darkened conditions, as governed by our cones and rods.

Sensitivity of the Eye





Photometry is based upon the V(λ) curve. Green wavelengths of light appear to be lighter than similar amounts (photons) of blue or red. The radiometric measurements are weighted or integrated by the V(λ) curve.

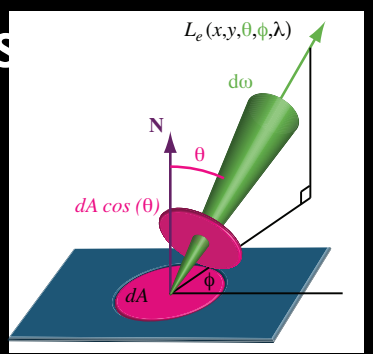
Radiometric & Photometric Terms

Radiometric quantity	Unit	Photometric quantity	Unit
Radiant energy (Q_e)	J (Joule)	Luminous energy (Q_v)	$lm.s$
Radiant flux (P_e)	$J.s^{-1} = W$ (Watt)	Luminous flux (P_v or F_v)	lm (lumen)
Radiant exitance (M_e)	$W.m^{-2}$	Luminous exitance (M_v)	$lm.m^{-2}$
Irradiance (E_e)	$W.m^{-2}$	Illuminance (E_v)	$lm.m^{-2}$ (lux)
Radiant intensity (I_e)	$W.sr^{-1}$	Luminous intensity (I_v)	$lm.sr^{-1} = cd$ (candela)
Radiance (L_e)	$W.m^{-2}.sr^{-1}$	Luminance (L_v)	$lm.m^{-2}.sr^{-1} = cd.m^{-2}$ (nit)

Very similar in nature, just weighted by V(λ)

Radiance and Luminance

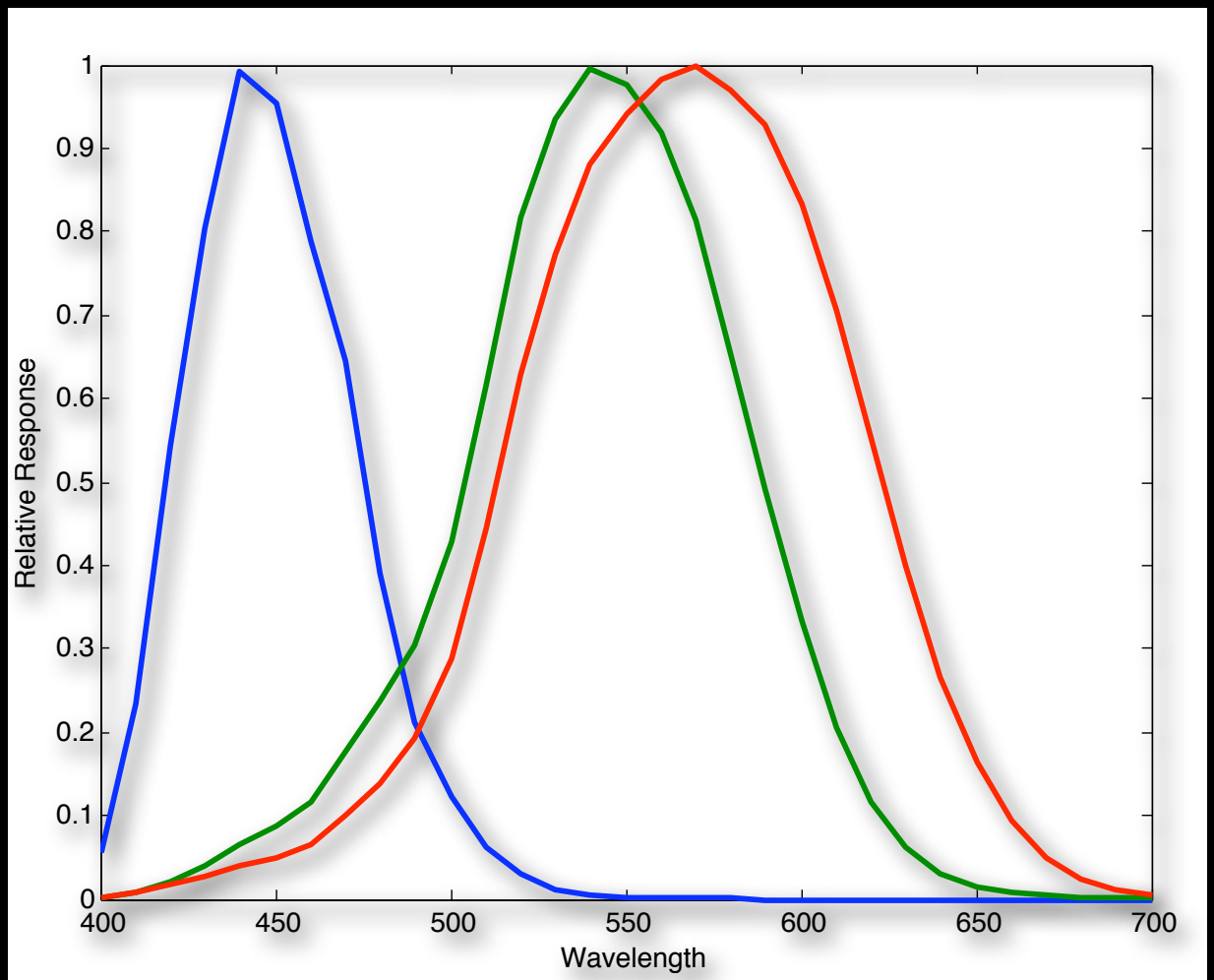
- Units of measurement as a function of time, unit surface area, and per solid angle.
- Watts · M^{-2} · St^{-1}
- Candelas · M^{-2}
- Basically cameras and the human visual system capture spectrally weighted radiance values



Speaking of spectrally
weighted radiance values...

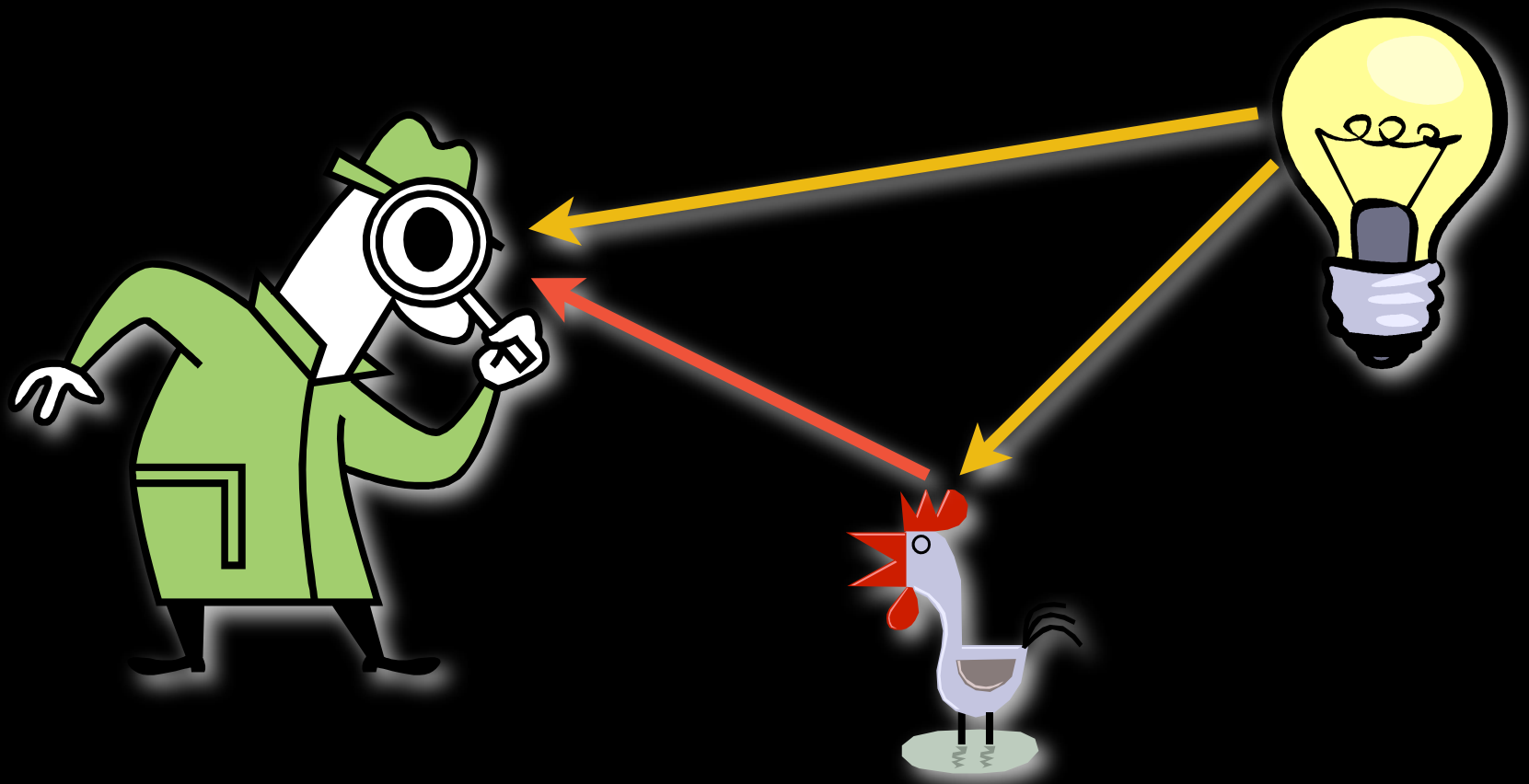
Colorimetry

Just as photometry is the measurement of light weighted by our photopic efficiency function, $V(\lambda)$, colorimetry is the measurement of color, or light weighted by our photoreceptors

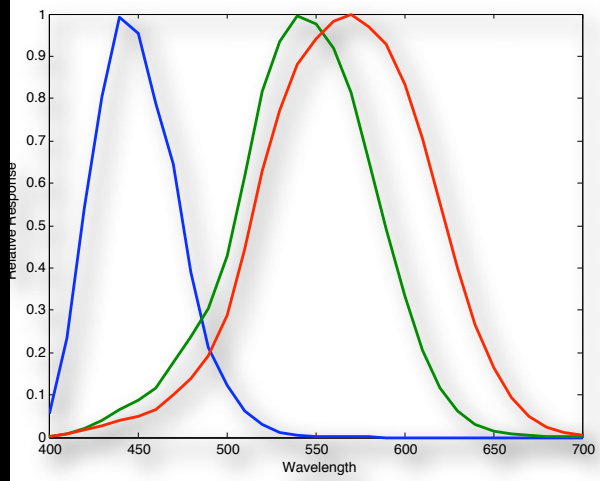


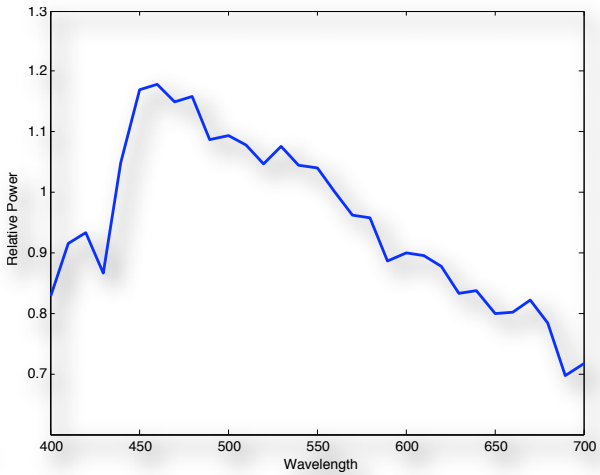
Your Cones - L M S

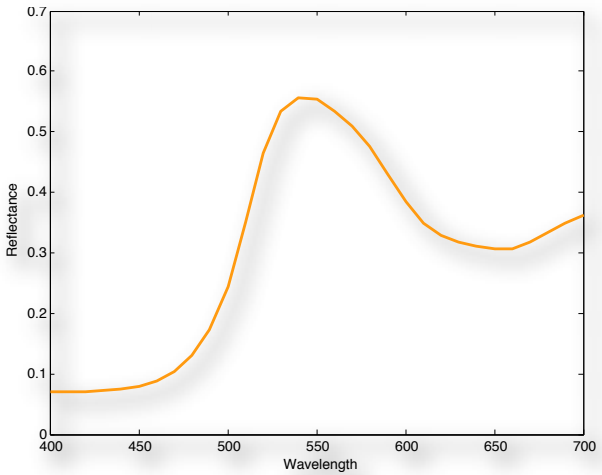
Color is a visual perception...that requires three[ish] things!



"Image" Formation

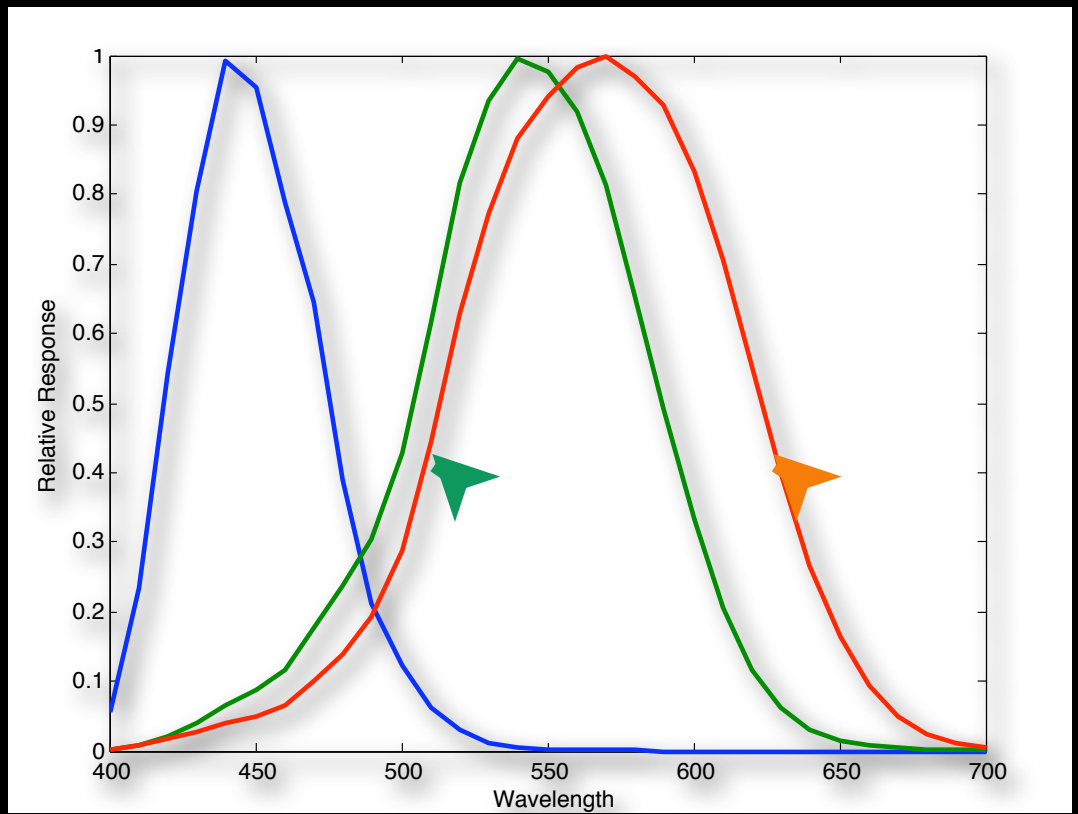






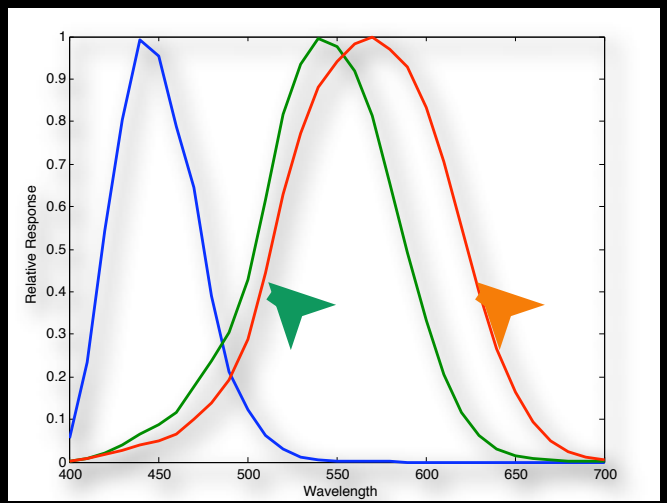
$$L = \int L(\lambda)S(\lambda)R(\lambda)d\lambda$$

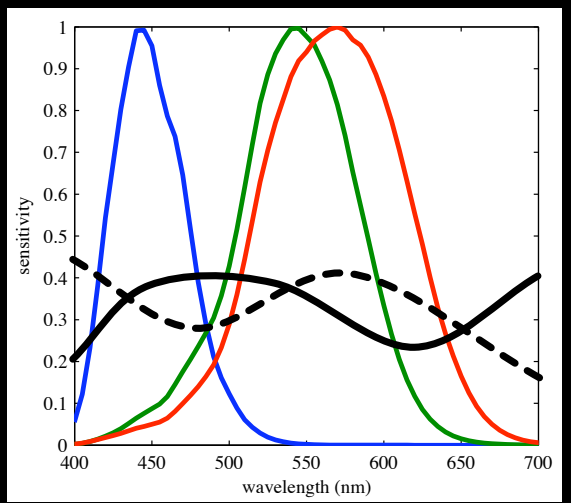
Univariance



Since your cones "integrate" all light, they do not care what the spectral properties of the light are

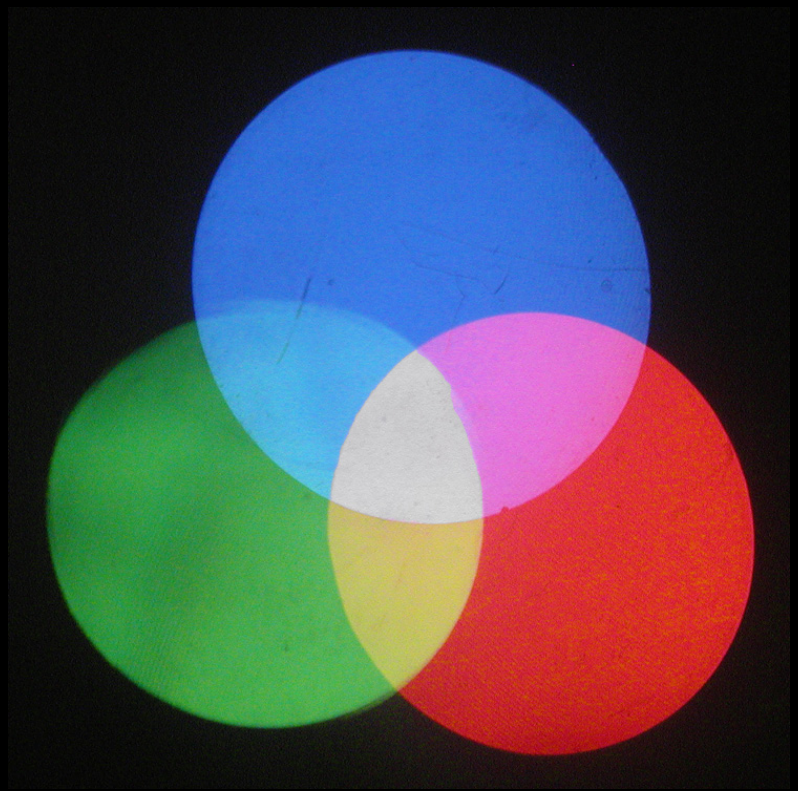
Metamerism



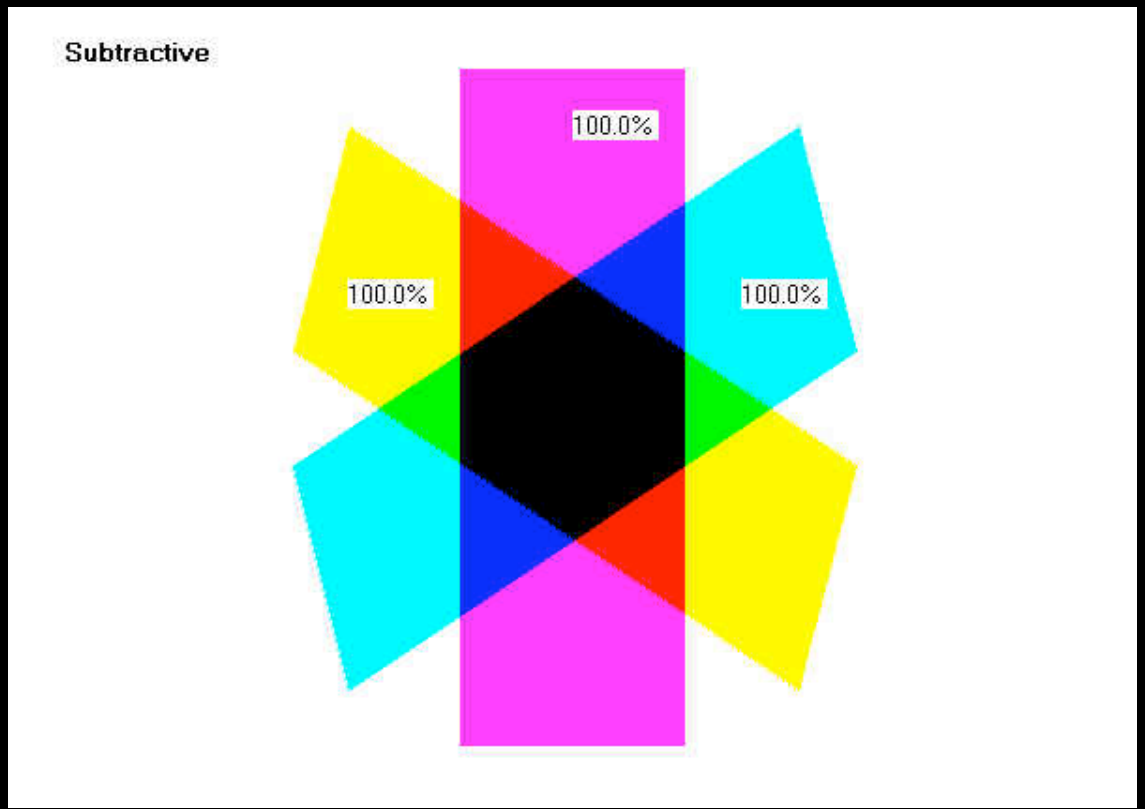


Because we have three "integrating" photoreceptors, it is possible to have two stimuli that have completely different spectral properties but the same visual response

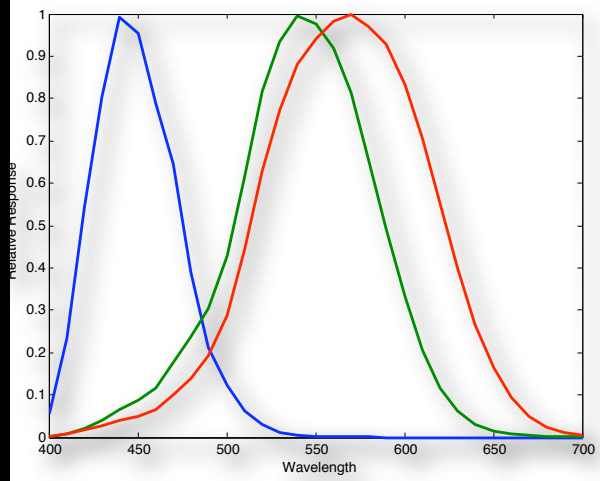
Additive Color Mixing

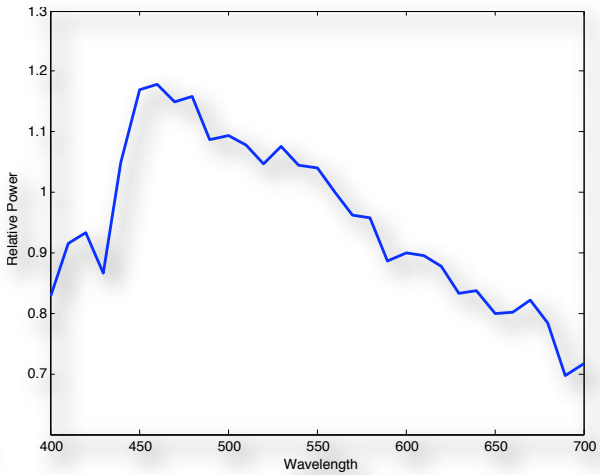


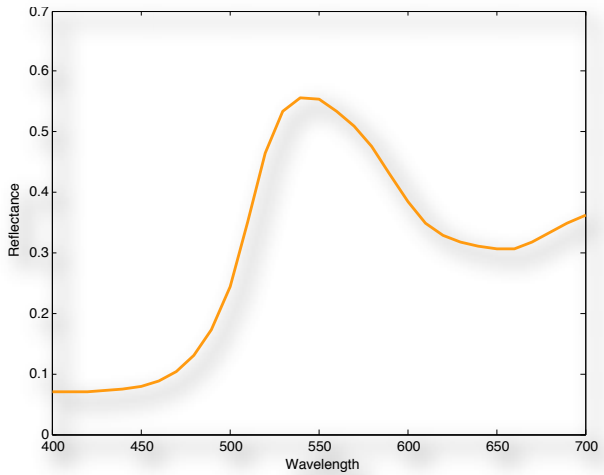
Subtractive Color Mixing



It's All Just Light!







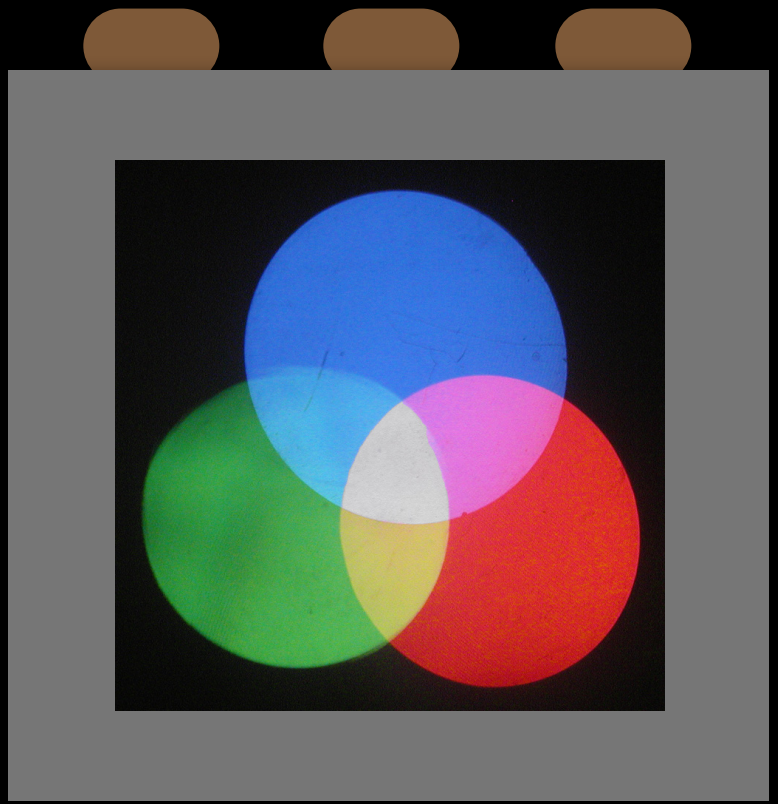
$$L = \int L(\lambda)S(\lambda)R(\lambda)d\lambda$$

How the "image" is formed before it hits your eye does not really matter!

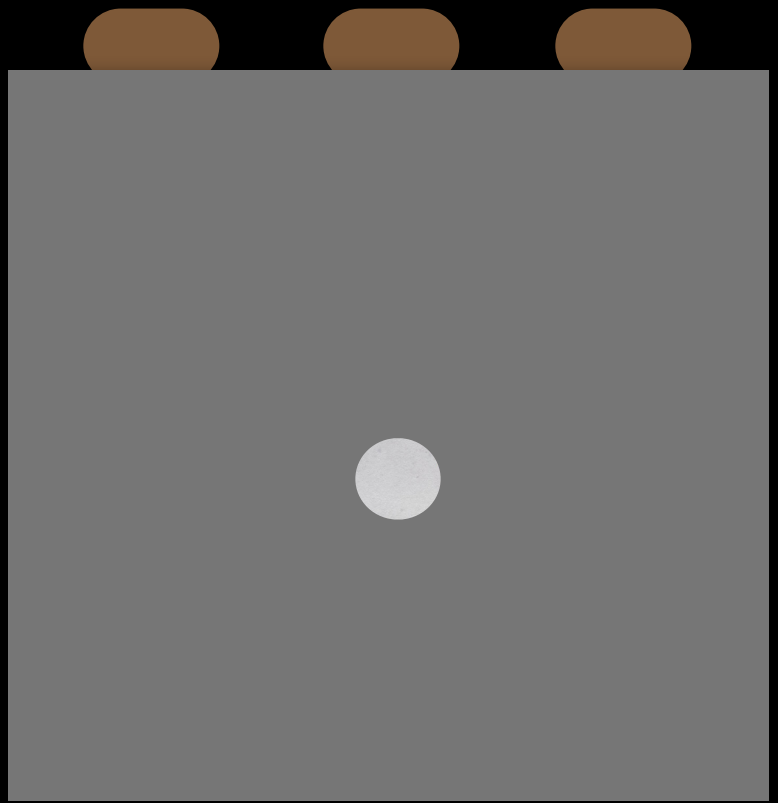
A fundamental problem is specifying an exact color

red, blue, pink, light-red, chartreuse, lapis, medium-gray, grey, freezing-cold blue lips, mauve

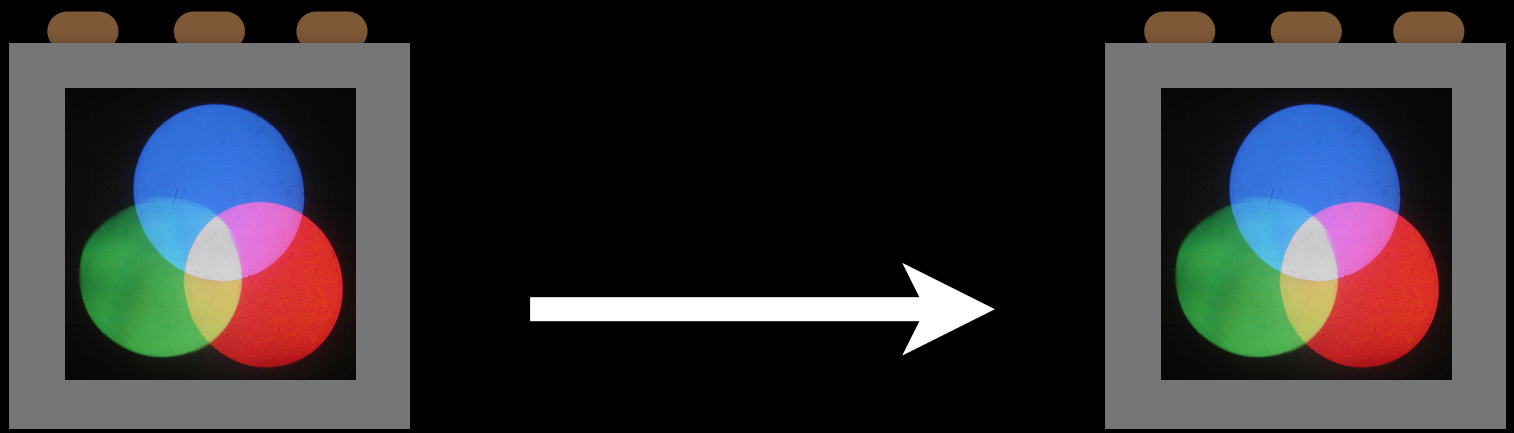
The Magic Color Box



The Magic Color Box



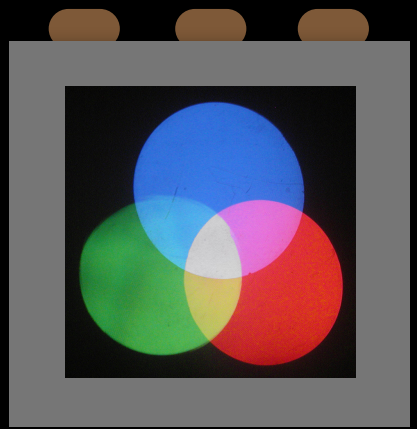
Visual Colorimetry



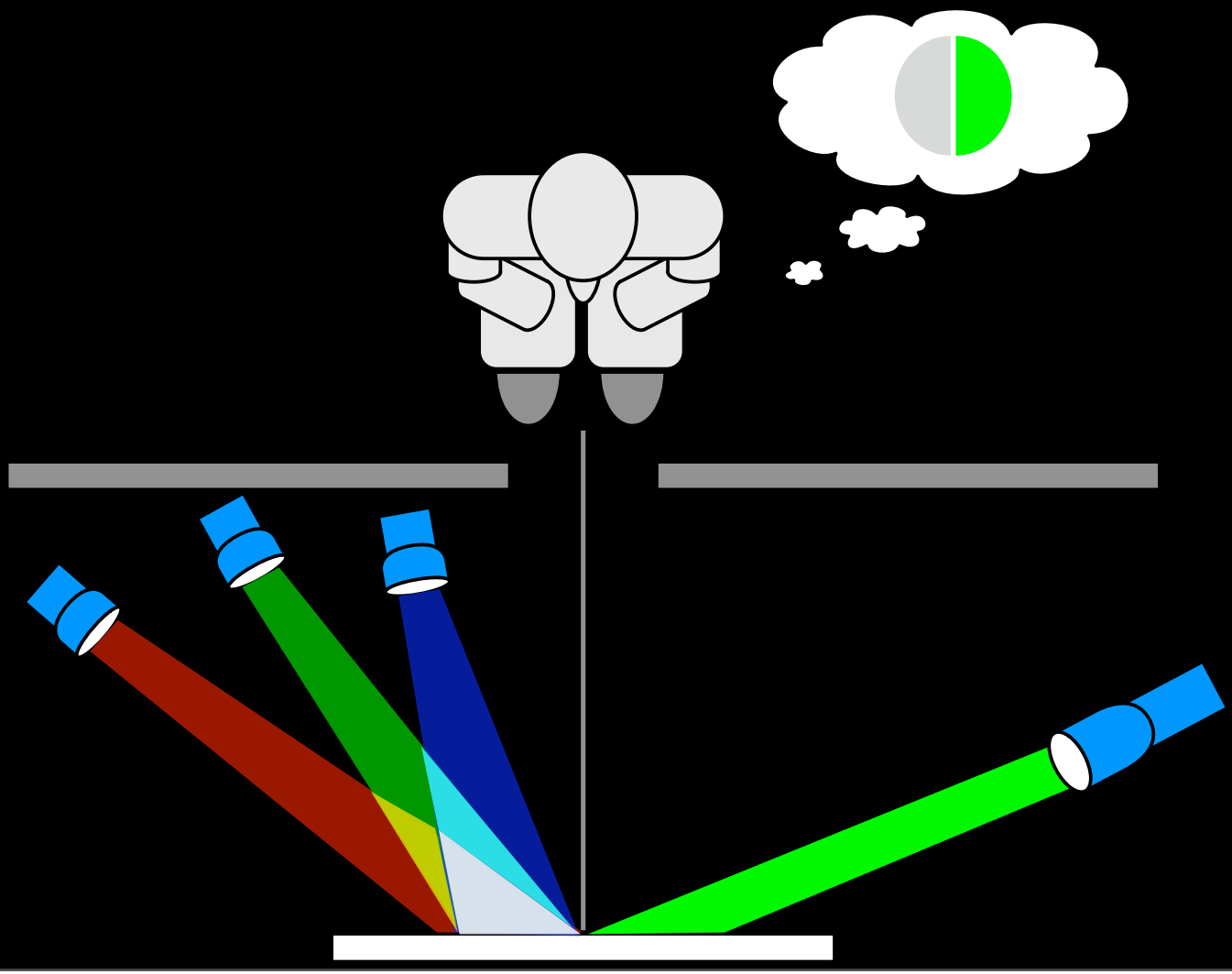
Dial in the color match, and send it to your coworker, who will dial it in on their machine

Visual Colorimetry

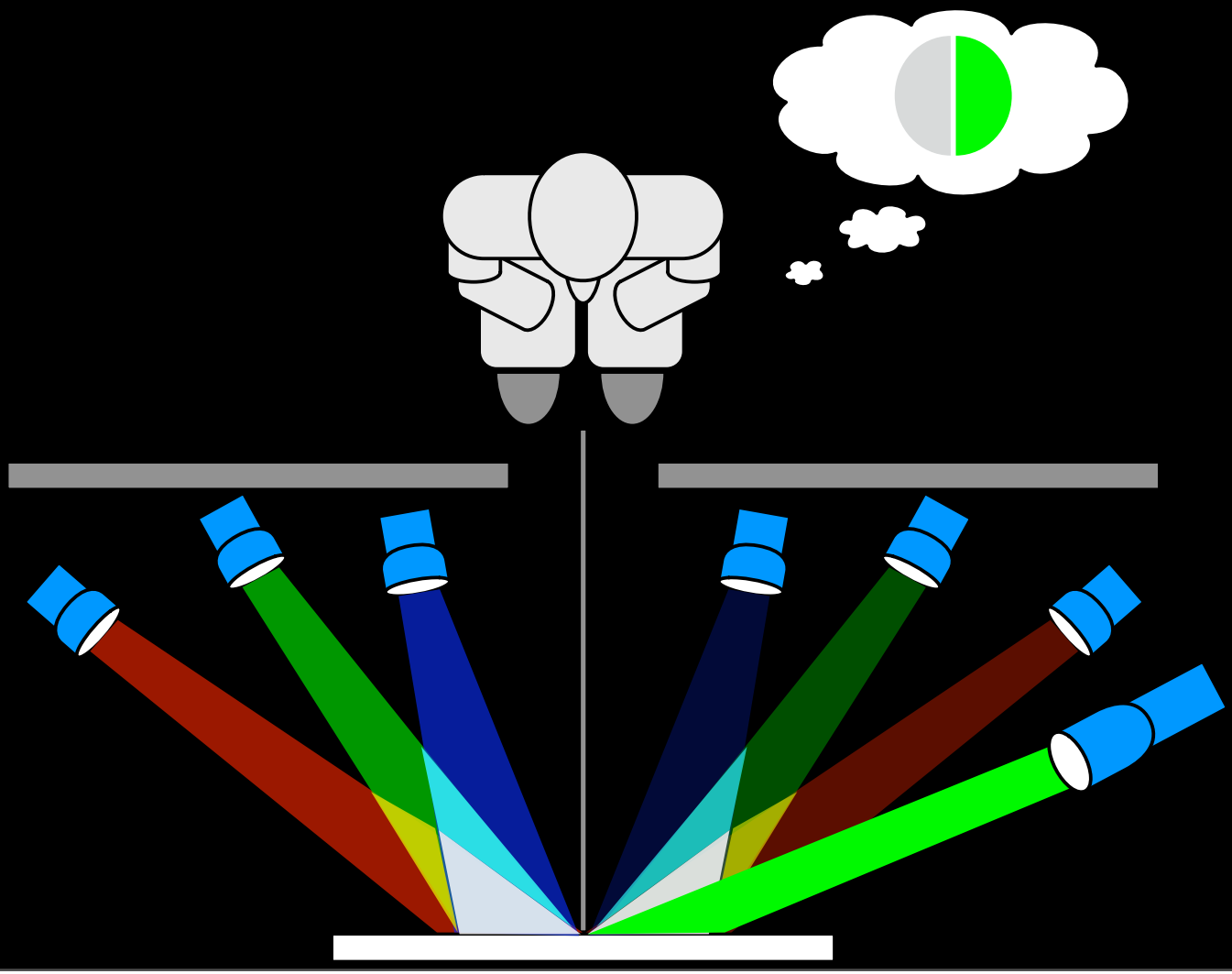
- Each device must be identical (think computer monitors!)
- The person doing the color "measurement" must make a visual match for every sample
- Each person's color vision is remarkably



Standardizing the Measurement

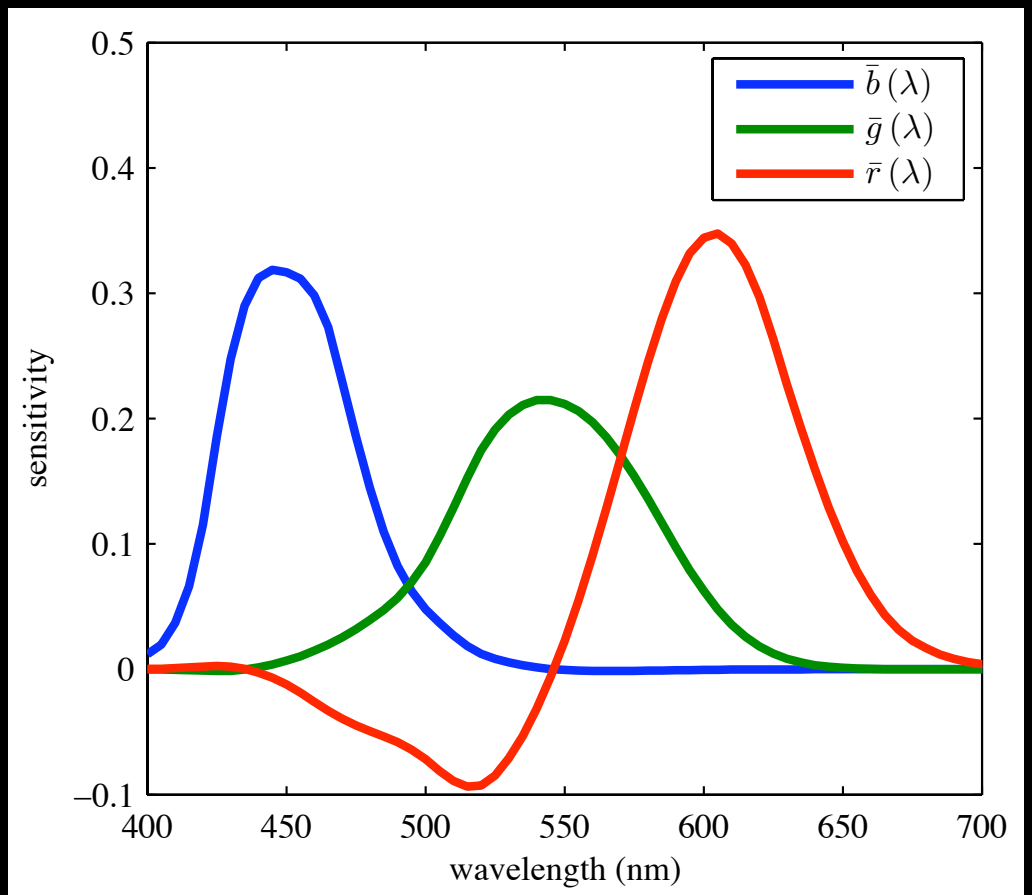


Standardizing the Measurement



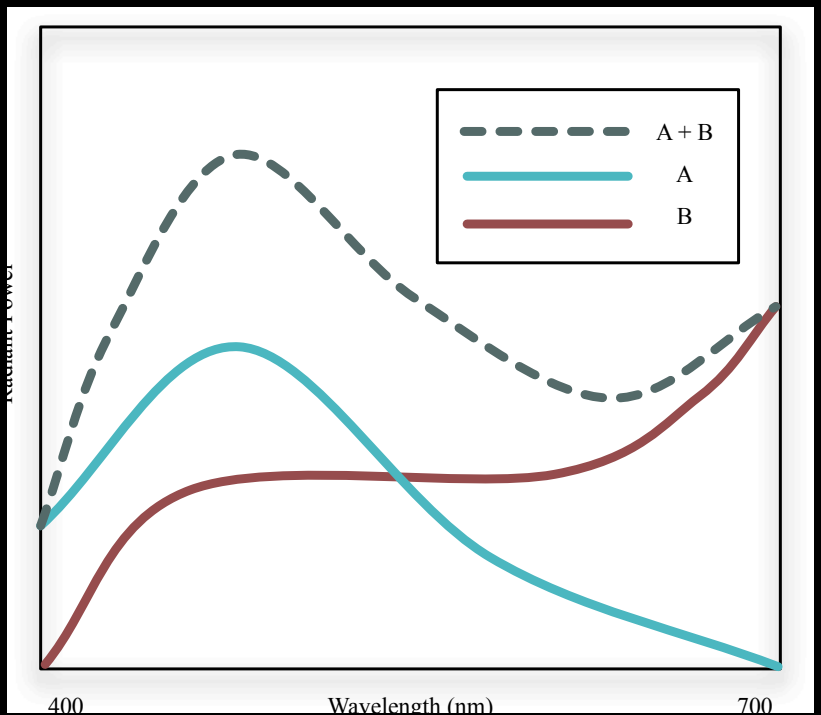
Color Matching Functions

435.8 nm
546.1 nm
700 nm



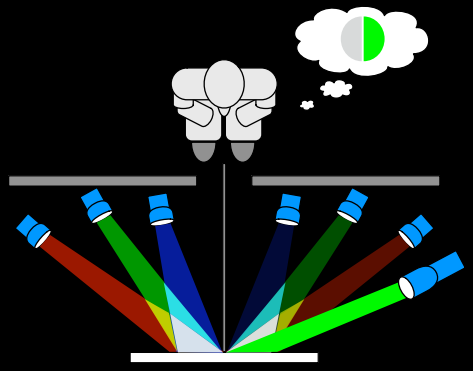
The amount of red, green & blue light needed to match any given wavelength

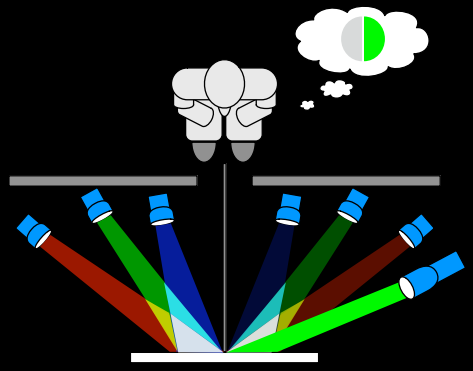
Additivity of Light

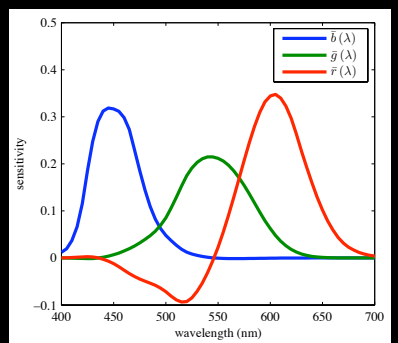


Grassmann's Laws...form the basis of Color Science

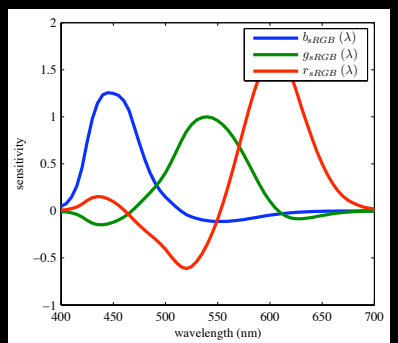
Transformation of Primaries







$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{bmatrix}$$



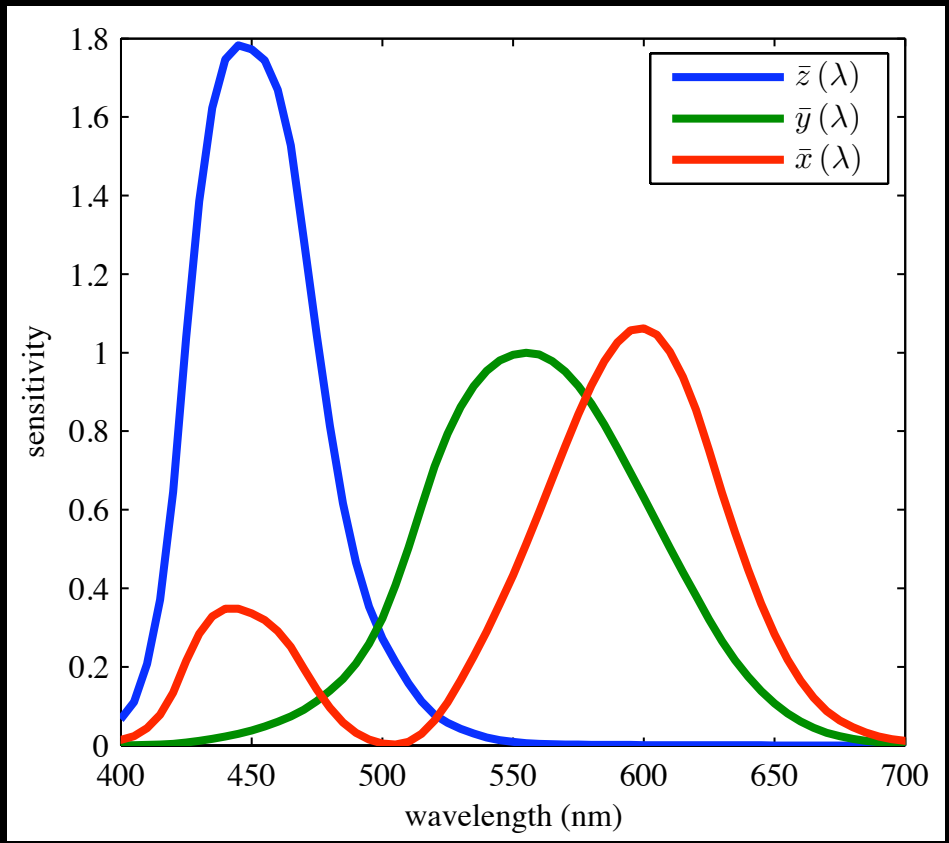
Since we know how much of any given "red, green, and blue" of one set of primaries(lights) it would take to match a color, we can use that to determine the ratio for any other given set of lights

Standard Colorimetric Observer

- We want to standardize the "primary" sets for color measurement and specification
- Ideally we would find a set of primaries that encompass all visual information (not possible in reality)
- Would like to tie back into the V(λ) photopic efficiency function

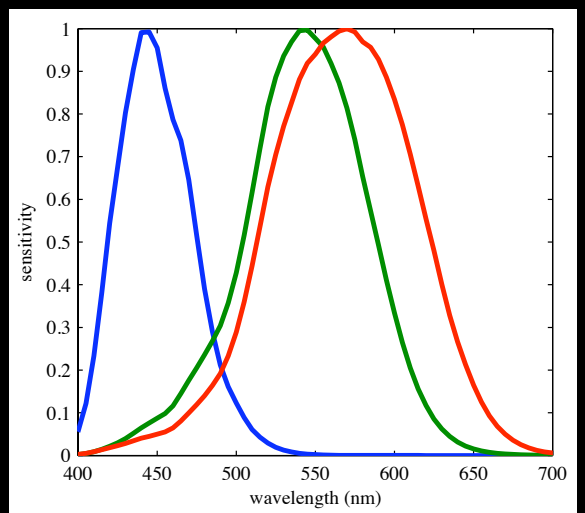
1931 Color Matching Functions

Negative numbers were really hard to deal with!

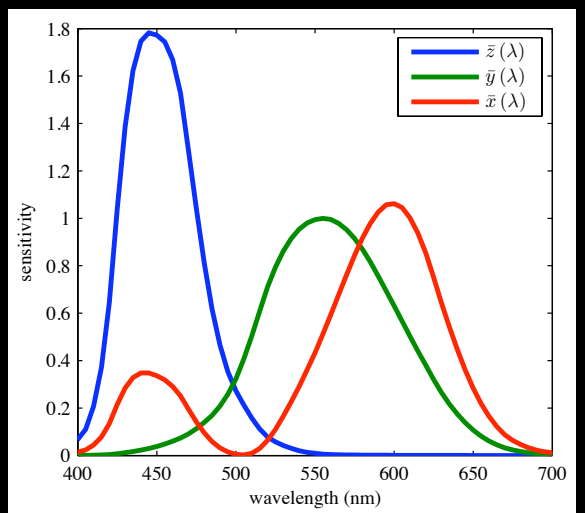


The amount of "imaginary" X, Y, or Z light it takes to match any given color

Imaginary Cones



$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{bmatrix}$$



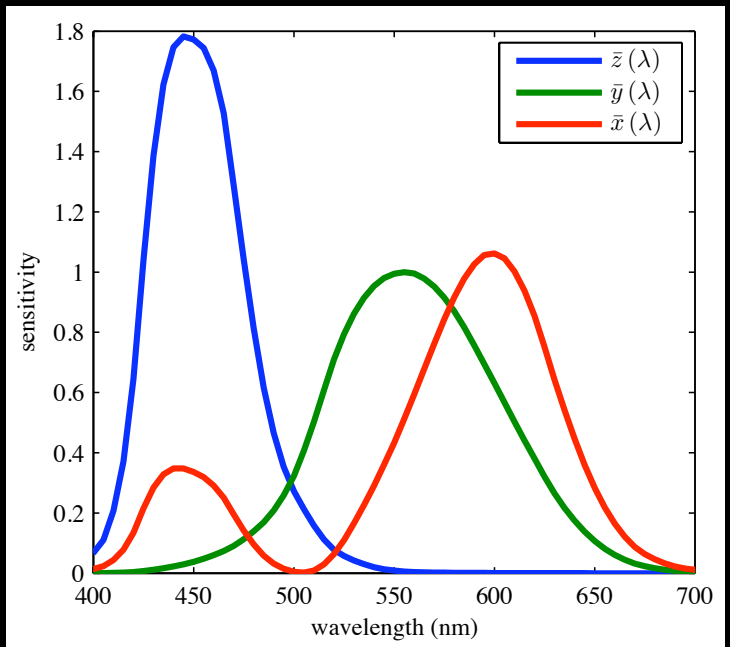
We can think of the color matching functions as a linear transform from our cones! (or close to one)

Measuring Color

$$X = \int_{\lambda} \Phi(\lambda) \cdot \bar{x}(\lambda) \cdot d\lambda$$

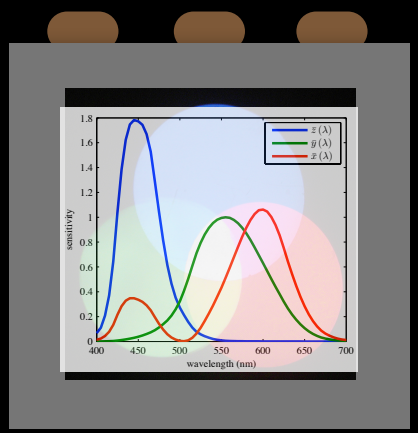
$$Y = \int_{\lambda} \Phi(\lambda) \cdot \bar{y}(\lambda) \cdot d\lambda$$

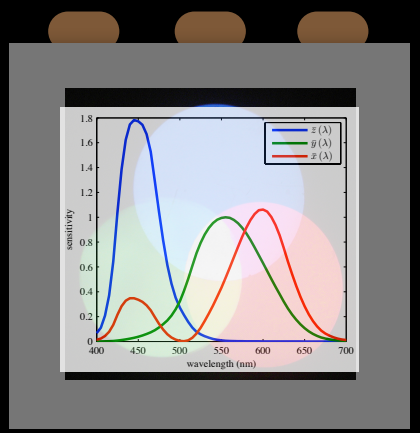
$$Z = \int_{\lambda} \Phi(\lambda) \cdot \bar{z}(\lambda) \cdot d\lambda$$



The integrated XYZ "Tristimulus Values" can be thought of as the measurement of the color!

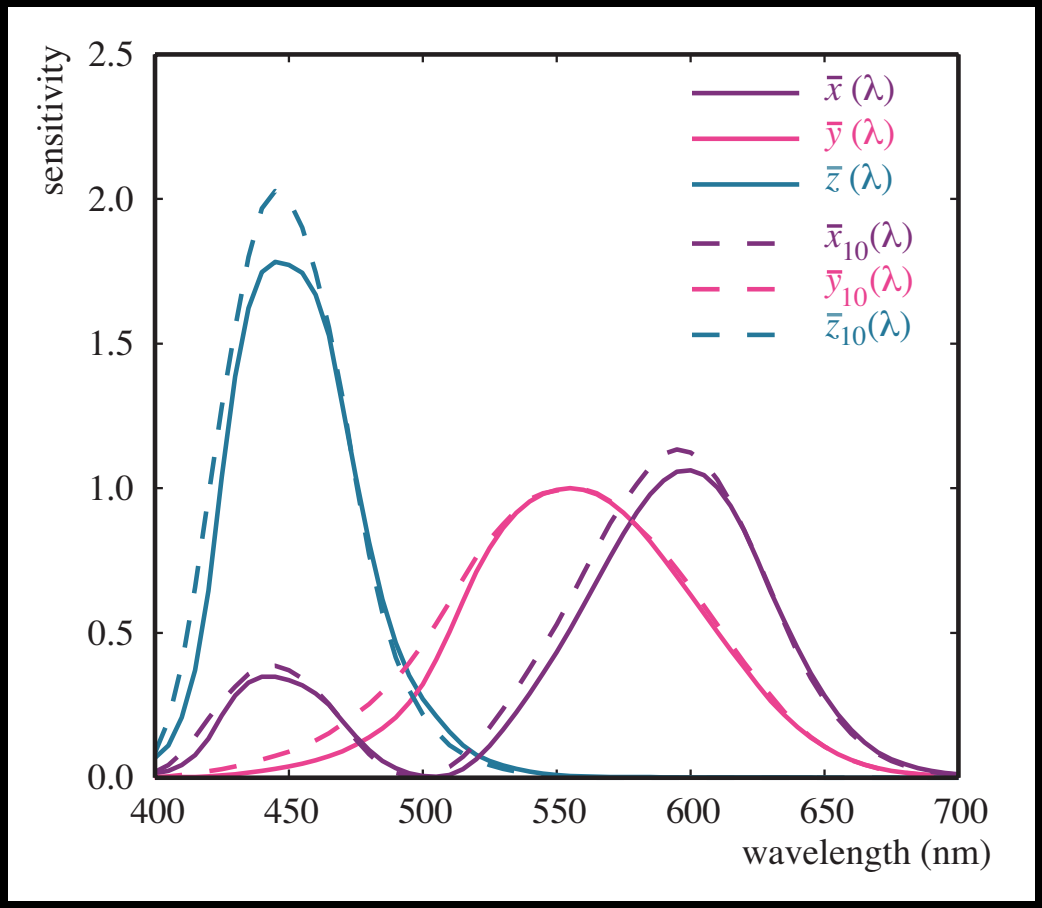
Colorimetry





If the measured XYZ of one stimuli matches the measured XYZ of the other, then the colors match! BUT...they have to be viewed under identical conditions!

Multiple Standard Observers...

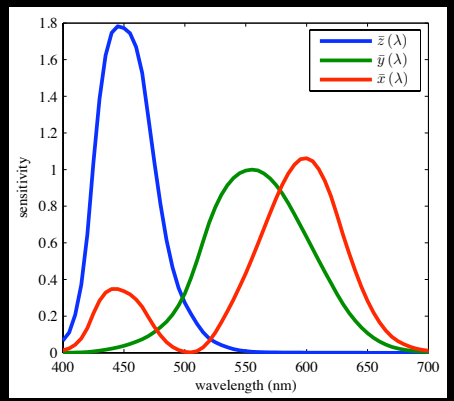


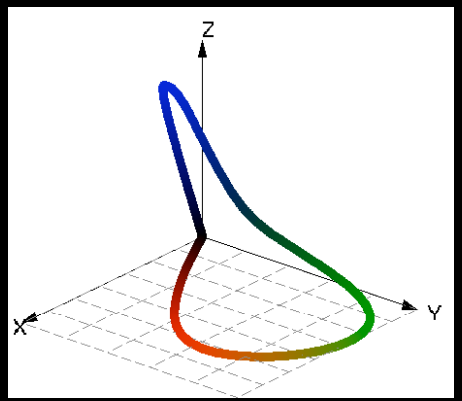
The original color matching experiments were performed with a small 2-degree field of view.

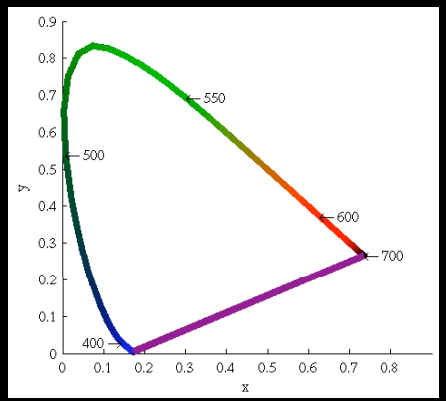
These have been suplemented with another set of functions for a larger 10-degree field, often called the 1964 Standard Observer

Clearly color is complicated, if changing the size changes our perception!

Chromaticity

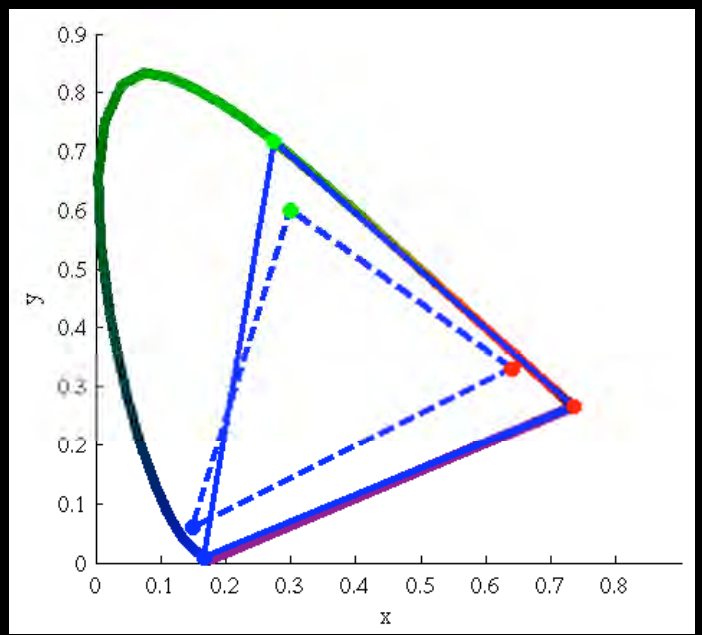






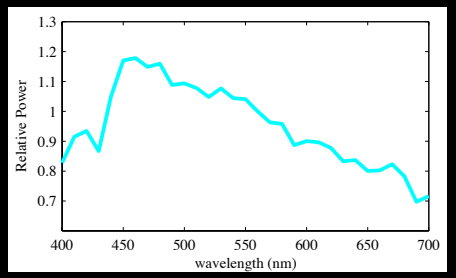
$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z}$$

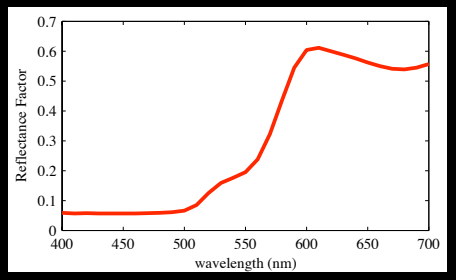
Chromaticity and displays...

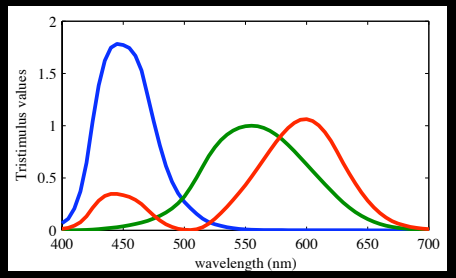


Chromaticity diagrams are often used to represent the "gamut" of reproducible colors for a display device. While they can be useful, they rarely tell the whole story and say nothing about the actual appearance of a color...

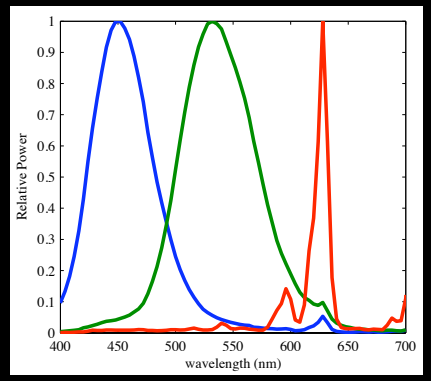
Colorimetry

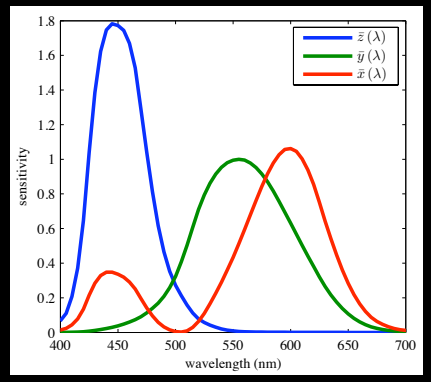






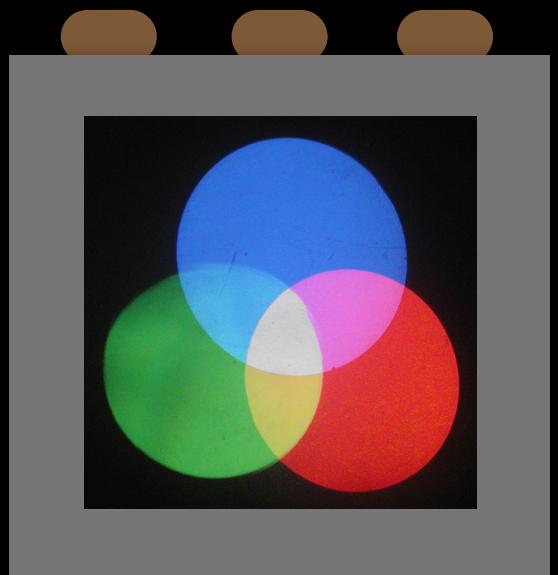
XYZ





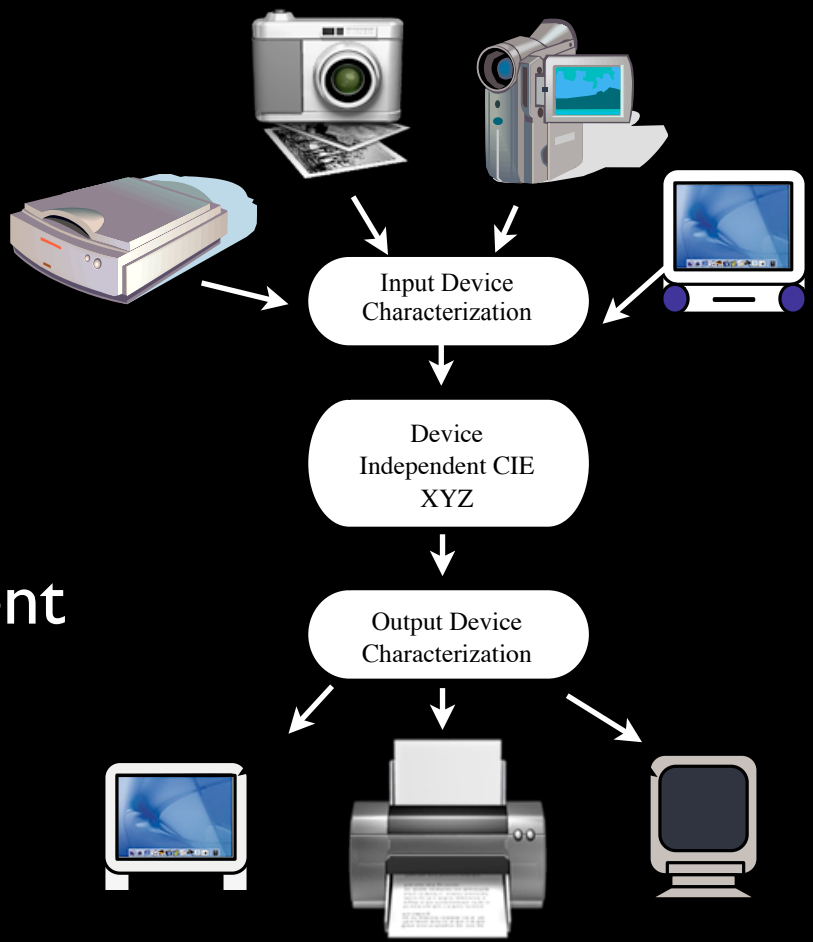
While it won't tell you what color look like, it will tell you when they match!

Example: Computer display



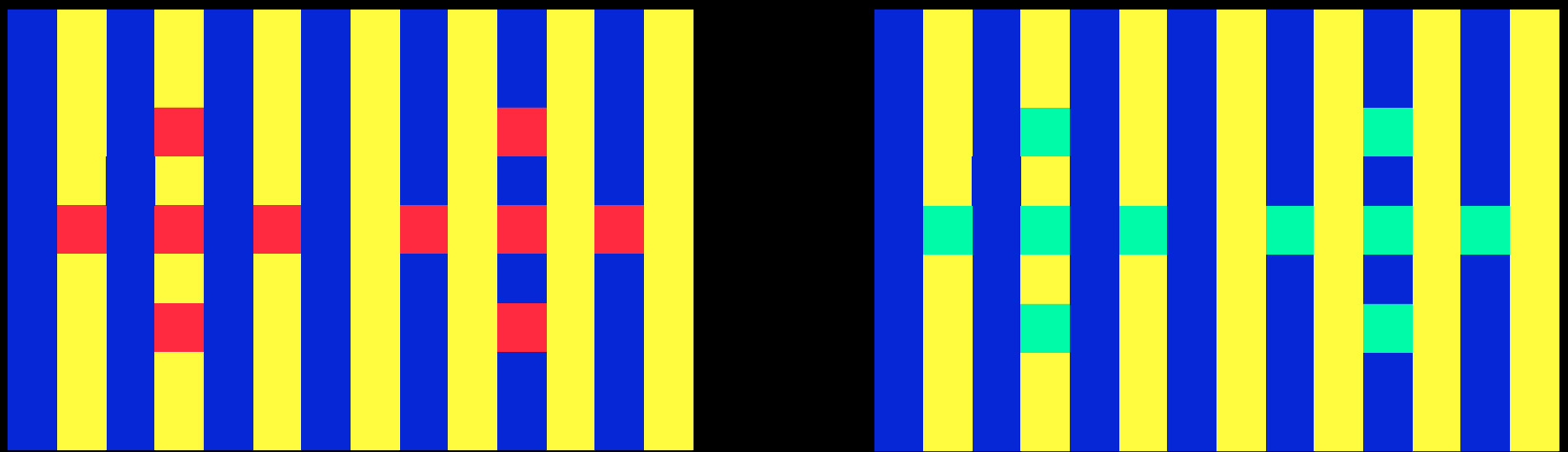
$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_R & X_G & X_B \\ Y_R & Y_G & Y_B \\ Z_R & Z_G & Z_B \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Colorimetry = Device Independent Color!

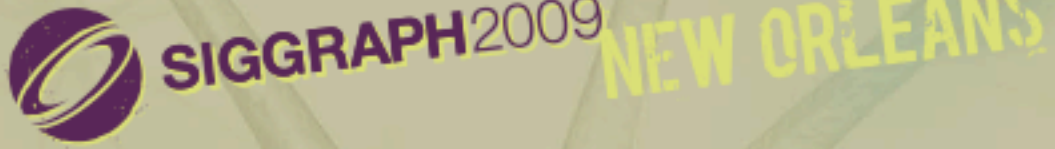


Color Management

Colorimetry ≠ Words



Advanced Colorimetry, or Color Appearance!



Color Spaces

Erik Reinhard

XYZ Color Space

- Properties
  - Tristimulus values are all positive
  - Primaries are imaginary
    - Cannot build a display with such primaries
    - Can build capture device with primaries close to XYZ primaries, but this would be expensive
  - Many possible XYZ values that do not correspond to a physical color
    - Inefficient for storage
    - Requires high bit-depth to preserve visual integrity

Device-Dependent Color

- Hence we cannot work with just CIE XYZ
- Transcoding into device-dependent spaces may be necessary

Color Spaces

- Reasons for development of different color spaces
  - Physical realizability
  - Efficient encoding
  - Perceptual uniformity
  - Intuitive color specification

Color Space Conversion

- For linear and additive trichromatic display devices, conversion to and from CIE XYZ is give by a 3x3 matrix
- Relies on Grassmann's Laws of additive color mixing
- Additional non-linear relationships can be defined, usually to minimize perceptual errors when storing values with a limited bit depth

Constructing a Color Space Transform

- Assume that we want to convert from CIE XYZ to a color space associated with a display device.
- Sending the color vectors (1,0,0), (0,1,0), and (0,0,1) to the display will yield the maximum chromaticity for each of the three channels when displayed alone.
- Measuring the display output will yield three pairs of CIE xy chromaticity coordinates, for example:

	R	G	B
x	0.6400	0.3000	0.1500
y	0.3300	0.6000	0.0600

White Point

- We also need to know the maximum luminance of the display, as well as the color obtained by sending the color vector (1,1,1) to the display.
- From this, we can compute the white point $X_W Y_W Z_W$

Matrix Construction

- Compute z = 1 - x - y to yield (x,y,z) triplets for each primary
- Construct set of equations with unknowns $(S_R\, S_G\, S_B)$:

$$X_W = x_R S_R + x_G S_G + x_B S_B$$
$$Y_W = y_R S_R + y_G S_G + y_B S_B$$
$$Z_W = z_R S_R + z_G S_G + z_B S_B$$

- Solve for these three unknowns

RGB to XYZ Conversion

- $$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x_R S_R & x_G S_G & x_B S_B \\ y_R S_R & y_G S_G & y_B S_B \\ z_R S_R & z_G S_G & z_B S_B \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- Luminance can be computed by evaluating the middle row

- $$Y = y_R S_R R + y_G S_G G + z_B S_B B$$

- To convert between XYZ and RGB, the above matrix can be inverted

Example: sRGB / HDTV Matrix

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.2405 & -1.5371 & -0.4985 \\ -0.9693 & 1.8706 & 0.0416 \\ 0.0556 & -0.2040 & 1.0572 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Non-Linear Encoding

- Additionally, the sRGB color space has a non-linear encoding:

$$R_{\mathrm{sRGB}} = \begin{cases} 1.055\,R^{1/2.4} - 0.055 & R > 0.0031308, \\ 12.92\,R & R \leq 0.0031308; \end{cases}$$

$$G_{\mathrm{sRGB}} = \begin{cases} 1.055\,G^{1/2.4} - 0.055 & G > 0.0031308, \\ 12.92\,G & G \leq 0.0031308; \end{cases}$$

$$B_{\mathrm{sRGB}} = \begin{cases} 1.055\,B^{1/2.4} - 0.055 & B > 0.0031308, \\ 12.92\,B & B \leq 0.0031308. \end{cases}$$

- This helps to minimize visibility of quantization artifacts

Device-Dependent Color

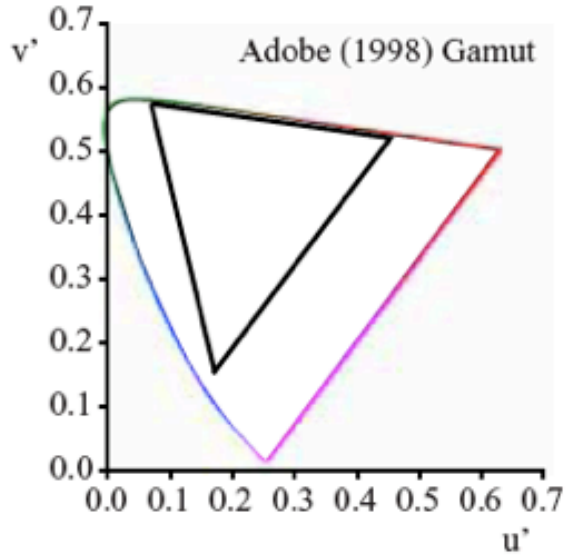
- Other RGB color spaces follow the same procedure:
  - Linear XYZ to RGB transform
  - Non-linear encoding

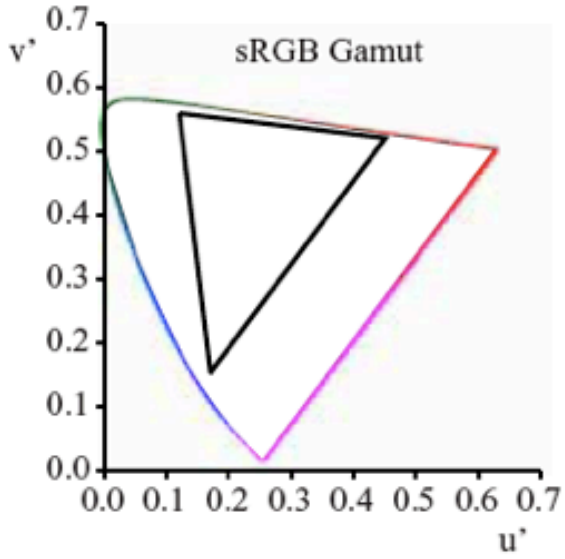
- Some examples:
  - sRGB
  - Adobe RGB (1998)
  - NTSC
  - PAL/SECAM
  - SMPTE
  - ...etc

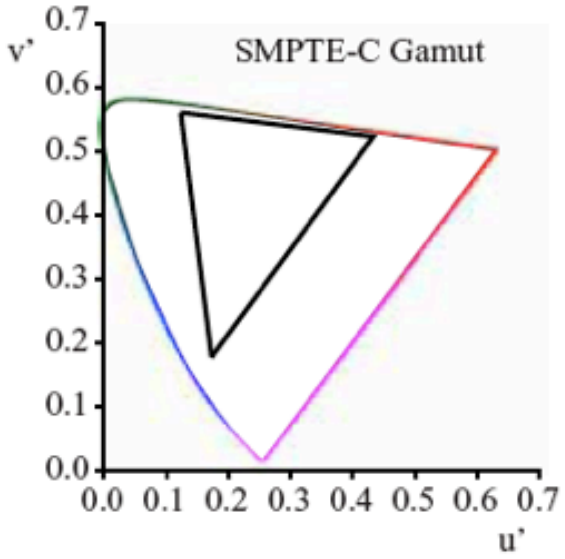
Device Dependent Color Spaces

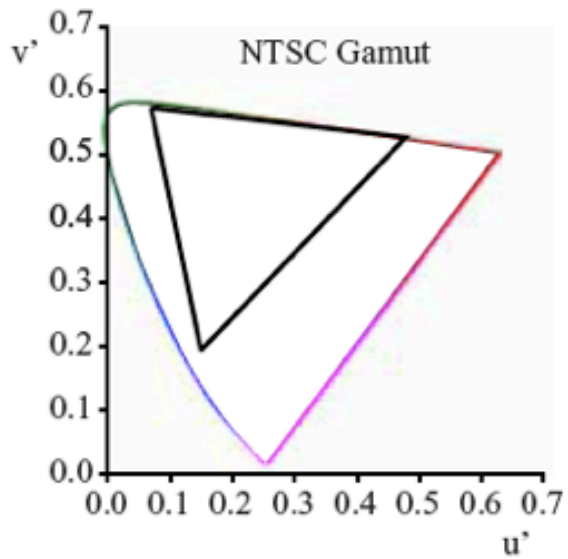
Color space	XYZ to RGB matrix	RGB to XYZ matrix	Non-linear transform
sRGB	$\begin{bmatrix} 3.2405 & -1.5371 & -0.4985 \\ -0.9693 & 1.8760 & 0.0416 \\ 0.0556 & -0.2040 & 1.0572 \end{bmatrix}$	$\begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix}$	$\gamma = 1/2.4 \approx 0.42$ $f = 0.055$ $s = 12.92$ $t = 0.0031308$
Adobe RGB (1998)	$\begin{bmatrix} 2.0414 & -0.5649 & -0.3447 \\ -0.9693 & 1.8760 & 0.0416 \\ 0.0134 & -0.1184 & 1.0154 \end{bmatrix}$	$\begin{bmatrix} 0.5767 & 0.1856 & 0.1882 \\ 0.2974 & 0.6273 & 0.0753 \\ 0.0270 & 0.0707 & 0.9911 \end{bmatrix}$	$\gamma = \frac{1}{2\frac{31}{256}} \approx \frac{1}{2.2}$ $f =$ N.A. $s =$ N.A. $t =$ N.A.
HDTV (HD-CIF)	$\begin{bmatrix} 3.2405 & -1.5371 & -0.4985 \\ -0.9693 & 1.8760 & 0.0416 \\ 0.0556 & -0.2040 & 1.0572 \end{bmatrix}$	$\begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix}$	$\gamma = 0.45$ $f = 0.099$ $s = 4.5$ $t = 0.018$
NTSC (1953)/ ITU-R BT.601-4	$\begin{bmatrix} 1.9100 & -0.5325 & -0.2882 \\ -0.9847 & 1.9992 & -0.0283 \\ 0.0583 & -0.1184 & 0.8976 \end{bmatrix}$	$\begin{bmatrix} 0.6069 & 0.1735 & 0.2003 \\ 0.2989 & 0.5866 & 0.1145 \\ 0.0000 & 0.0661 & 1.1162 \end{bmatrix}$	$\gamma = 0.45$ $f = 0.099$ $s = 4.5$ $t = 0.018$
PAL/SECAM	$\begin{bmatrix} 3.0629 & -1.3932 & -0.4758 \\ -0.9693 & 1.8760 & 0.0416 \\ 0.0679 & -0.2289 & 1.0694 \end{bmatrix}$	$\begin{bmatrix} 0.4306 & 0.3415 & 0.1783 \\ 0.2220 & 0.7066 & 0.0713 \\ 0.0202 & 0.1296 & 0.9391 \end{bmatrix}$	$\gamma = 0.45$ $f = 0.099$ $s = 4.5$ $t = 0.018$
SMPTE-C	$\begin{bmatrix} 3.5054 & -1.7395 & -0.5440 \\ -1.0691 & 1.9778 & 0.0352 \\ 0.0563 & -0.1970 & 1.0502 \end{bmatrix}$	$\begin{bmatrix} 0.3936 & 0.3652 & 0.1916 \\ 0.2124 & 0.7010 & 0.0865 \\ 0.0187 & 0.1119 & 0.9582 \end{bmatrix}$	$\gamma = 0.45$ $f = 0.099$ $s = 4.5$ $t = 0.018$

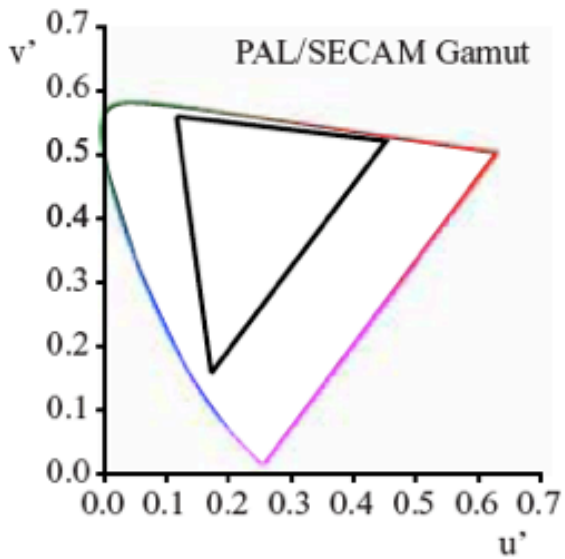
Color Gamuts

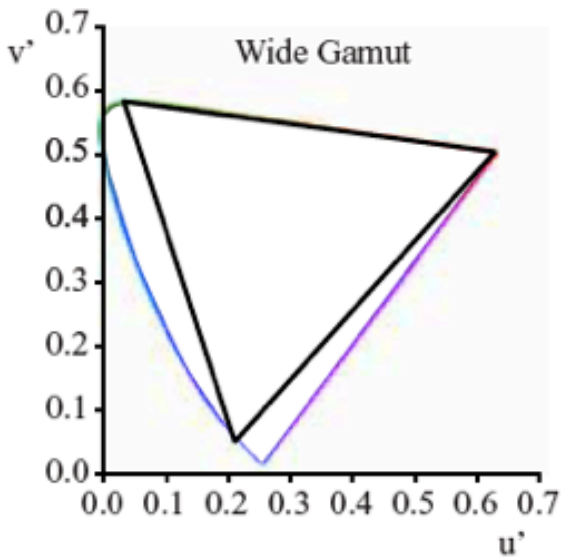




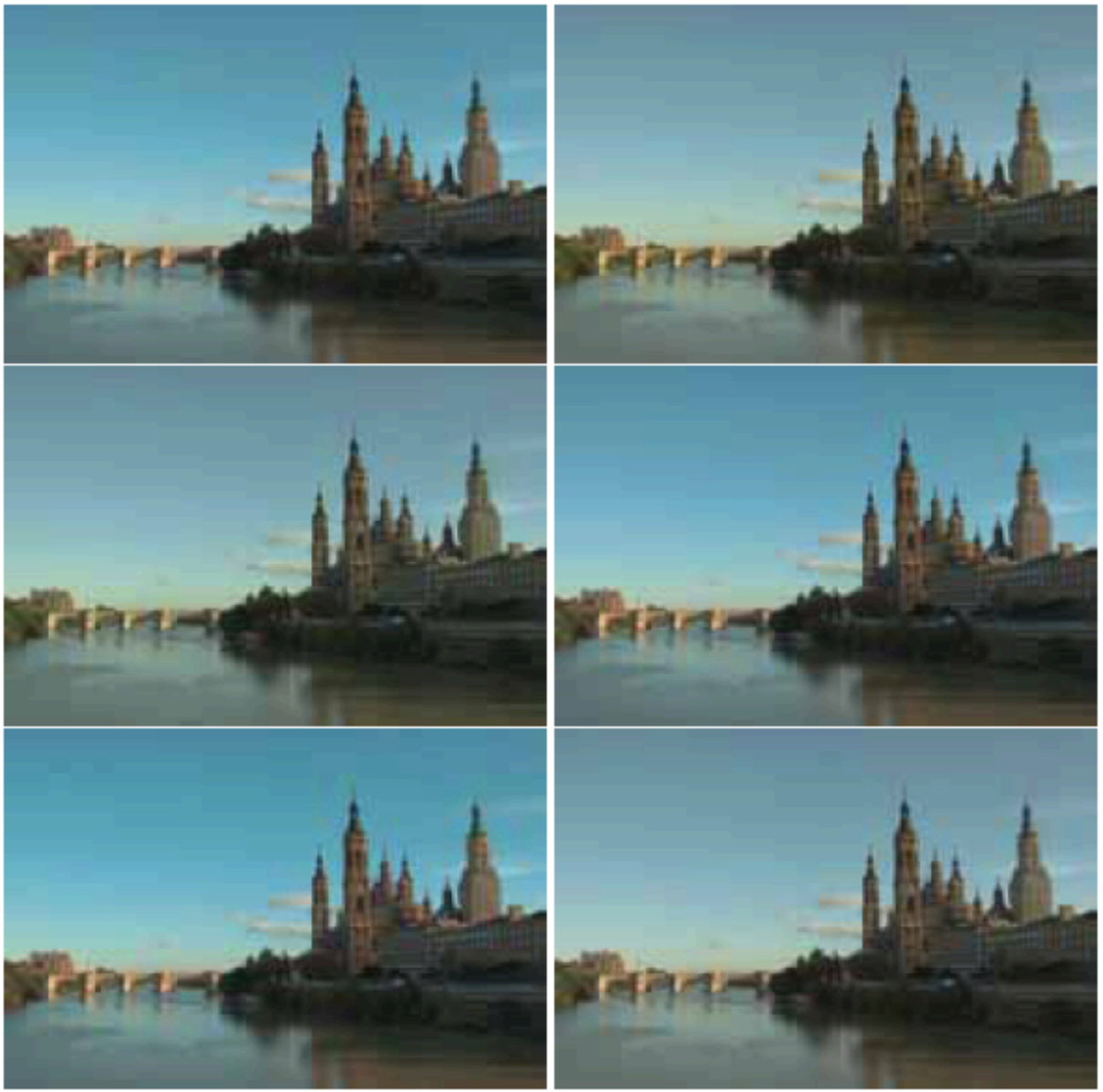








Different RGB spaces



Other Color Spaces

- Television and Video (e.g. YIQ)
- Printing (e.g. CMYK)
- Hue Saturation Lightness Spaces (e.g. HSB)
- Luminance Chrominance Spaces (e.g. Yuv)
- Color Opponent Spaces (e.g. CIELAB)

Color Opponent Spaces

- One achromatic channel (luminance)
- Two color opponent channels, usually:
  - red-green
  - yellow-blue
- Examples: CIELAB, CIELUV, IPT
- Aim is perceptual uniformity
  - Allows Euclidian distance to be used to assess color differences

CIE L*a*b*

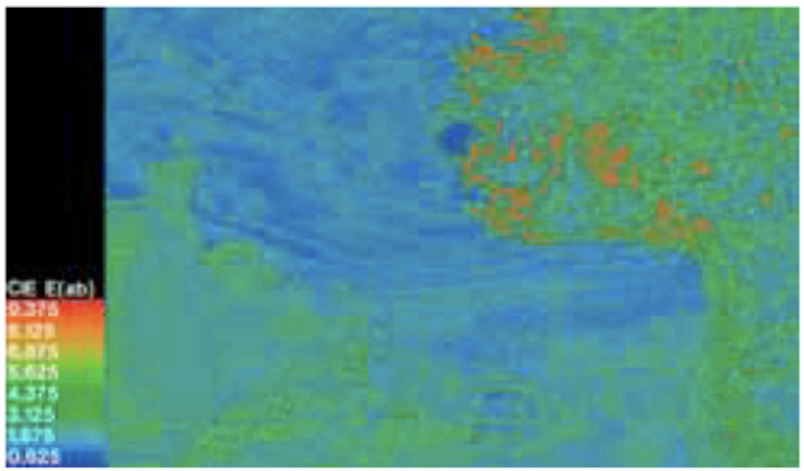
$$\begin{bmatrix} L^* \\ a^* \\ b^* \end{bmatrix} = \begin{bmatrix} 0 & 116 & 0 & -16 \\ 500 & -500 & 0 & 0 \\ 0 & 200 & -200 & 0 \end{bmatrix} \begin{bmatrix} f(X/X_n) \\ f(Y/Y_n) \\ f(Z/Z_n) \\ 1 \end{bmatrix}$$

$$f(r) = \begin{cases} \sqrt[3]{r} & \text{for } r > 0.008856, \\ 7.787r + \dfrac{16}{116} & \text{for } r \leq 0.008856. \end{cases}$$

- Input:
  - XYZ tristimulus value
  - $X_nY_nZ_n$ tristimulus value of white reflecting patch illuminated by a known illuminant

CIE L*a*b* Color Difference

- Euclidian distance: $$\Delta E^*_{ab} = \left[(\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2\right]^{1/2}$$



IPT Color Space

- Better perceptual uniformity (especially in hue), based on newer perceptual data
- In cone excitation space, apply non-linearity, then convert to opponent space
- Step 1:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.4002 & 0.7075 & -0.0807 \\ -0.2280 & 1.1500 & 0.0612 \\ 0.0000 & 0.0000 & 0.9184 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

IPT Color Space

- Step 2:

$$L' = \begin{cases} L^{0.43} & \text{if } L \geq 0, \\ -(-L)^{0.43} & \text{if } L < 0; \end{cases}$$

$$M' = \begin{cases} M^{0.43} & \text{if } M \geq 0, \\ -(-M)^{0.43} & \text{if } M < 0; \end{cases}$$

$$S' = \begin{cases} S^{0.43} & \text{if } S \geq 0, \\ -(-S)^{0.43} & \text{if } S < 0. \end{cases}$$

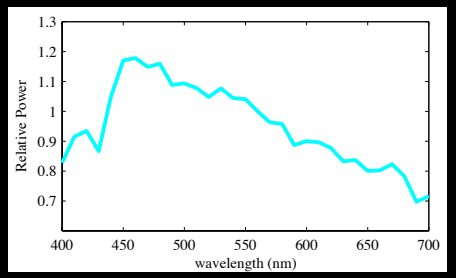
- Step 3:

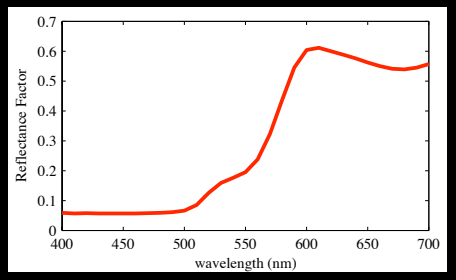
$$\begin{bmatrix} I \\ P \\ T \end{bmatrix} = \begin{bmatrix} 0.4000 & 0.4000 & 0.2000 \\ 4.4550 & -4.8510 & 0.3960 \\ 0.8056 & 0.3572 & -1.1628 \end{bmatrix} \begin{bmatrix} L' \\ M' \\ S' \end{bmatrix}$$

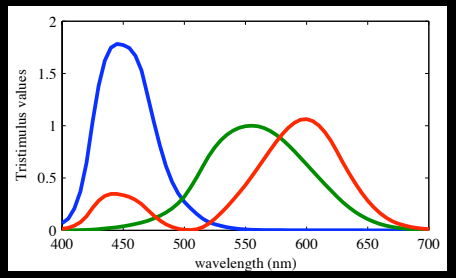
Color Appearance: Or Advanced Colorimetry

Garrett M. Johnson

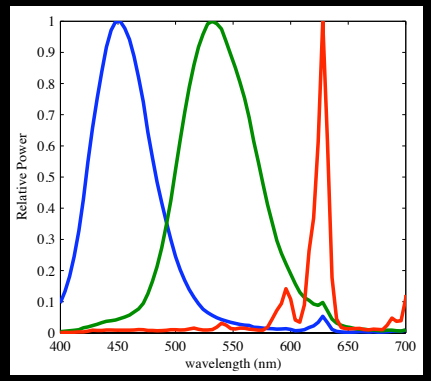
Colorimetry

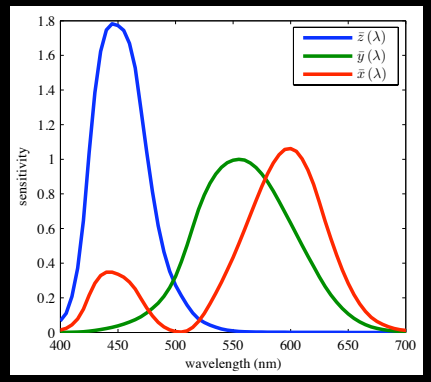






XYZ





While it won't tell you what color look like, it will tell you when they match!

But what color is it?

- XYZ Colorimetry tells us when two stimuli match
- The two stimuli must be viewed under identical viewing conditions to guarantee a match
- We cannot describe what any color actually looks like, based upon its XYZ values

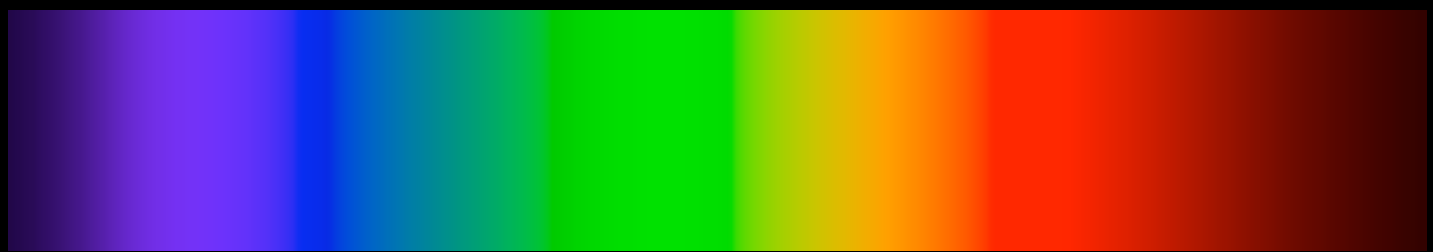
Color Appearance = Vocabulary

- Brightness, Lightness
- Colorfulness, Chroma, Saturation
- Hue

We need 5-6 of these to fully describe color!

Hue

The degree to which a stimulus can be described as similar to or different from stimuli that are described as red, green, blue, and yellow.



Only 4 physiological hues!

Brightness

The perceived quantity of light emanating from a stimulus. (Absolute Quantity)

Lightness

The brightness of a stimulus relative to the brightness of a stimulus that appears white under similar viewing situations.

Colorfulness

The perceived quantity of hue content (difference from gray) in a stimulus.

Colorfulness increases with luminance.

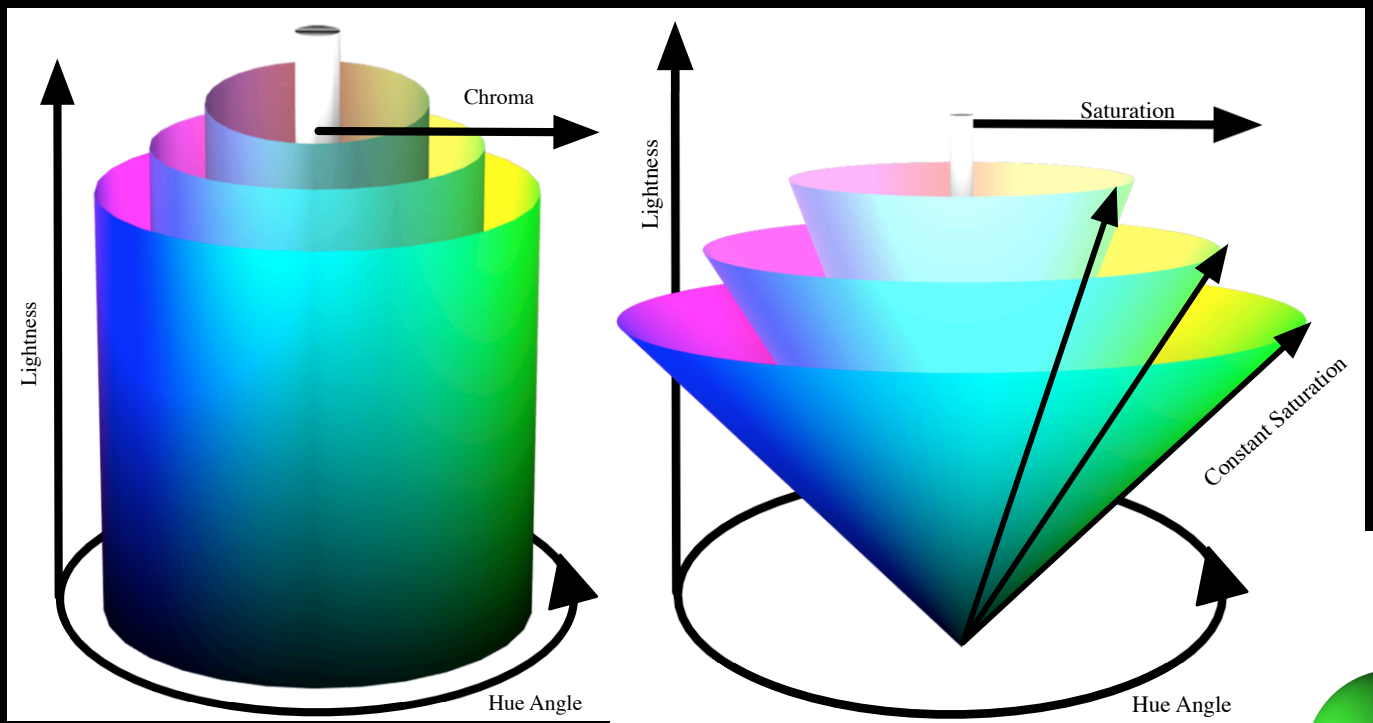
Chroma

The colorfulness of a stimulus relative to the brightness of a stimulus that appears white under similar viewing conditions.

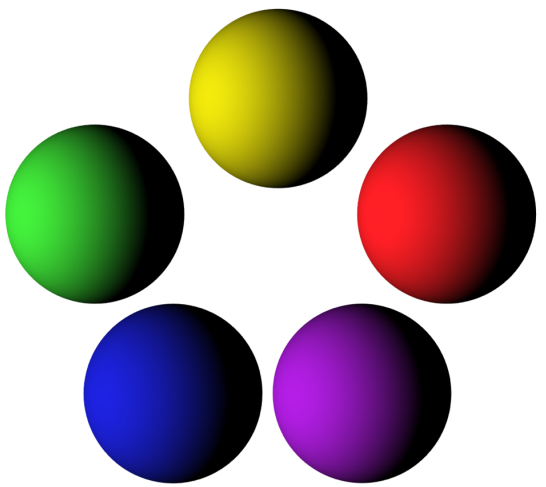
Saturation

The colorfulness of a stimulus relative to its own brightness.

Chroma/Saturation



A shadowed object has constant saturation, but decreasing chroma



Definitions in "Equations"

Chroma = (Colorfulness)/(Brightness of White)

Saturation = (Colorfulness)/(Brightness)

Lightness = (Brightness)/(Brightness of White)

Saturation = (Chroma)/(Lightness)
= [(Colorfulness)/(Brightness of White)][(Brightness of White)/(Brightness)]
=(Colorfulness)/(Brightness)

How Many Terms?

Any color perception can be described completely by its:

- Brightness
- Lightness
- Colorfulness
- Chroma
- Hue

and only one of brightness or colorfulness is required to derive the others.

In general, the relative appearance attributes are adequate for object colors in typical viewing environments:

- Lightness
- Chroma
- Hue

Saturation is often redundant.

Lightness/Chroma vs. Brightness/Colorfulness

When predicting color matches across different viewing conditions, Lightness-Chroma matches are not identical to Brightness-Colorfulness matches.

For related colors, and typical conditions, Lightness-Chroma matching (and therefore reproduction) is the only practical choice.

Reproduction at Higher Luminance

Original (50 cd/m^2)

Lightness/Chroma

Brightness/Colorfulness

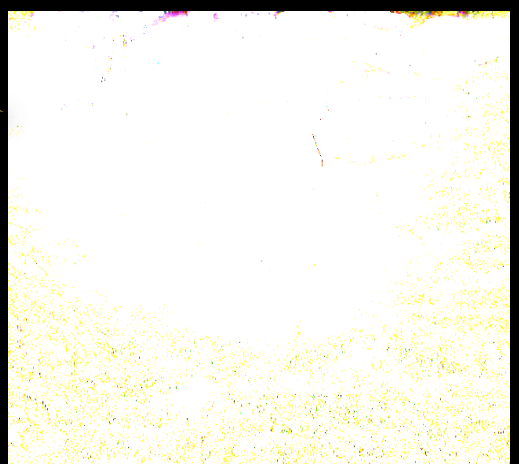
Reproductions (5000 cd/m^2)

Reproduction at Lower Luminance

Original (5000 cd/m^2)

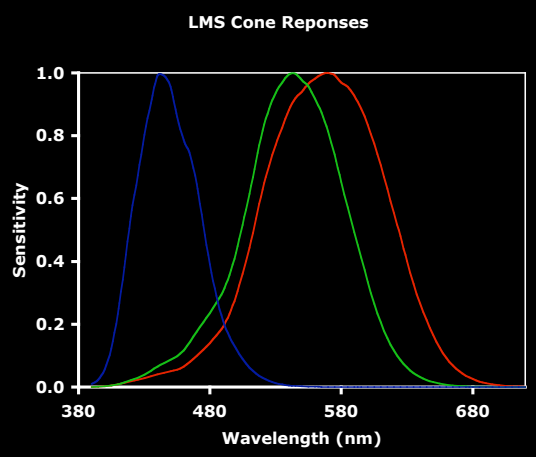
Lightness/Chroma

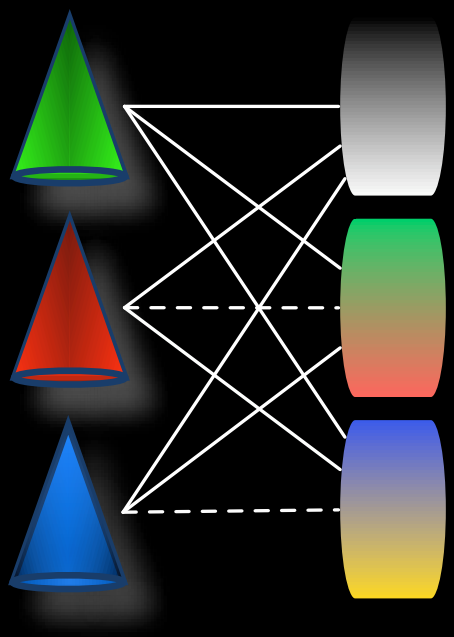
Brightness/Colorfulness

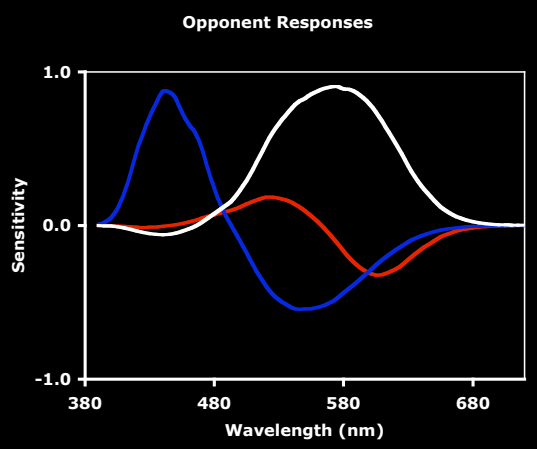


Reproductions (50 cd/m^2)

Color Opponency







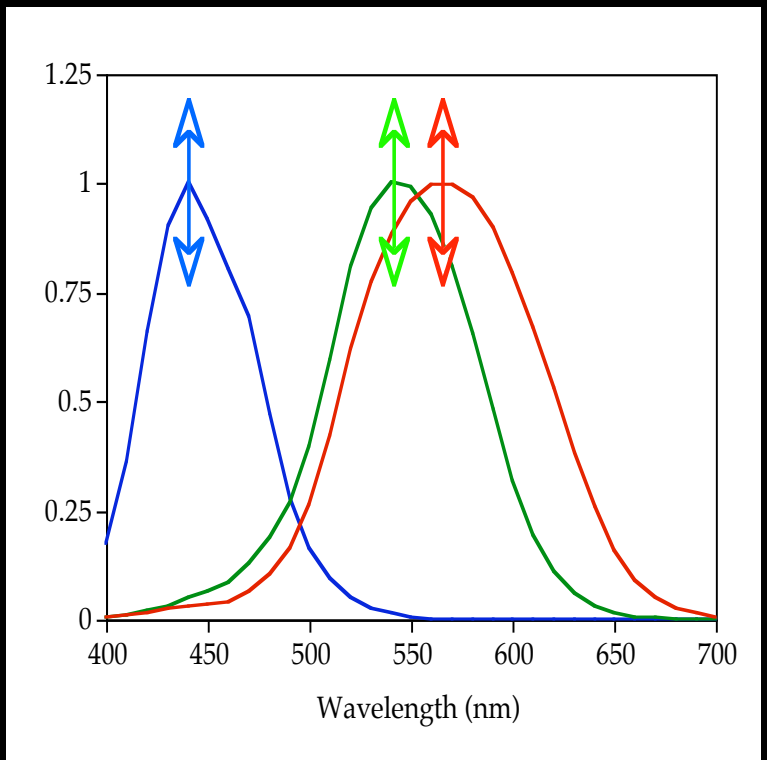
Color Appearance Models

- A Color Appearance Model provides mathematical formulae to transform physical measurements of the stimulus and viewing environment into correlates of perceptual attributes of color (e.g., lightness, chroma, hue, etc.).

Minimal Pieces

- Chromatic Adaptation Transform
- Uniform Color Space

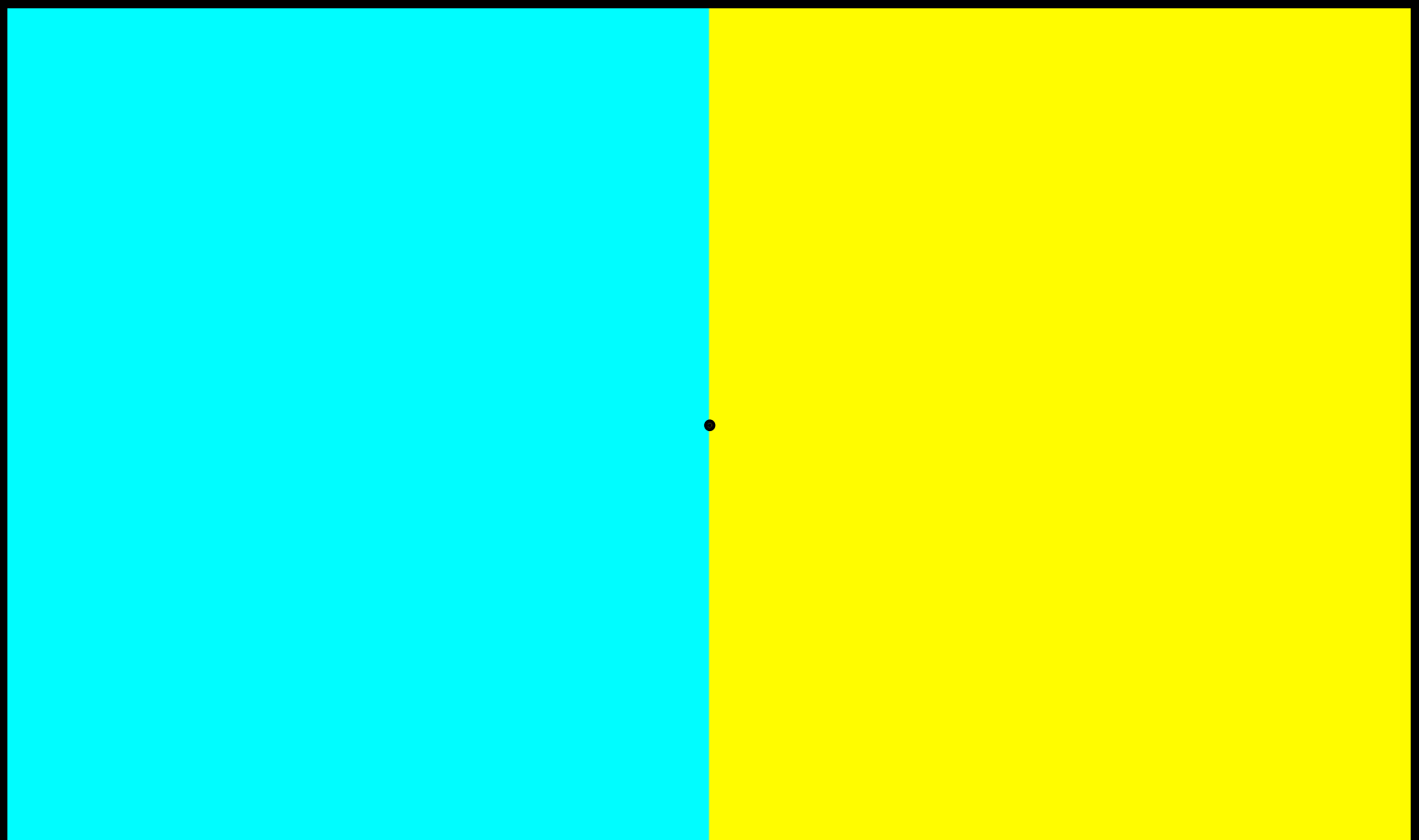
Chromatic Adaptation



The three cone types, LMS, are capable of independent sensitivity regulation. (Adaptation occurs in higher-level mechanisms as well.)

Magnitudes of chromatic responses are dependent on the state of adaptation (local, spatial, temporal). Afterimages provide evidence.

Chromatic Adaptation



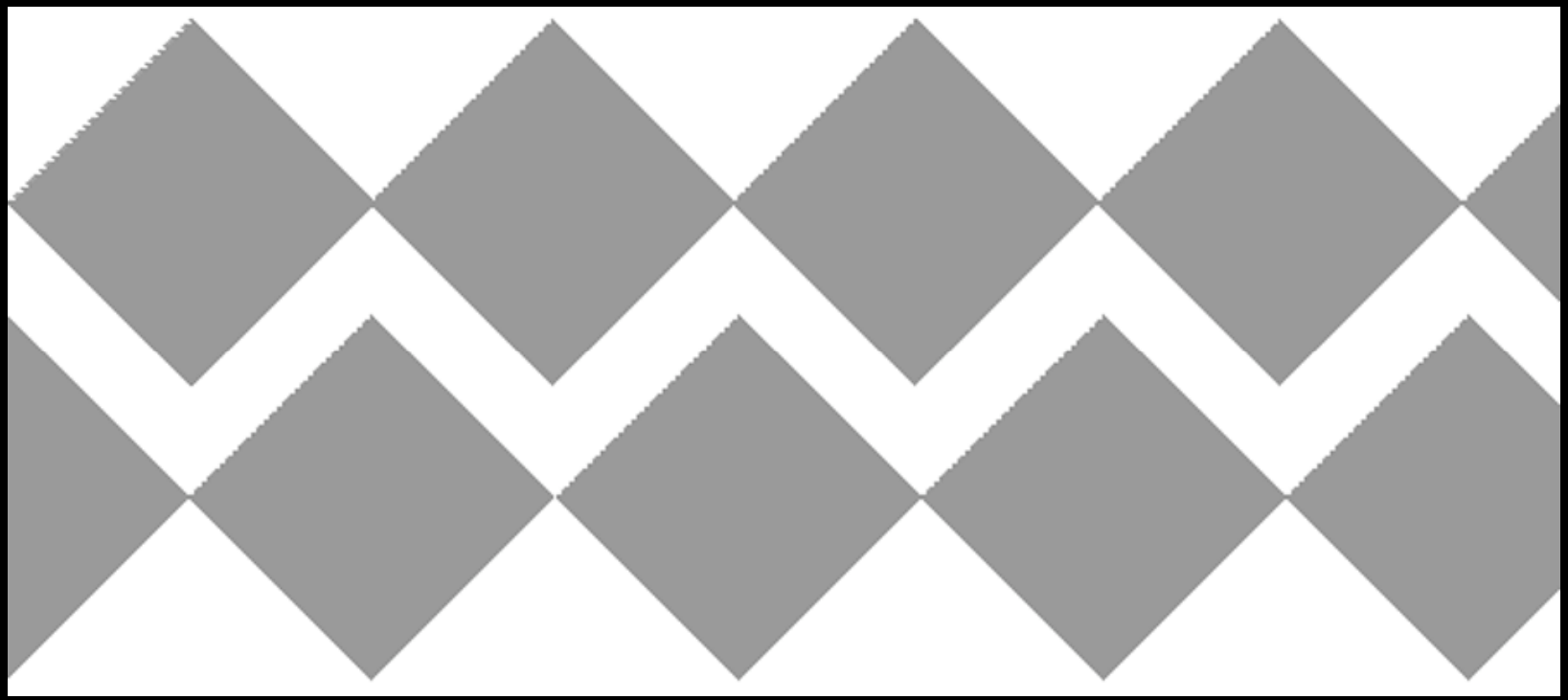
Wednesday, 27 May 2009

Chromatic Adaptation Model Output

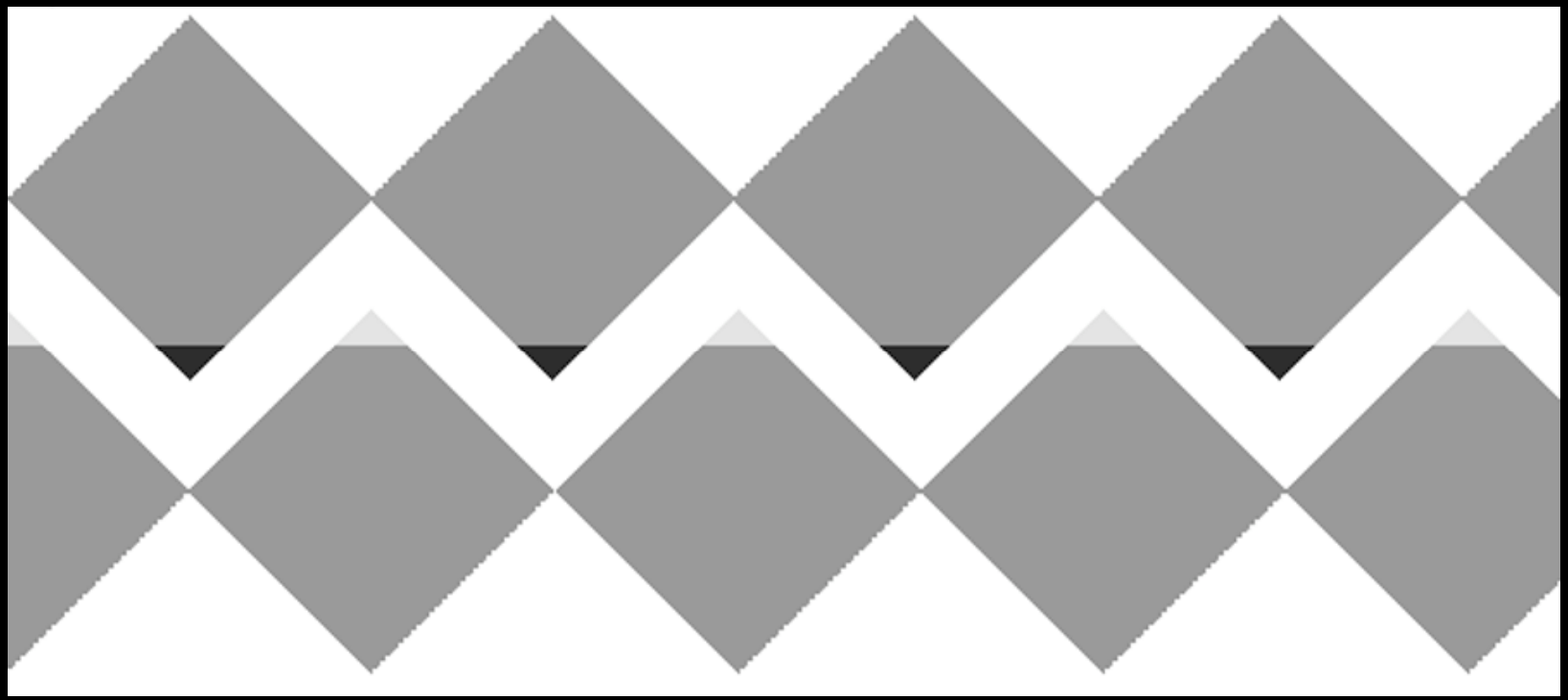
Raw "Radiance" Images

Adapted "Perceptual" Images

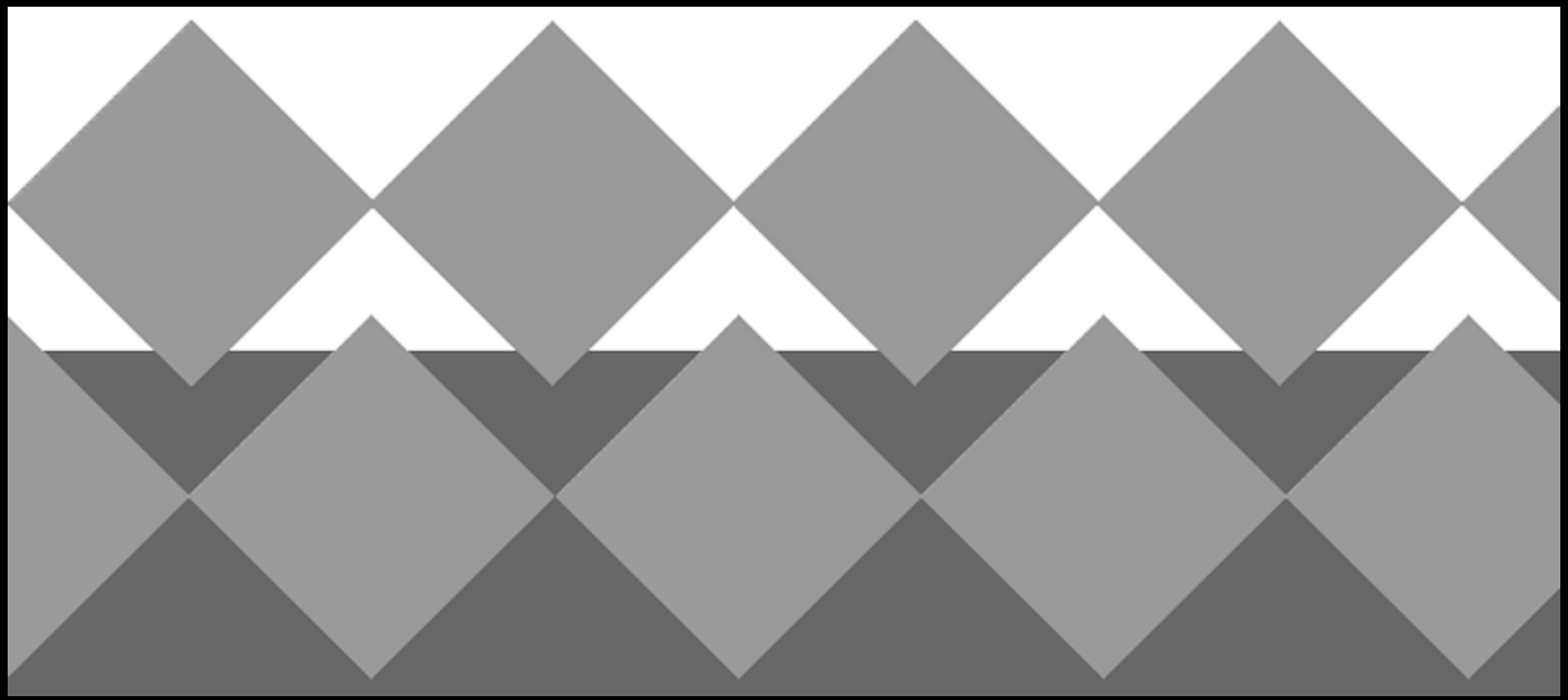
Diamonds



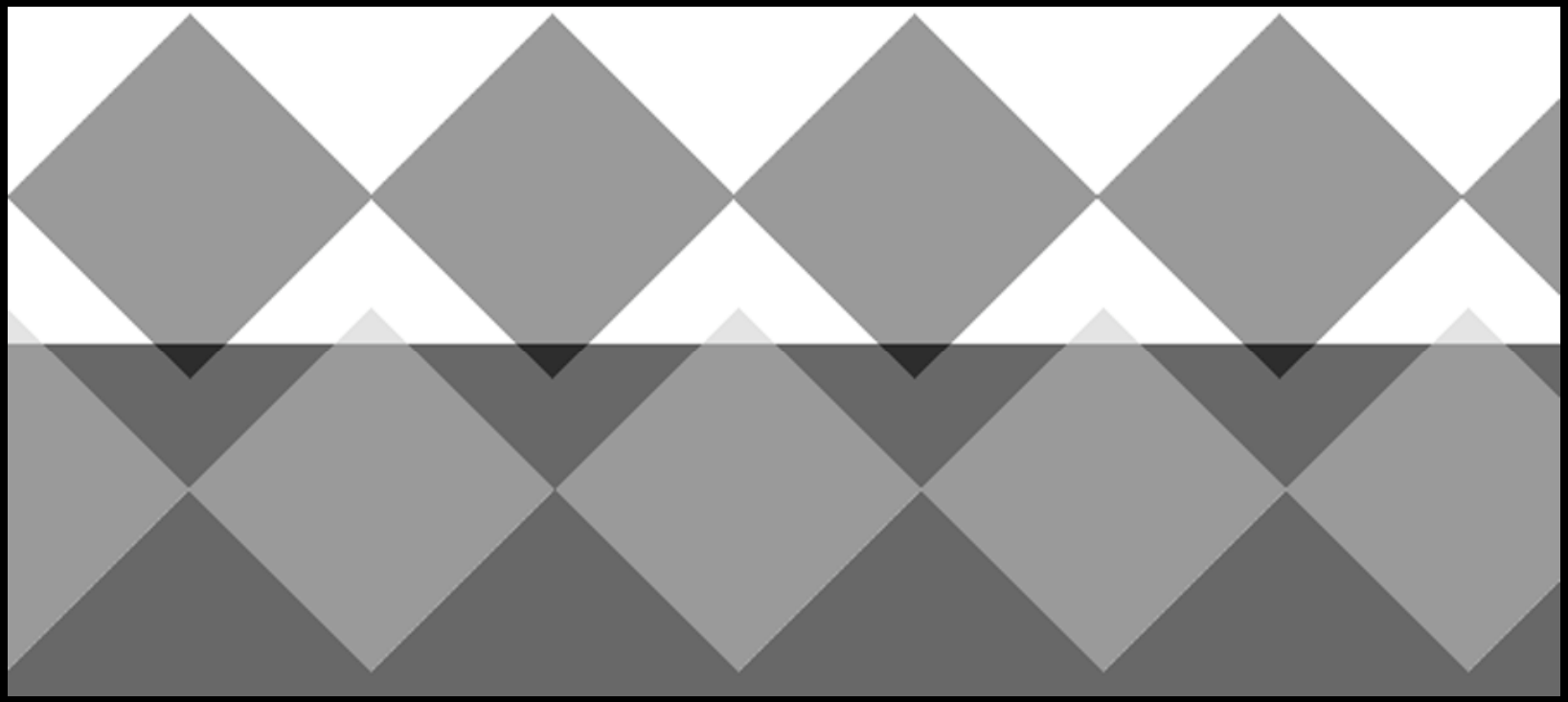
With Tips



On Backgrounds



Both Tips & Backgrounds!



Real-World Discounting

CIELAB (CIElab, CIEL*a*b*)

CIELAB Does:

- Model Chromatic Adaptation
- Model Response Compression
- Include Correlates for Lightness, Chroma, Hue
- Include Useful Color Difference Measure

CIELAB Doesn't:

- Predict Luminance Dependent Effects
- Predict Background or Surround Effects
- Have an Accurate Adaptation Transform

CIELAB

Chromatic Adaptation

$X/X_n, Y/Y_n, Z/Z_n$

Opponent Processes

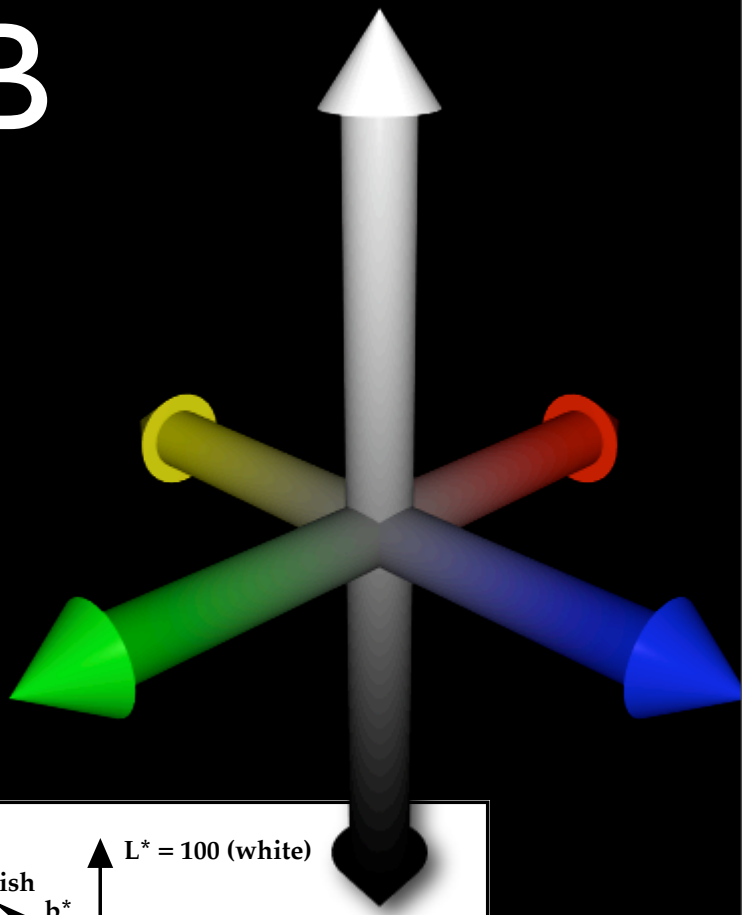
X-Y

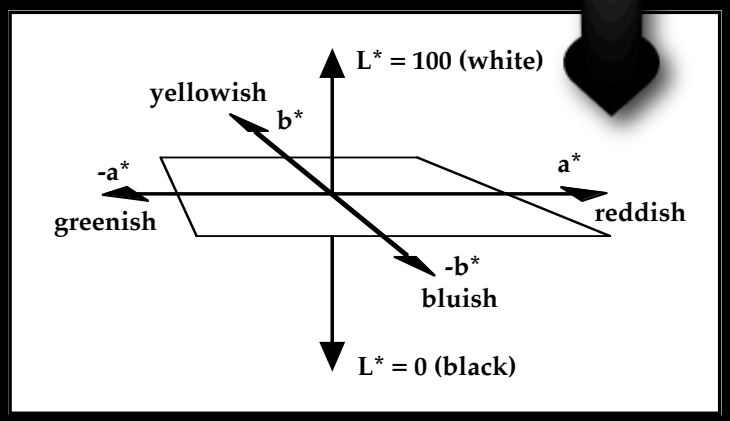
Y-Z

Uniform Spacing

Constants 116, 500, 200

Cube Root





CIELAB Equations

$$L^* = 116 f(Y / Y_n) - 16$$

$$a^* = 500\left[f(X / X_n) - f(Y / Y_n) \right]$$

$$b^* = 200\left[f(Y / Y_n) - f(Z / Z_n) \right]$$

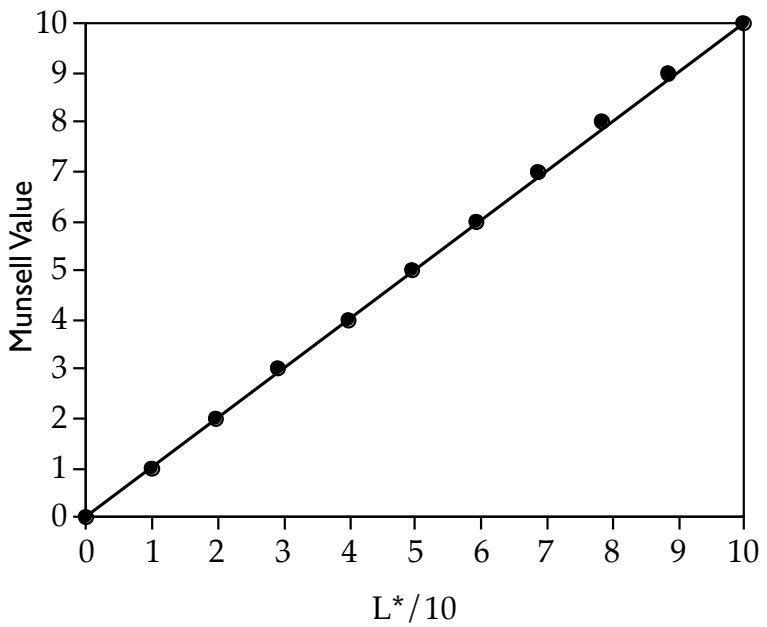
$$f(\omega) = (\omega)^{1/3} \qquad \omega > 0.008856$$

$$f(\omega) = 7.787(\omega) + 16/116 \qquad \omega \le 0.008856$$

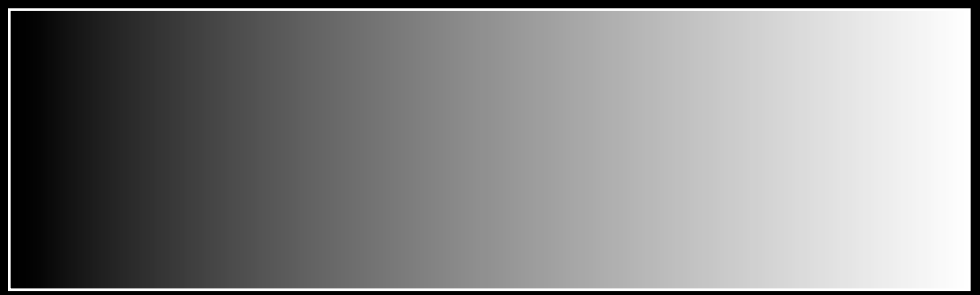
CIELAB Lightness

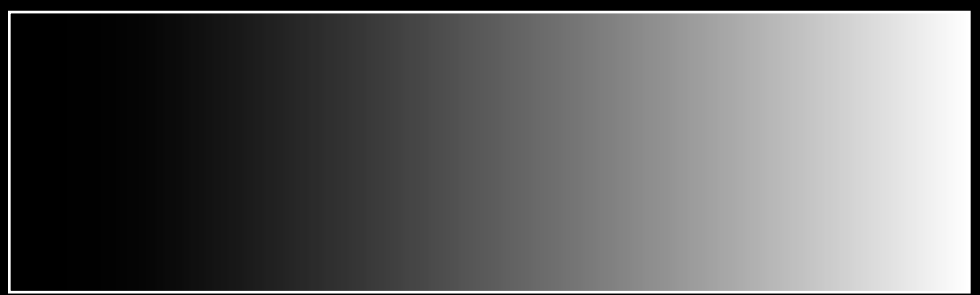
$$f(\omega) = (\omega)^{1/3} \qquad \omega > 0.008856$$

$$f(\omega) = 7.787(\omega) + 16/116 \qquad \omega \leq 0.008856$$



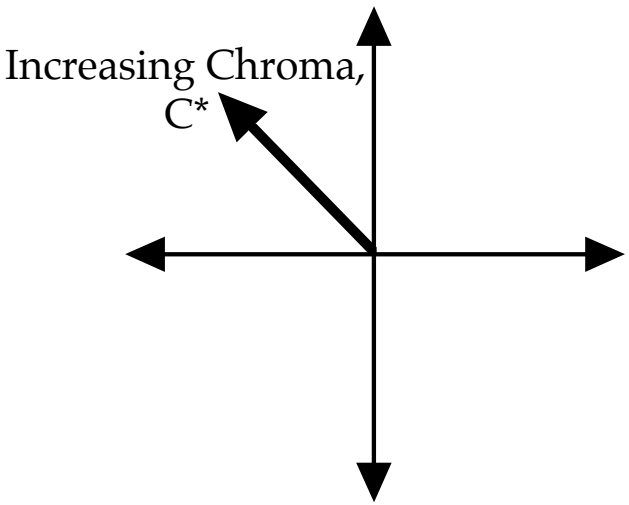
Lightness v. Luminance





CIELAB Chroma

$$C^* = \left(a^{*2} + b^{*2}\right)^{1/2}$$



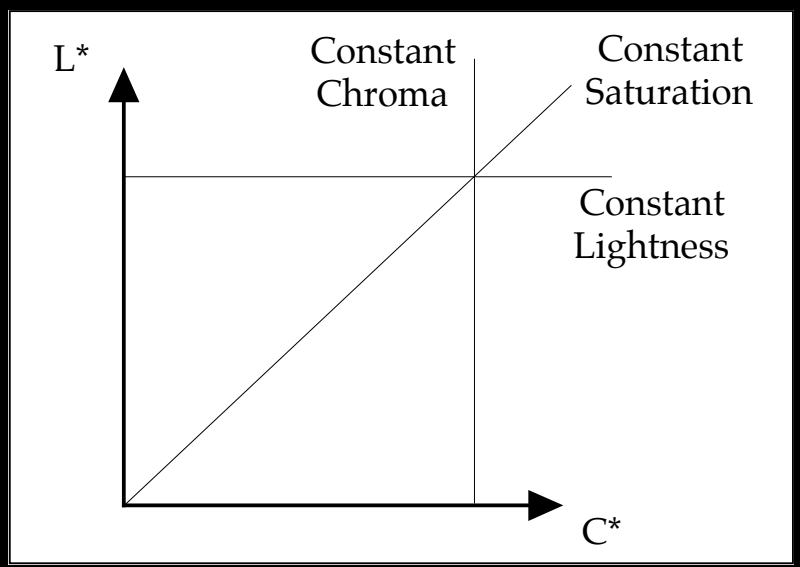
Neutrals Have Zero Chroma, C* = 0.0

Saturation in CIELAB

Due to the lack of a related chromaticity diagram, saturation is not officially defined in CIELAB.

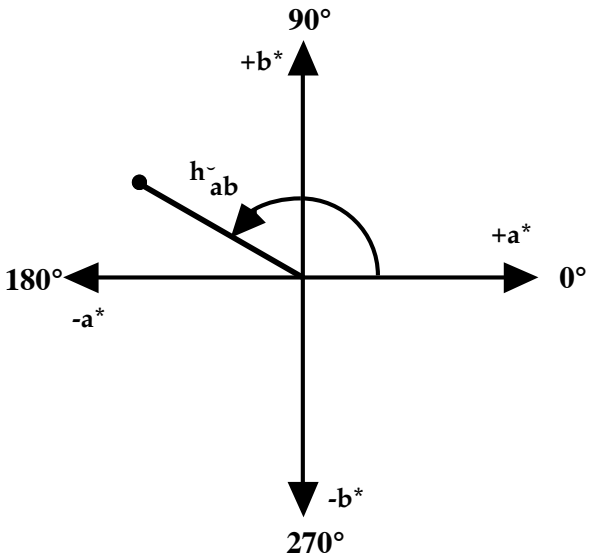
However recalling the definitions of chroma (colorfulness/brightness of white), lightness (brightness/brightness of white), and saturation (colorfulness/brightness).

Saturation = C*/L*



CIELAB Hue

$$h_{ab} = \tan^{-1}\left(\frac{b^*}{a^*}\right)$$

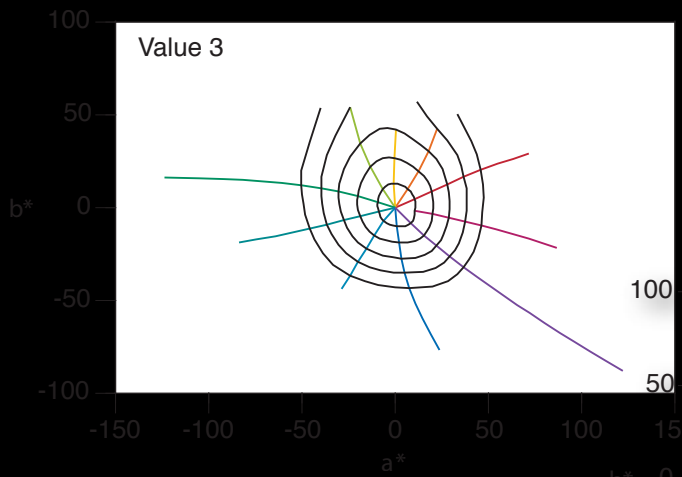


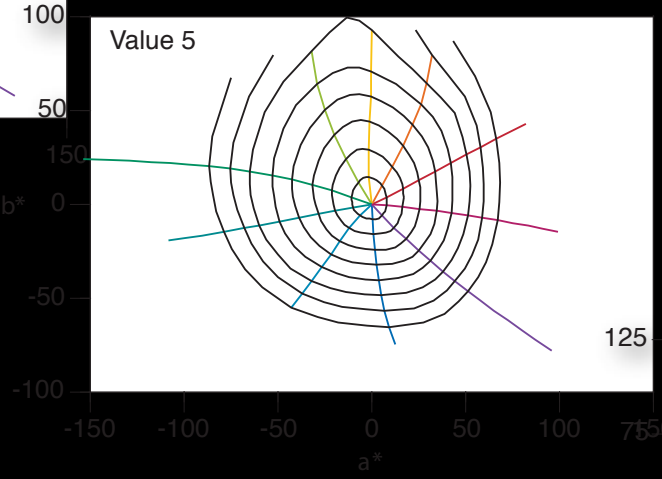
Relative Hue Scale — Where's Red?

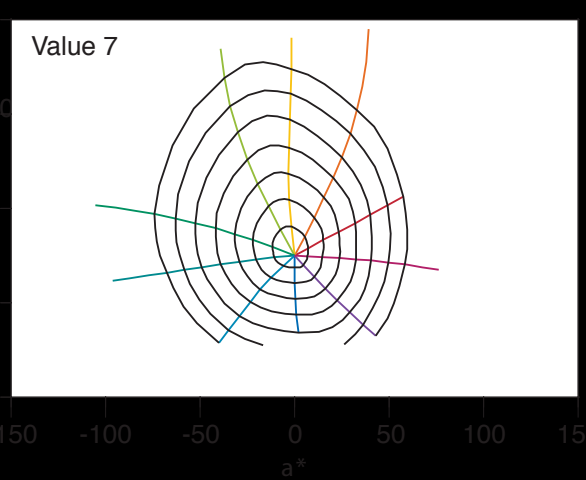
Approximate Hue Angles of NCS Unique Hues

R — 24°
Y — 90°
G — 162°
B — 246°

CIELAB Performance







Color Difference Equations

$$\Delta E^*_{ab} = \sqrt{\Delta L^* + \Delta a^* + \Delta b^*}$$

$$\Delta E^*_{ab} = \sqrt{\Delta L^* + \Delta C^* + \Delta H^*}$$

$$\Delta E^*_{94} = \sqrt{\frac{\Delta L^*}{k_L} + \frac{\Delta C^*}{k_C\left(1 + 0.045C^*\right)} + \frac{\Delta H^*}{k_H\left(1 + 0.015C^*\right)}}$$

Why Not Just CIELAB?

Positive Aspects:

- Accounts for Chromatic Adaptation
- Works Well for Near-Daylight Illuminants
  (also Medium Gray Background & Surround and Moderate Luminance Levels)
-

Negative Aspects:

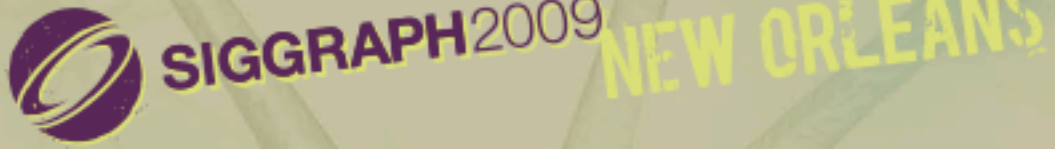
- Does Not Account for Changes in:
  - Background
  - Surround
  - Luminance
  - Cognition
- Cannot Predict Brightness & Colorfulness
- "Wrong" von Kries Transform Works Poorly for Large Changes From Daylight
- Constant-Hue Predictions could be Improved
  (especially Blue)

CIELAB Makes a Good, Simple Baseline to start

Beyond CIELAB

- More Accurate Adaptation Transform
- Luminance Dependencies
- Surround Dependencies
- Brightness & Colorfulness
- Hue Linearity

(Hunt, Nayatani, RLAB, LLAB, CIECAM97s, CIECAM02)



Color Application to Rendering

Greg Ward, Dolby Canada

Picture Perfect *RGB* Rendering Using Spectral Prefiltering and Sharp Color Primaries

Greg Ward

(Exponent - Failure Analysis Assoc.)

Elena Eydelberg-Vileshin

(Stanford University)

Presented at the 13th Eurographics Workshop on Rendering in Pisa, Italy, June 2002

Talk Overview

1. Color Rendering Techniques
2. Getting the Most Out of *RGB*
   a) Spectral prefiltering
   b) The von Kries white point transform
3. Three Tristimulus Spaces
4. Experimental Results
5. Conclusions

1. A Brief Comparison of Color Rendering Techniques

- Spectral Rendering
  - ✓N spectrally pure samples
- Component Rendering
  - ✓M vector basis functions
- *RGB* (Tristimulus) Rendering
  - ✓Tristimulus value calculations

Spectral Rendering

1. Divide visible spectrum into N wavelength samples
2. Process spectral samples separately throughout rendering calculation
3. Compute final display color using CIE color matching functions and standard transformations

Component Rendering

[Peercy, Siggraph '93]

1. Divide visible spectrum into M vector bases using component analysis
2. Process colors using MxM matrix multiplication at each interaction
3. Compute final display color with 3xM matrix transform

RGB (Tristimulus) Rendering

1. Precompute tristimulus values
2. Process 3 samples separately throughout rendering calculation
3. Compute final display color with 3x3 matrix transform (if necessary)

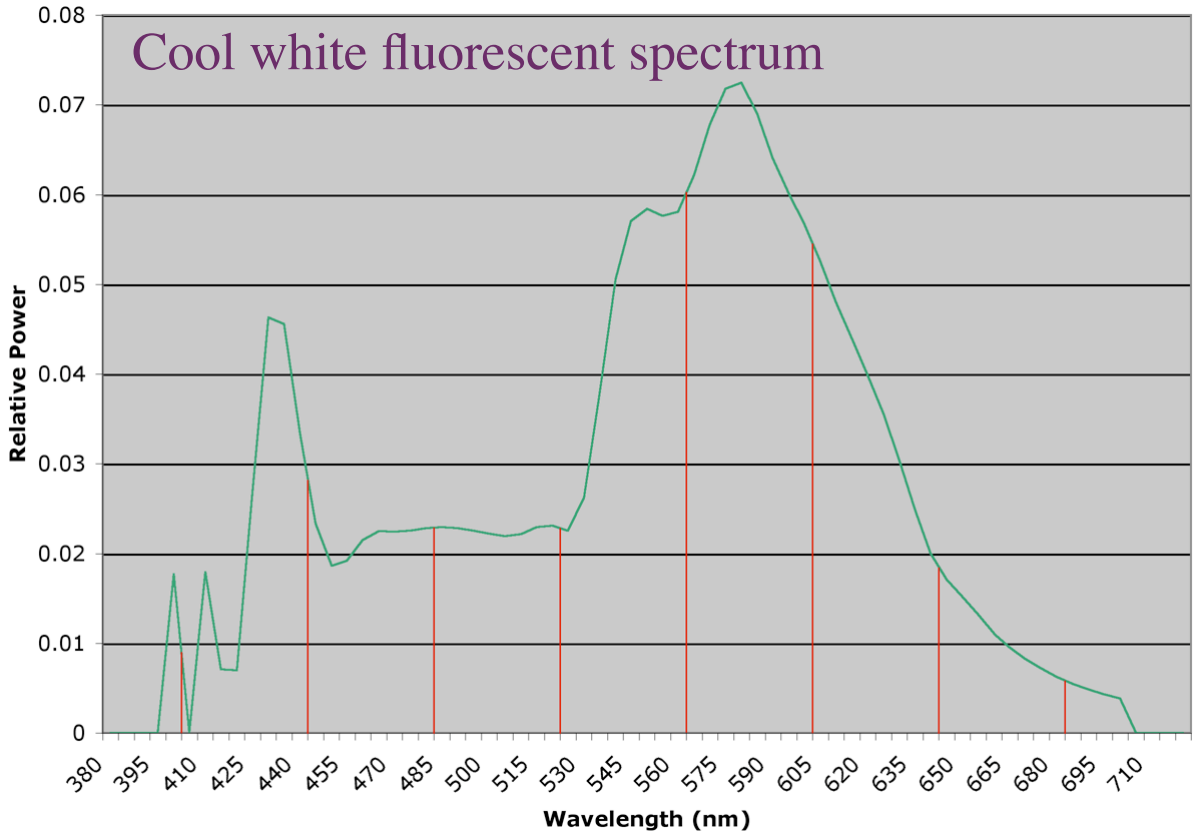
Rendering Cost Comparison

	Pre-processing	Multiplies / Interaction	Post-processing
Spectral	None	N ($N \geq 9$)	N multiplies per pixel
Component	Vector analysis	MxM ($M \geq 3$)	3×M per pixel
RGB	Little or none	3	0 to 9 per pixel

Strengths and Weaknesses

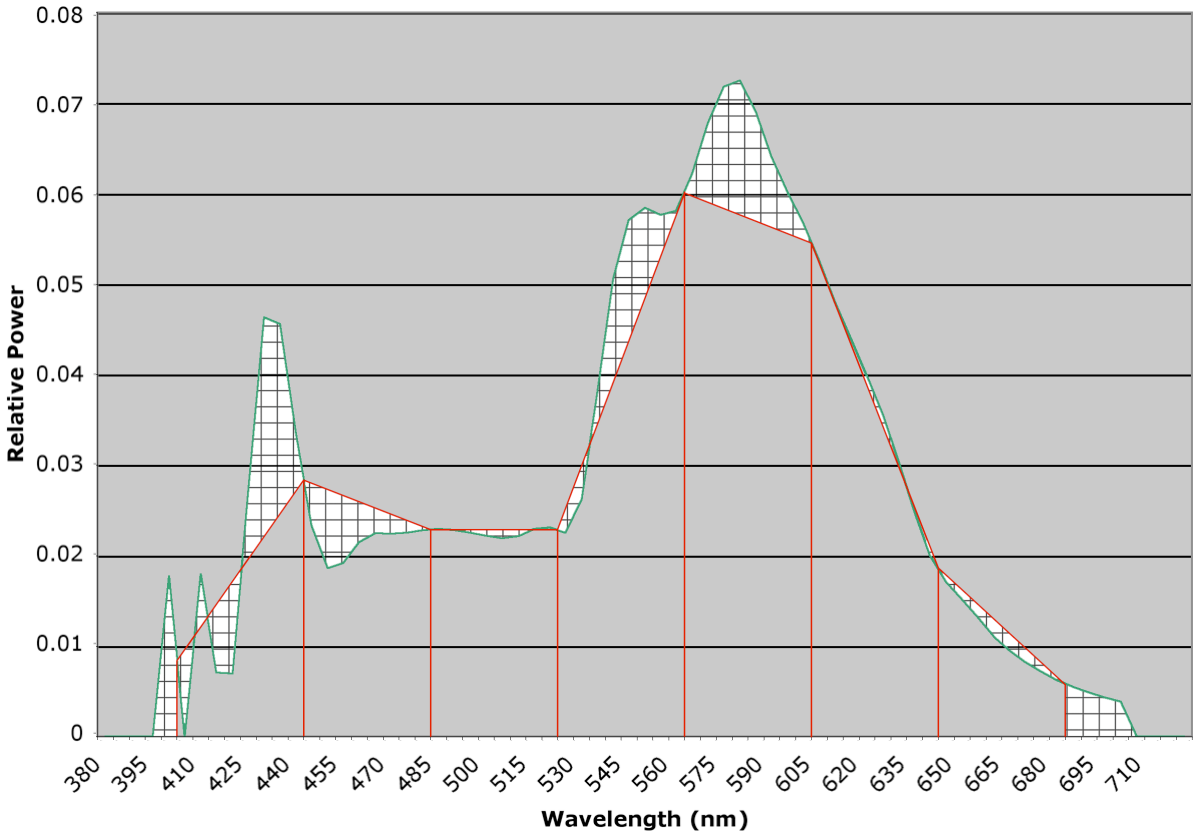
	Strengths	Weaknesses
Spectral	Potential accuracy	Cost, aliasing, data mixing
Component	Optimizes cost/ benefit	Preprocessing requirements
RGB	Fast, widely supported	Limited accuracy

Spectral Aliasing



[Meyer88] suffers worse with only 4 samples

Spectral Aliasing



[Meyer88] suffers worse with only 4 samples

The Data Mixing Problem

- Typical situation:
  - Illuminants known to 5 nm resolution
  - Some reflectances known to 10 nm
  - Other reflectances given as tristimulus
- Two alternatives:
  A. Reduce all spectra to lowest resolution
  B. Interpolate/synthesize spectra [Smits '99]

2. Getting the Most Out of *RGB*

A. How Does *RGB* Rendering Work and When Does It Not?
B. Can *RGB* Accuracy Be Improved?
C. Useful Observations
D. Spectral Prefiltering
E. The von Kries White Point Transform

Status Quo Rendering

- White Light Sources
  - E.g., (R,G,B)=(1,1,1)
- *RGB* material colors obtained by dubious means
  - E.g., “That looks pretty good.”
    - ✓This actually works for fictional scenes!
- Color correction with ICC profile if at all

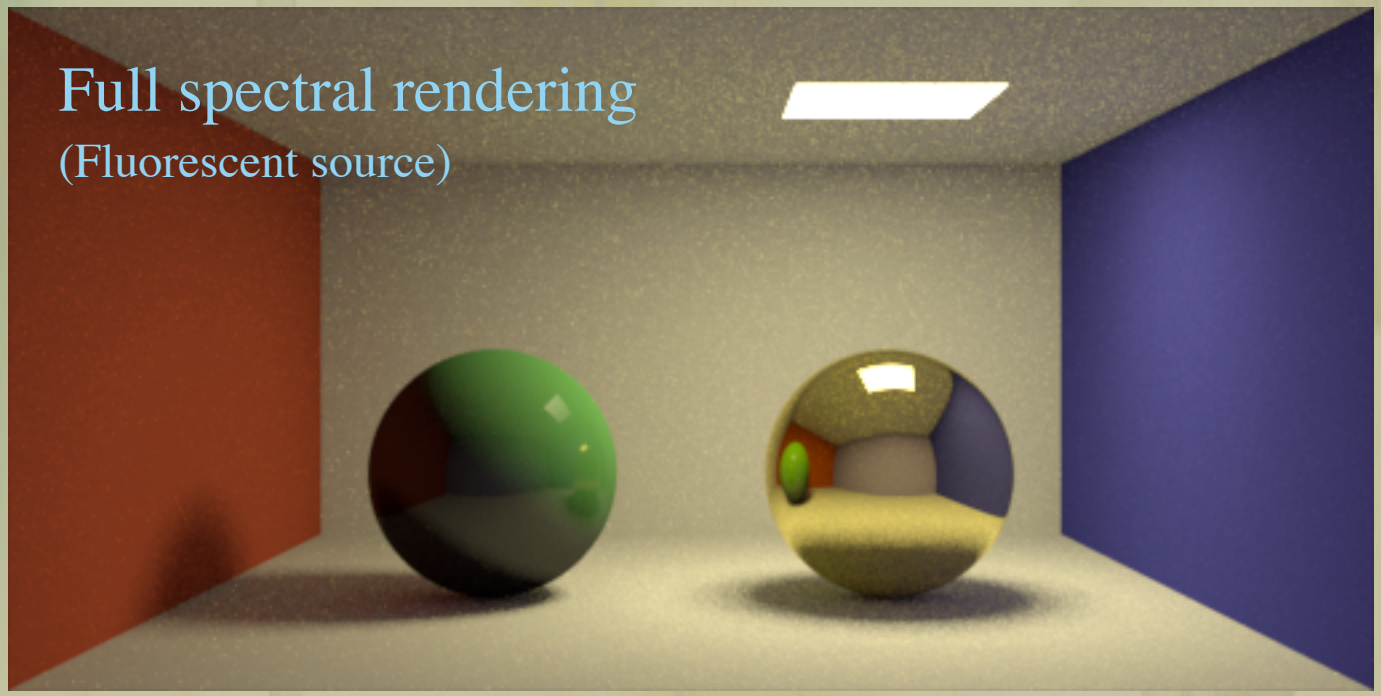
When Does RGB Rendering Normally Fail?

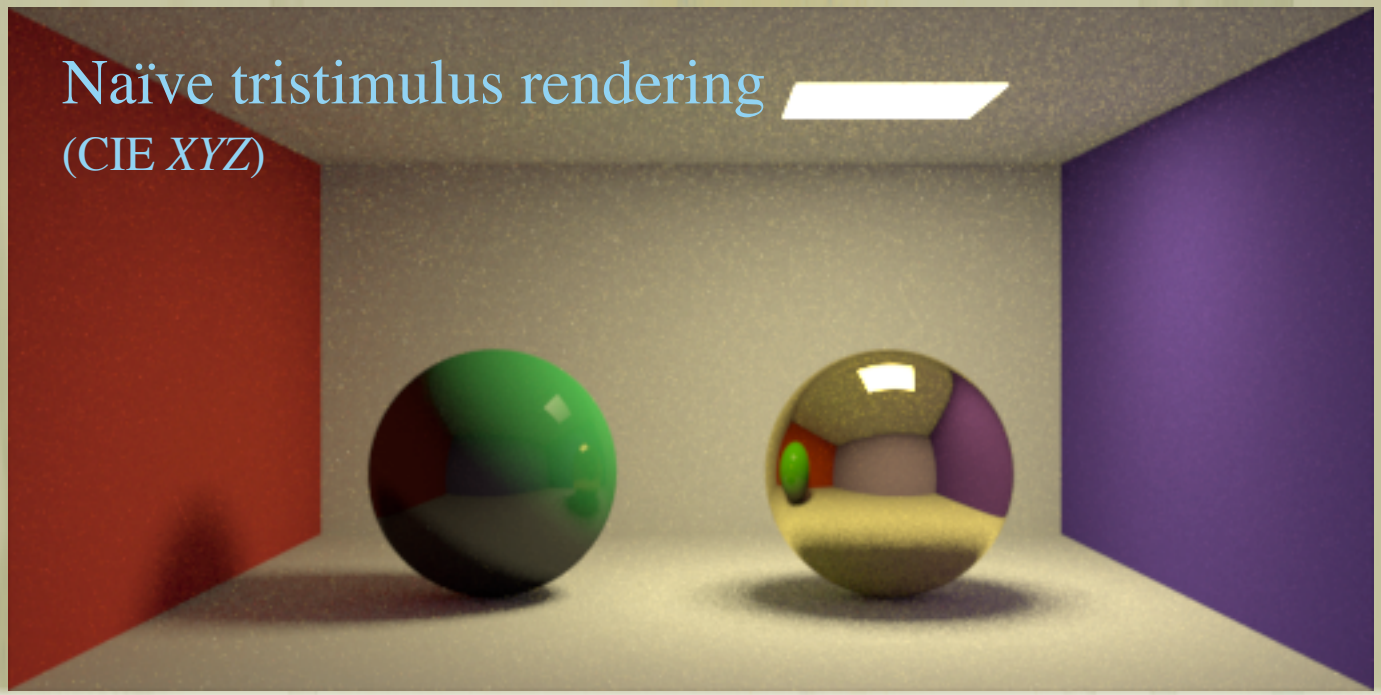
- When you start with measured colors
- When you want to simulate color appearance under another illuminant
- When your illuminant and surface spectra have sharp peaks and valleys

When Does RGB Rendering Normally Fail?

- When you start with measured colors
- When you want to simulate color appearance under another illuminant
- When your illuminant and surface spectra have sharp peaks and valleys

The Result: Wrong COLORS!


Full spectral rendering
(Fluorescent source)


Naïve tristimulus rendering
(CIE XYZ)

Can *RGB* Accuracy Be Improved?

- Identify and minimize sources of error
  - Source-surface interactions
  - Choice of rendering primaries
- Overcome ignorance and inertia
  - Many people render in *RGB* without really understanding what it means
  - White-balance problem scares casual users away from colored illuminants

A Few Useful Observations

a) Direct illumination is the first order in any rendering calculation
b) Most scenes contain a single, dominant illuminant spectrum
c) Scenes with mixed illuminants will have a color cast regardless

A Few Useful Observations

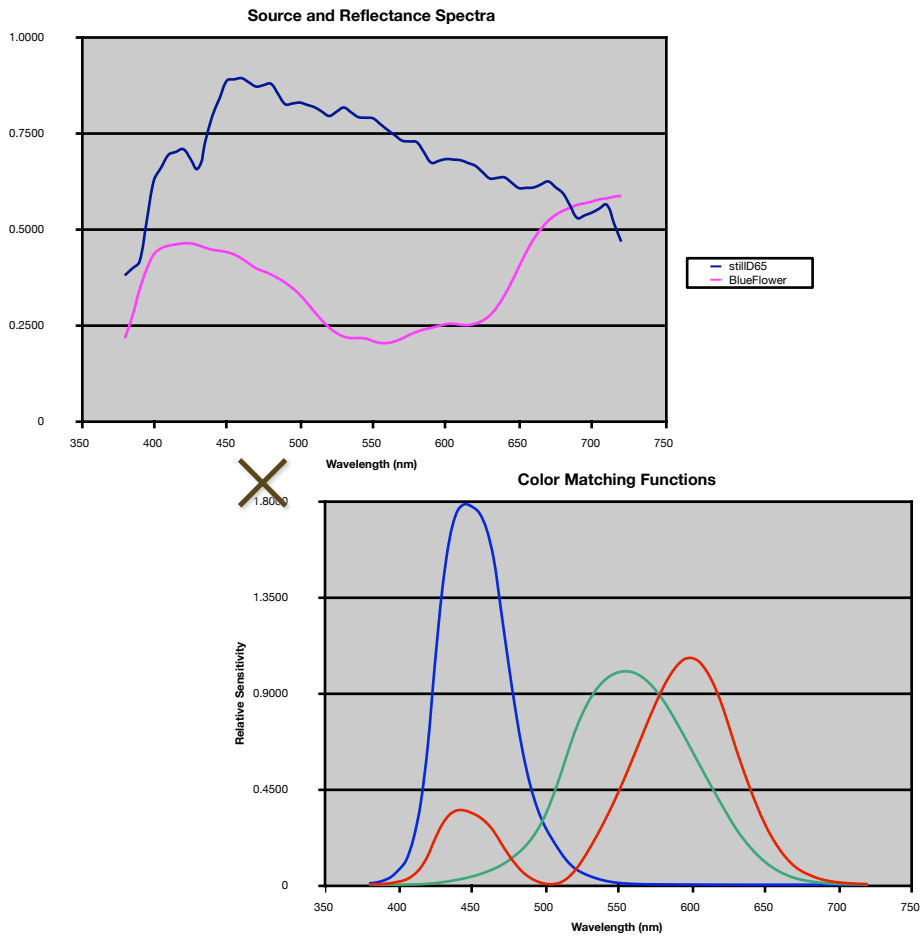
a) Direct illumination is the first order in any rendering calculation
b) Most scenes contain a single, dominant illuminant spectrum
c) Scenes with mixed illuminants will have a color cast regardless

Conclusion: Optimize for the Direct→Diffuse Case

Picture Perfect *RGB* Rendering

1. Identify dominant illuminant spectrum
   a) Prefilter material spectra to obtain tristimulus colors for rendering
   b) Adjust source colors appropriately
2. Perform tristimulus (*RGB*) rendering
3. Apply white balance transform and convert pixels to display color space

Spectral Prefiltering



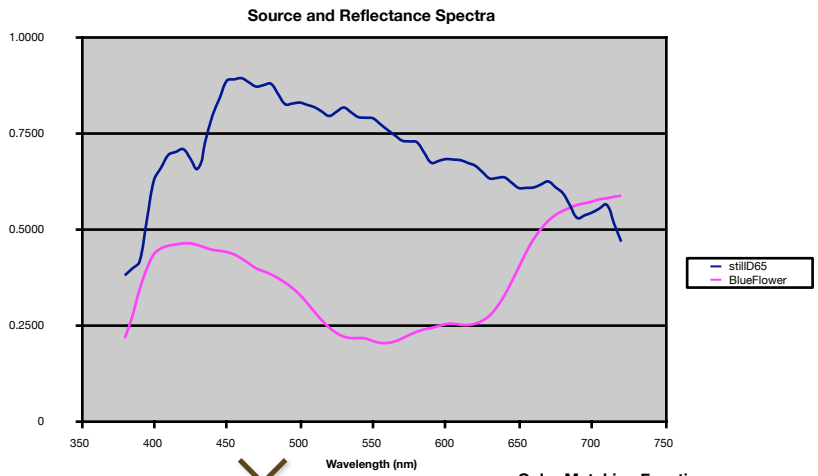
To obtain a tristimulus color, you *must* know the illuminant spectrum

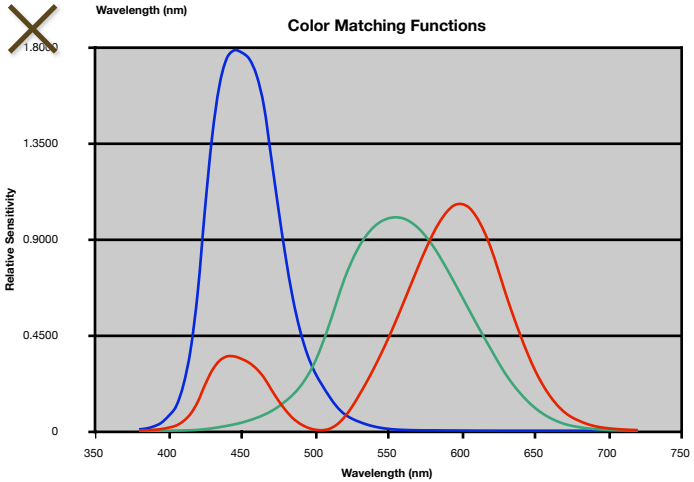
$$X = \int I(\lambda)\,\rho(\lambda)\,\bar{x}(\lambda)\,d\lambda$$

$$Y = \int I(\lambda)\,\rho(\lambda)\,\bar{y}(\lambda)\,d\lambda$$

$$Z = \int I(\lambda)\,\rho(\lambda)\,\bar{z}(\lambda)\,d\lambda$$

Spectral Prefiltering





To obtain a tristimulus color, you *must* know the illuminant spectrum

$$X = \int I(\lambda)\,\rho(\lambda)\,\bar{x}(\lambda)\,d\lambda$$

$$Y = \int I(\lambda)\,\rho(\lambda)\,\bar{y}(\lambda)\,d\lambda$$

$$Z = \int I(\lambda)\,\rho(\lambda)\,\bar{z}(\lambda)\,d\lambda$$

XYZ may then be transformed by 3×3 matrix to any linear tristimulus space (e.g., *sRGB*)

Prefiltering vs. Full Spectral Rendering

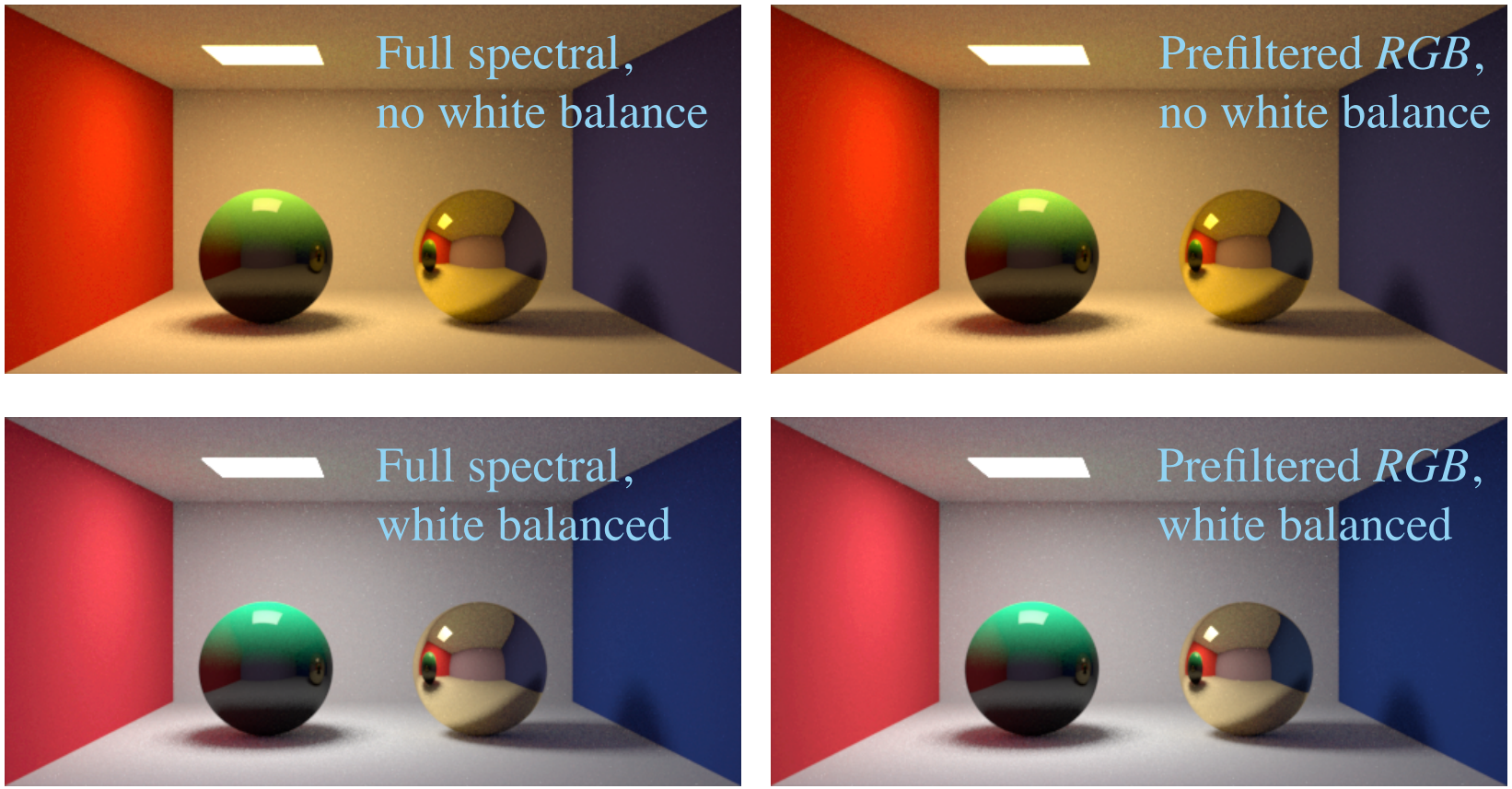
+ Prefiltering performed once per material vs. every rendering interaction
+ Spectral aliasing and data mixing problems disappear with prefiltering
- However, mixed illuminants and interreflections not computed exactly

Prefiltering vs. Full Spectral Rendering

+ Prefiltering performed once per material vs. every rendering interaction
+ Spectral aliasing and data mixing problems disappear with prefiltering
- However, mixed illuminants and interreflections not computed exactly

Regardless which technique you use, remember to apply white balance to result!

Quick Comparison



The von Kries Transform for Chromatic Adaptation

The von Kries transform takes colors from absolute *XYZ* to adapted equiv. *XYZ'*

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \mathbf{M}_C^{-1} \begin{bmatrix} \frac{R_w'}{R_w} & 0 & 0 \\ 0 & \frac{G_w'}{G_w} & 0 \\ 0 & 0 & \frac{B_w'}{B_w} \end{bmatrix} \mathbf{M}_C \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

The von Kries Transform for Chromatic Adaptation

The von Kries Transform for Chromatic Adaptation

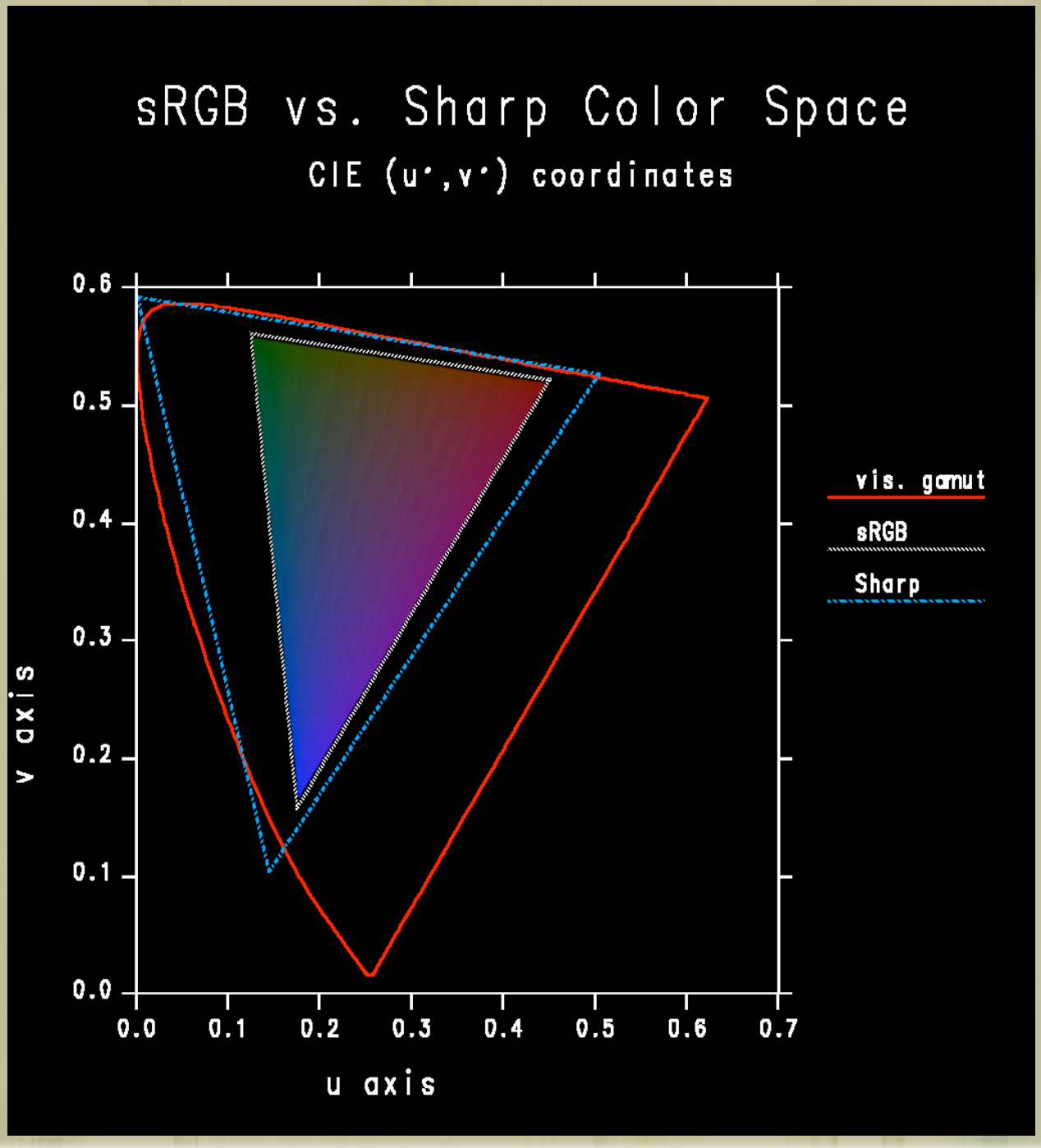
Where:

$$\begin{bmatrix} R_w' \\ G_w' \\ B_w' \end{bmatrix} = \mathbf{M}_c \begin{bmatrix} X_w' \\ Y_w' \\ Z_w' \end{bmatrix} \qquad \begin{bmatrix} R_w \\ G_w \\ B_w \end{bmatrix} = \mathbf{M}_c \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix}$$

Display white point Scene white point

Chromatic Adaptation Matrix

- The matrix $\mathbf{M}_C$ transforms *XYZ* into an "adaptation color space"
- Finding the optimal CAM is an under-constrained problem -- many candidates have been suggested
- "Sharper" color spaces tend to perform better for white balance transforms
  - See [Finlayson & Susstrunk, CIC '00]


sRGB vs. Sharp Color Space
CIE (u',v') coordinates
0.6
0.5
0.4
0.3
0.2
0.1
0.0
v axis
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
u axis
vis. gamut
sRGB
Sharp

3. Three Tristimulus Spaces for Color Rendering

- CIE *XYZ*
  - Covers visible gamut with positive values
  - Well-tested standard for color-matching
- *sRGB*
  - Common standard for image encoding
  - Matches typical CRT display primaries
- *Sharp RGB*
  - Developed for chromatic adaptation

XYZ Rendering Process

1. Apply prefiltering equation to get absolute *XYZ* colors for each material
   a) Divide materials by illuminant:

$$X_m^* = \frac{X_m}{X_w}, \quad Y_m^* = \frac{Y_m}{Y_w}, \quad Z_m^* = \frac{Z_m}{Z_w}$$

   b) Use absolute *XYZ* colors for sources
2. Render using tristimulus method
3. Finish w/ CAM and display conversion

sRGB Rendering Process

1. Perform prefiltering and von Kries transform on material colors
   a) Model dominant light sources as neutral
   b) For spectrally distinct light sources use:

$$R_s^* = \frac{R_s}{R_w}, \quad G_s^* = \frac{G_s}{G_w}, \quad B_s^* = \frac{B_s}{B_w}$$

2. Render using tristimulus method
3. Resultant image is *sRGB*

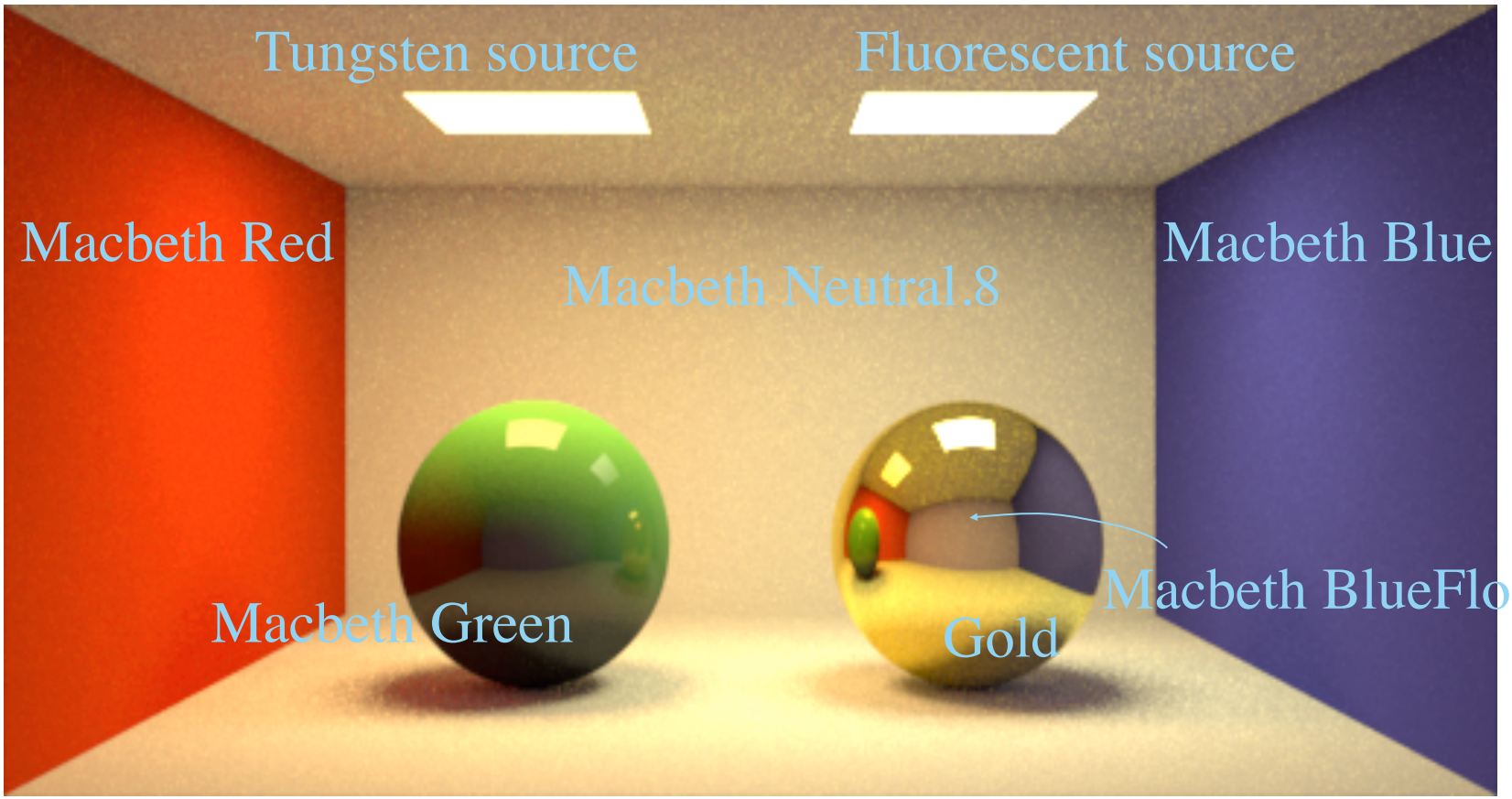
Sharp RGB Rendering Process

1. Prefilter material colors and apply von Kries transform to *Sharp RGB* space:

$$\begin{bmatrix} R_m^* \\ G_m^* \\ B_m^* \end{bmatrix} = \begin{bmatrix} \frac{1}{R_w} & 0 & 0 \\ 0 & \frac{1}{G_w} & 0 \\ 0 & 0 & \frac{1}{B_w} \end{bmatrix} \mathbf{M}_{Sharp} \begin{bmatrix} X_m \\ Y_m \\ Z_m \end{bmatrix}$$

2. Render using tristimulus method
3. Finish up CAM and convert to display

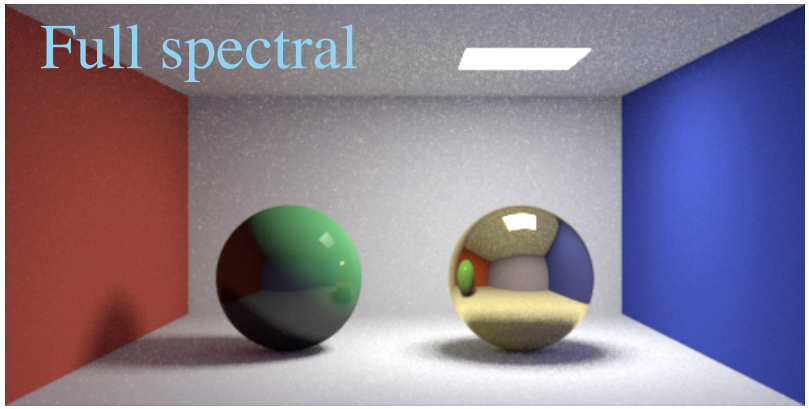
Our Experimental Test Scene

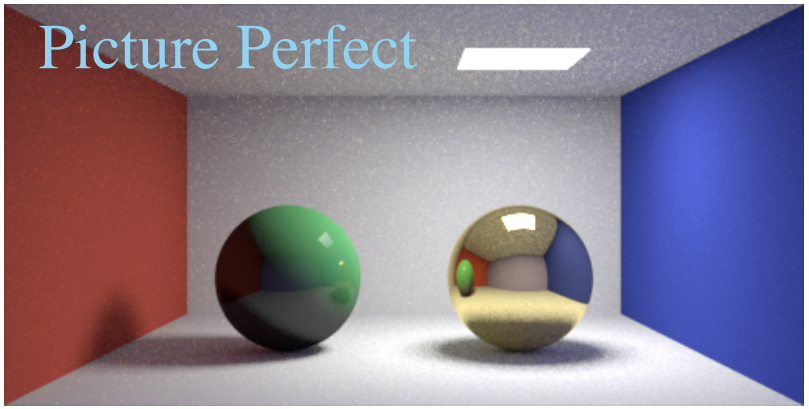


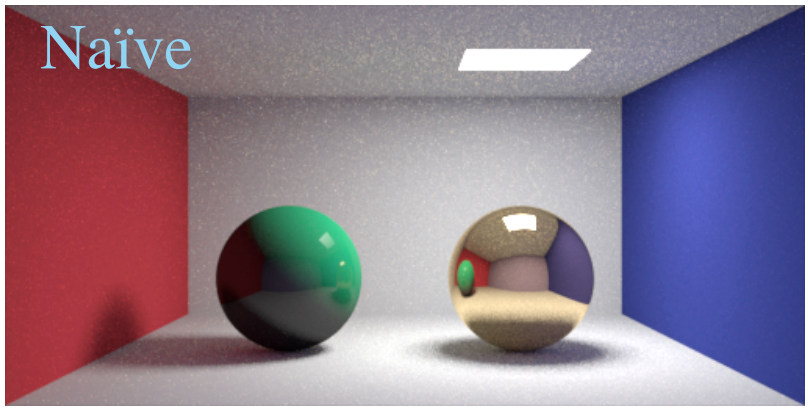
4. Experimental Results

- Three lighting conditions
  - Single 2856°K tungsten light source
  - Single cool white fluorescent light source
  - Both light sources (tungsten & fluorescent)
- Three rendering methods
  - Naïve *RGB* (assumes equal-energy white)
  - Picture Perfect *RGB*
  - Full spectral rendering (380 to 720 nm / 69 samp.)
- Three color spaces (*XYZ*, *sRGB*, *Sharp RGB*)

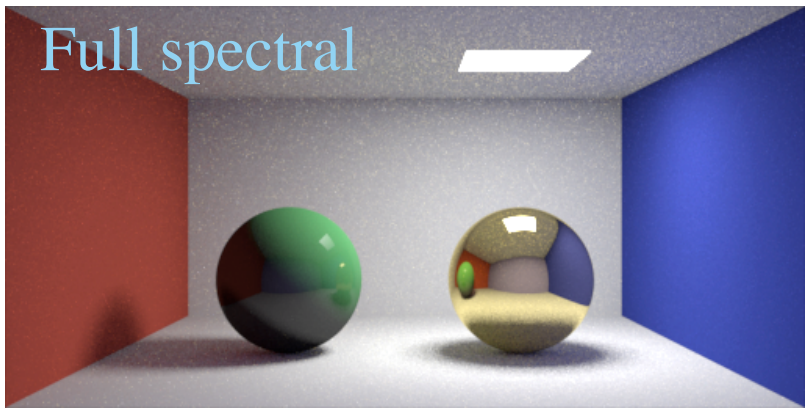
Example Comparison (*sRGB*)

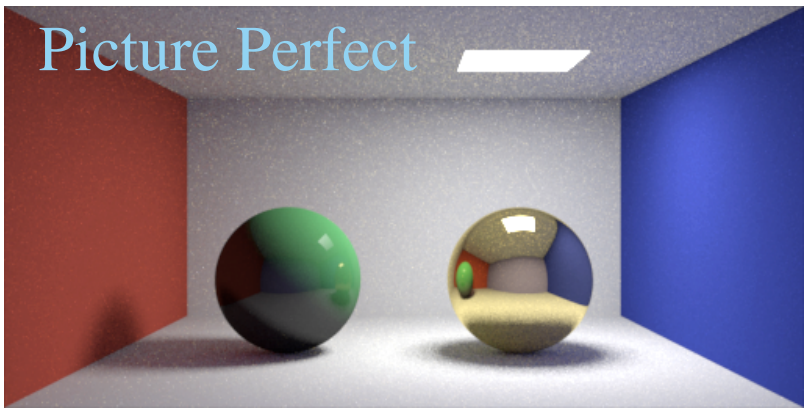


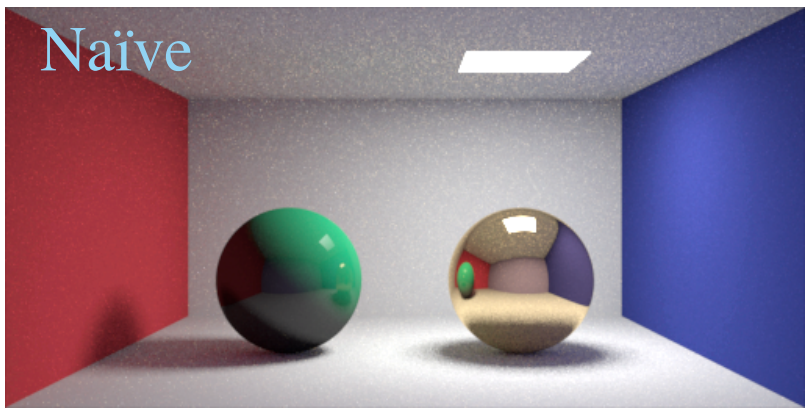


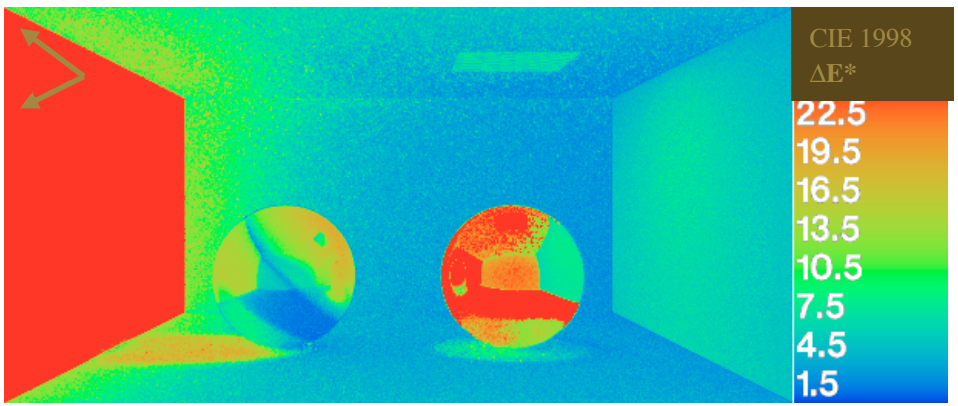


Example Comparison (*sRGB*)



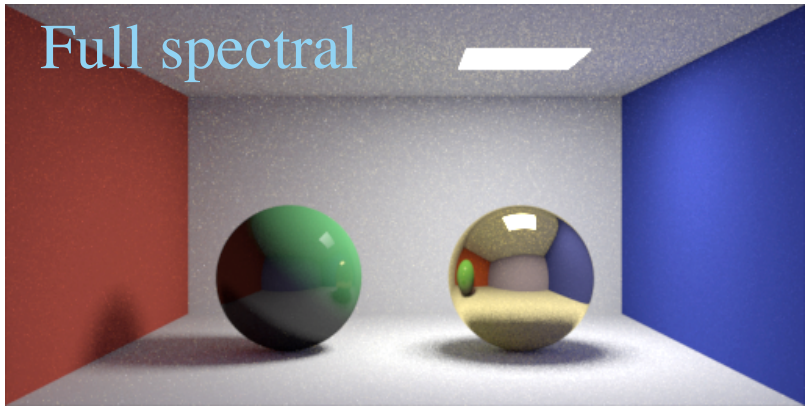


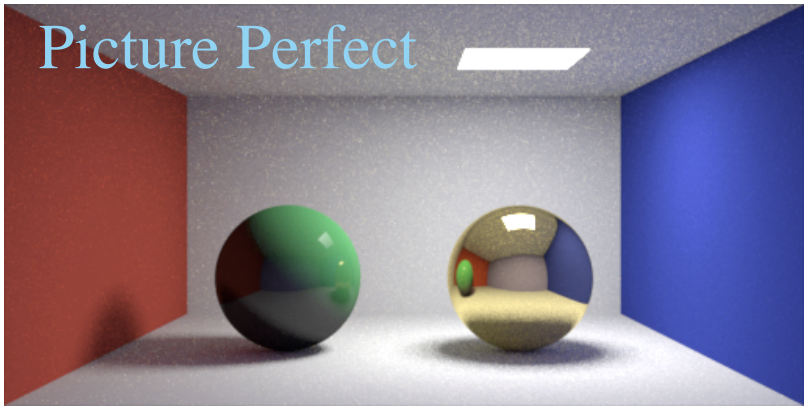


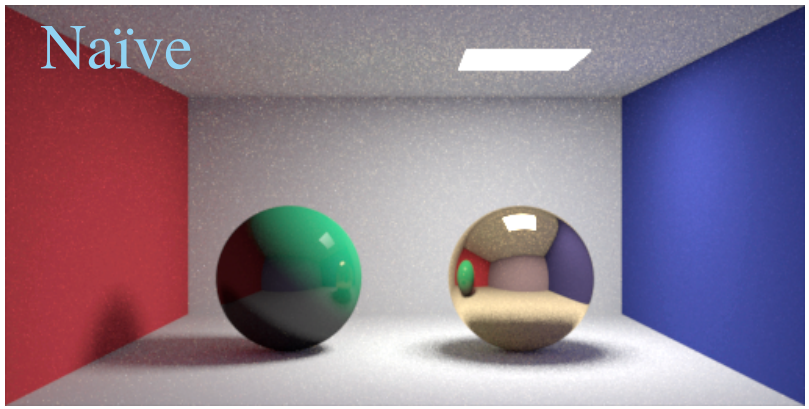


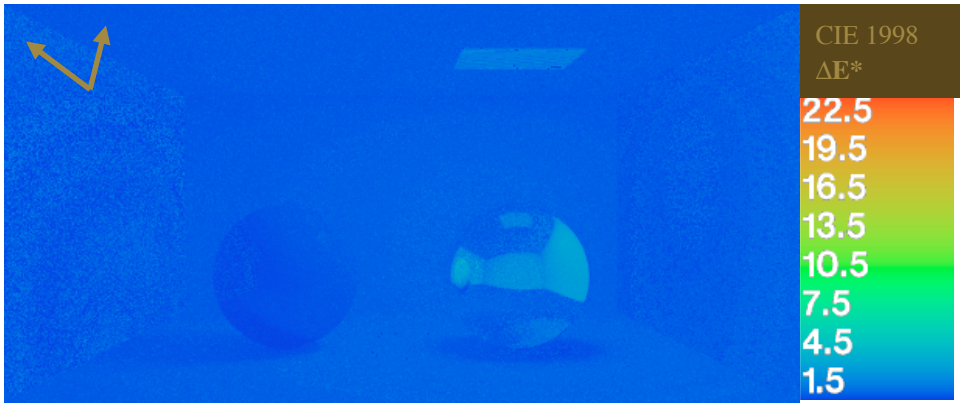
CIE 1998 ΔE* of 5 or above is visible in side-by-side comparisons

Example Comparison (*sRGB*)



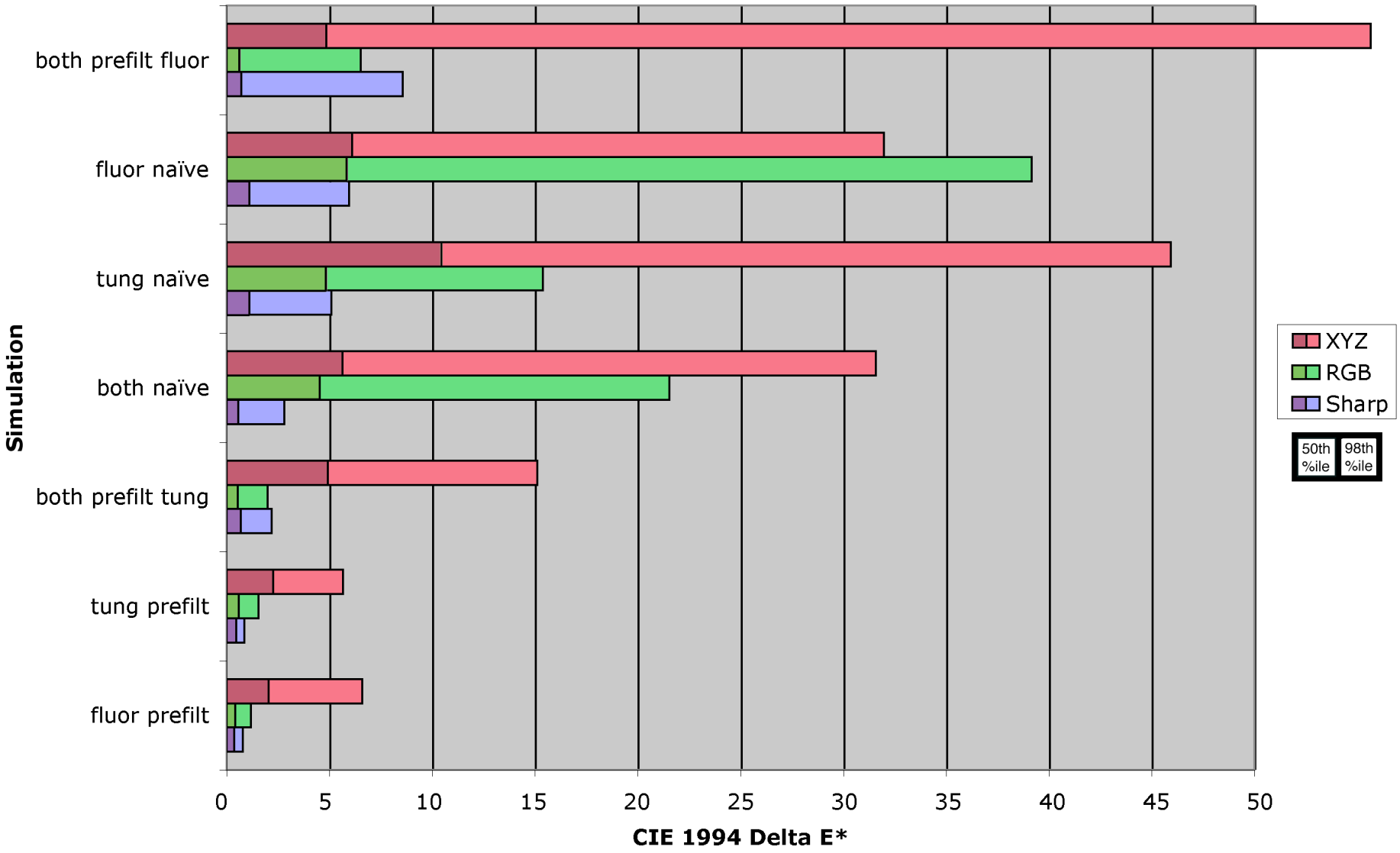






CIE 1998 ΔE* of 5 or above is visible in side-by-side comparisons

ΔE* Error Percentiles for All Experiments

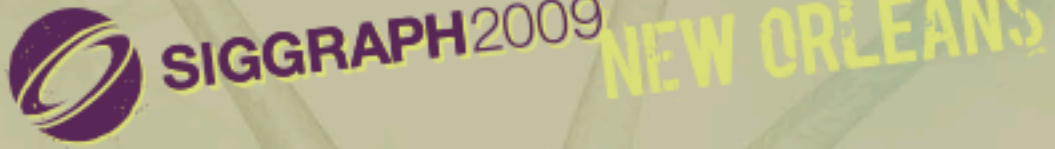


Results Summary

- Prefiltering has ~1/6 the error of naïve rendering for single dominant illuminant
- Prefiltering errors similar to naïve in scenes with strongly mixed illuminants
- CIE *XYZ* color space has 3 times the rendering errors of *sRGB* on average
- *Sharp RGB* rendering space reduces errors to 1/3 that of *sRGB* on average

5. Conclusions

- Prefiltering is simple and practically free
- Avoids aliasing and data mixing problems of full spectral rendering
- Error comparable to 3 component rendering [Peercy93] at 1/3 the cost
- Mixed illuminants and specular reflections no worse than naïve *RGB*



Color Application to HDR Imaging

Greg Ward, Dolby Canada

High Dynamic Range and Color

High Dynamic Range and Color

- Luckily, luminance and color encoding are somewhat separable

High Dynamic Range and Color

- Luckily, luminance and color encoding are somewhat separable
- Most 24-bit/pixel formats restrict both

High Dynamic Range and Color

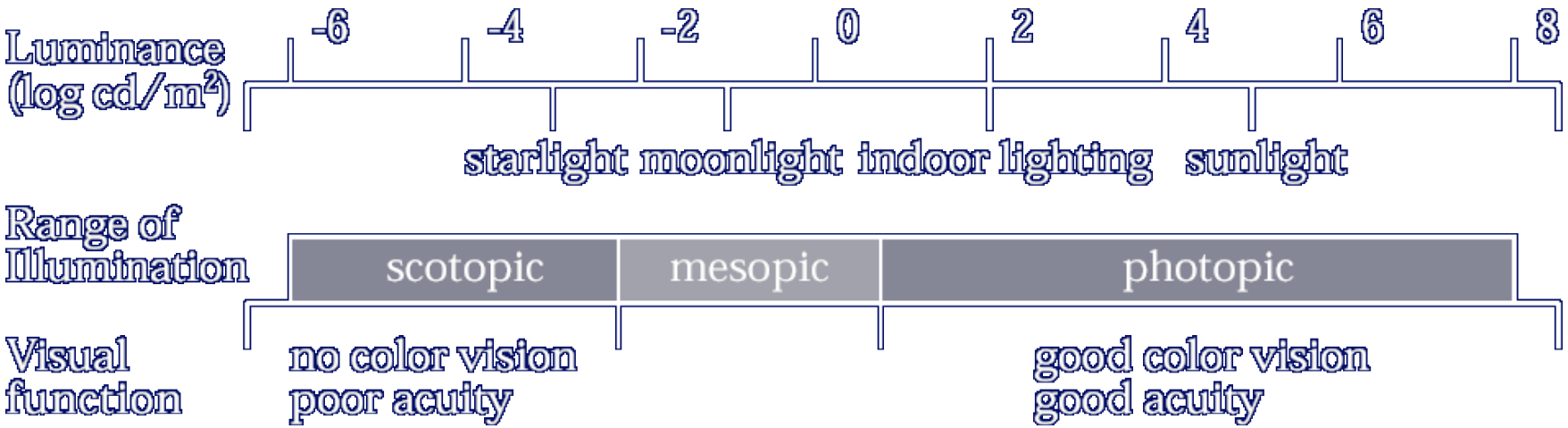
- Luckily, luminance and color encoding are somewhat separable
- Most 24-bit/pixel formats restrict both
- Original HDR format extended dynamic range without including entire visible gamut

High Dynamic Range and Color

- Luckily, luminance and color encoding are somewhat separable
- Most 24-bit/pixel formats restrict both
- Original HDR format extended dynamic range without including entire visible gamut
- This has been rectified in more recent encodings, such as LogLuv & OpenEXR

Dynamic Range

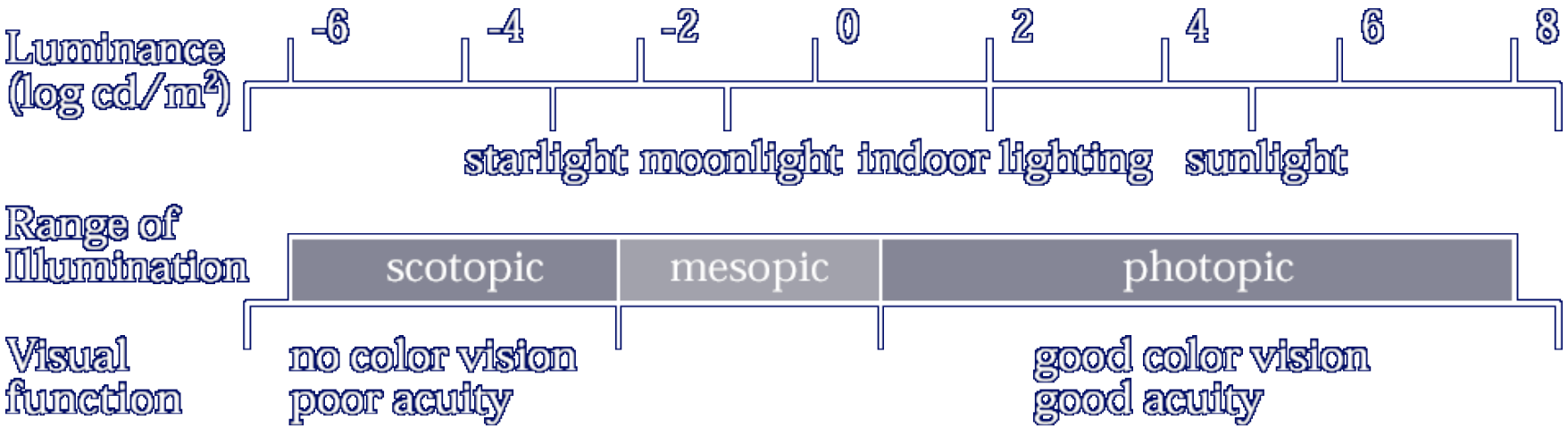
From Ferwerda et al, SIGGRAPH '96



sRGB range ⟷

Dynamic Range

From Ferwerda et al, SIGGRAPH '96

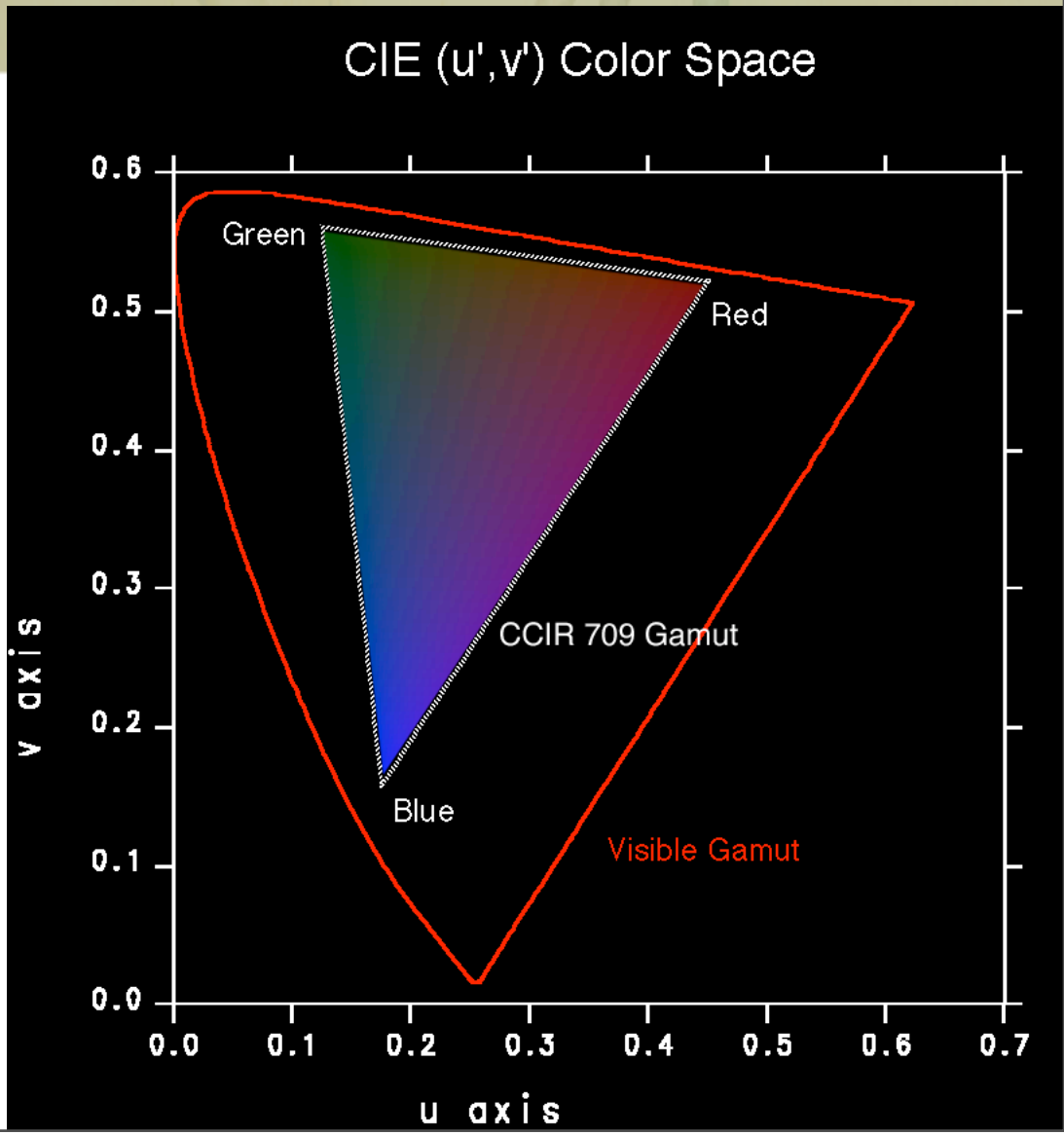


sRGB range ⟷

Human simultaneous range ⟷

CCIR-709 Color Space

- Human visible gamut is much larger than standard display's
- Saturated blues, greens, and purples are lost in sRGB
- Many HDR image formats also cover a larger color gamut



A Gamut Is a Volume!

- HDR can represent brighter colors
- This *delays* saturation near white
- Result is larger color gamut

← Comparison of standard LCD display to BrightSide HDR display

A Gamut Is a Volume!

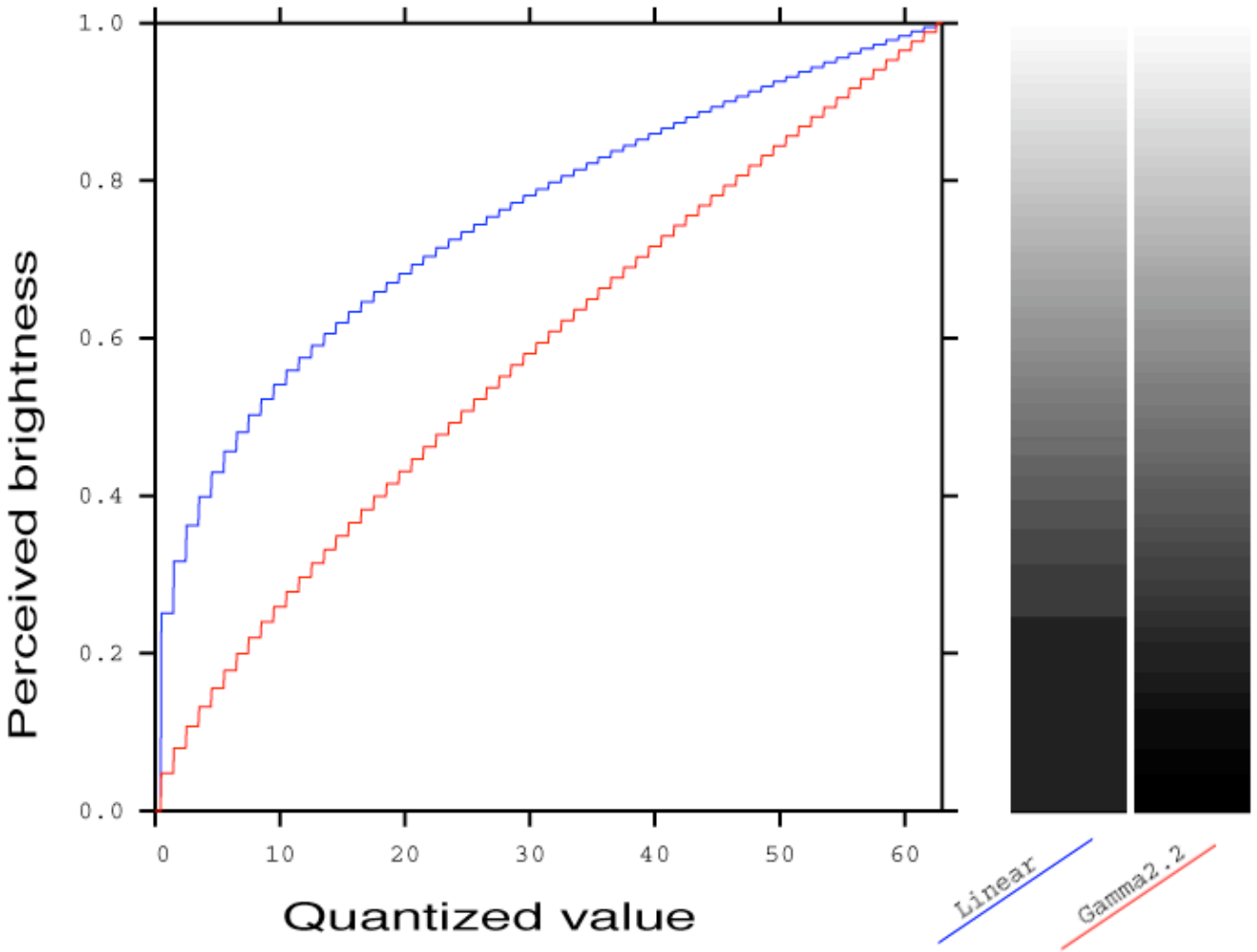
- HDR can represent brighter colors
- This *delays* saturation near white
- Result is larger color gamut

←Comparison of standard LCD display to BrightSide HDR display

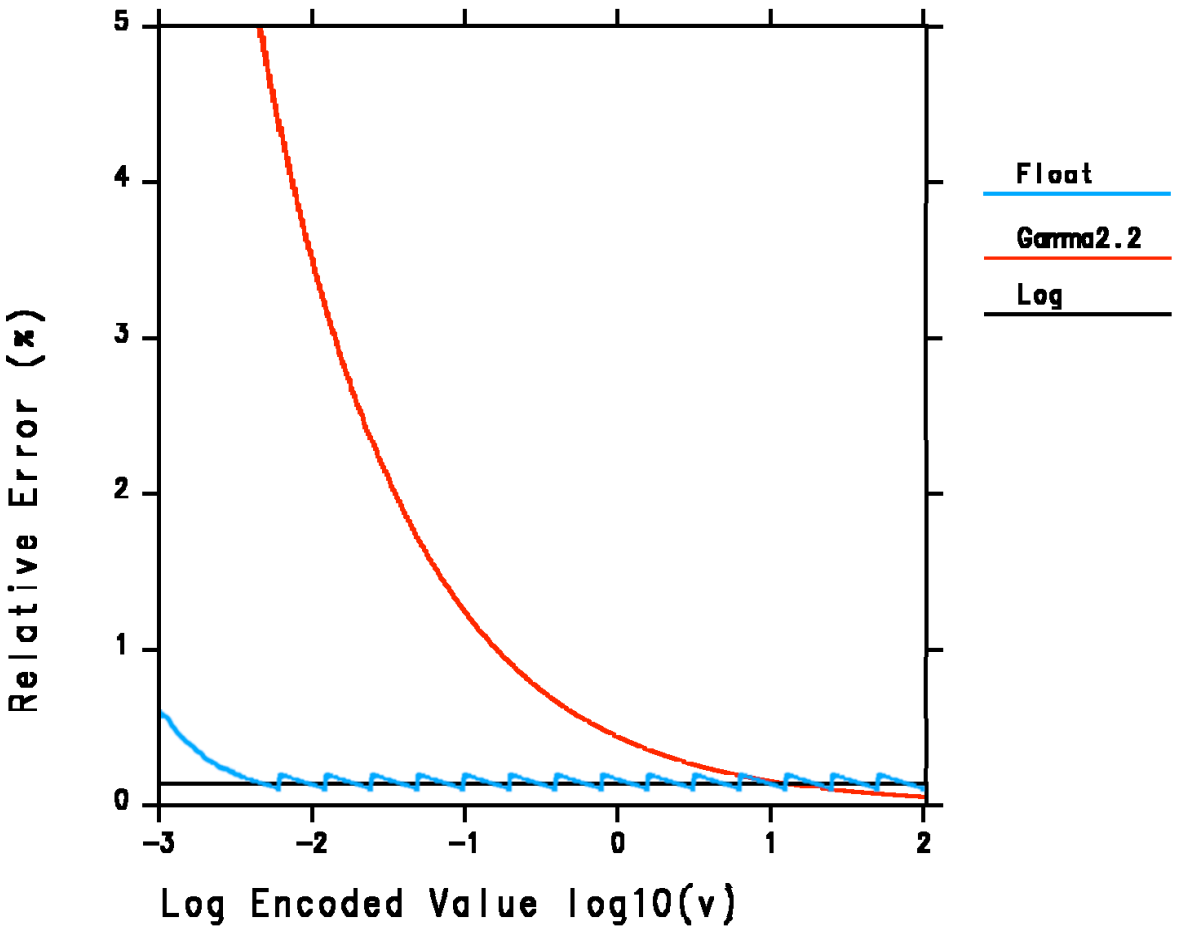
Value Encoding Methods

- Linear quantization
- Gamma function (e.g., CRT curve)
- Logarithmic encoding
- Floating point
- Perceptual

Linear vs. Gamma Encoding

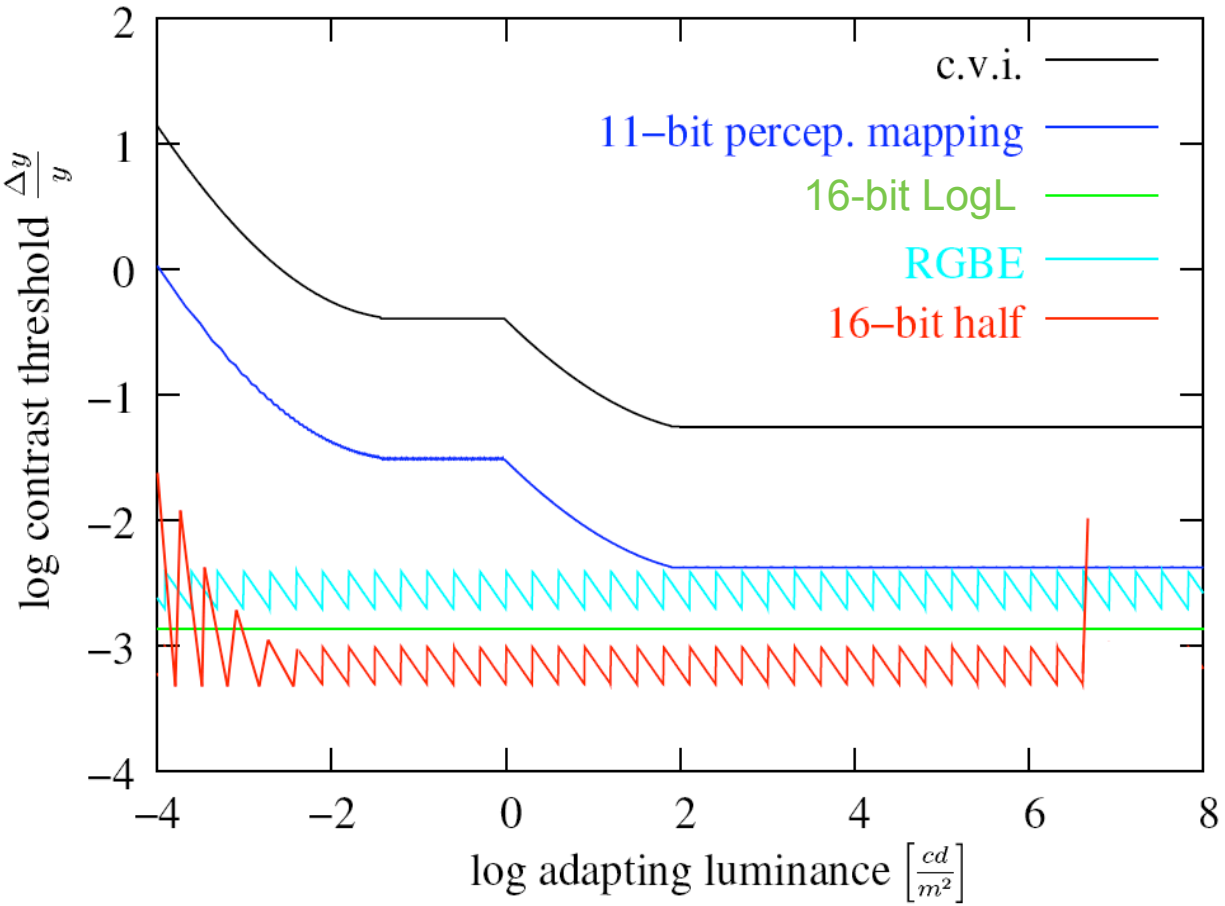


Logarithmic vs. Floating Point



Fictitious 12-bit encoding

Perceptual Encoding



Mantiuk et al., SIGGRAPH 2004

HDR Image & Video Formats

- Available high dynamic-range formats:
  - *Radiance* 32-bit RGBE and XYZE pictures
  - TIFF 48-bit integer and 96-bit float formats
  - SGI 24-bit and 32-bit LogLuv TIFF
  - ILM OpenEXR format
  - BrightSide JPEG-HDR format
- Proposals and extensions:
  - HDR extensions to MPEG from MPI [Mantiuk et al. ‘04, ‘06]
  - HDR extensions to JPEG 2000 from UFL [Xu et al. 2005]
  - HDR texture compression (Lunds Univ. & Nokia papers)

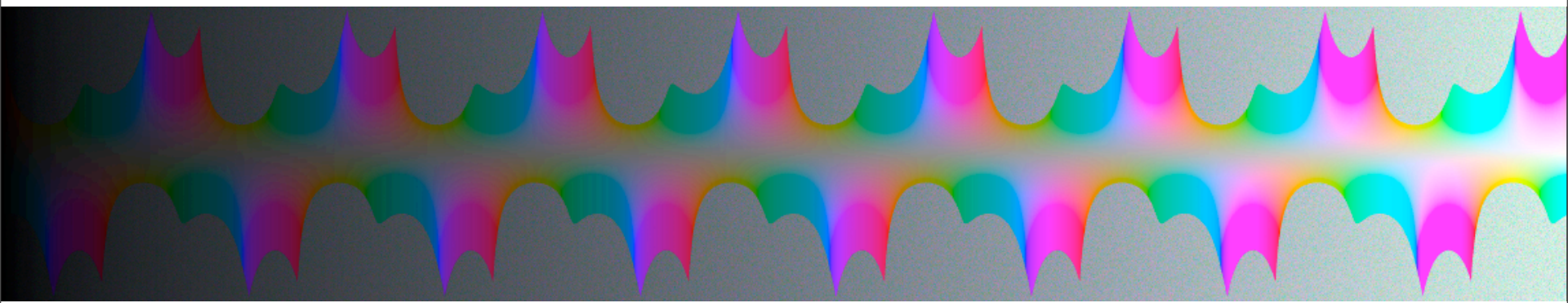
Encoding Comparison Chart

Encoding	Bits / pixel	Dynamic Range	Quant. Step	Covers Gamut
sRGB	24	1:10^1.6	Variable	No
Radiance RGBE Radiance XYZE	32 "	1:10^76 "	1% "	No Yes
LogLuv 24	24	1:10^4.8	1.1%	Yes
LogLuv 32	32	1:10^38	0.3%	Yes
OpenEXR	48	1:10^10.7	0.1%	Yes
JPEG-HDR	1-7	1:10^9.5	Variable	Can

Encoding Comparison Chart

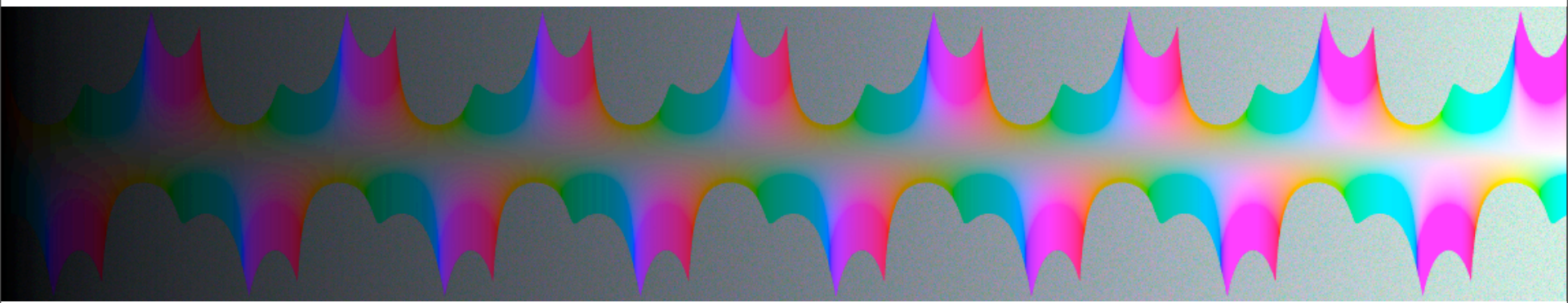
Encoding	Bits / pixel	Dynamic Range	Quant. Step	Covers Gamut
sRGB	24	1:10^1.6	Variable	No
Radiance RGBE	32	1:10^76	1%	No
Radiance XYZE	"	"	"	Yes
LogLuv 24	24	1:10^4.8	1.1%	Yes
LogLuv 32	32	1:10^38	0.3%	Yes
OpenEXR	48	1:10^10.7	0.1%	Yes
JPEG-HDR	1-7	1:10^9.5	Variable	Can

HDR Acid Test Image

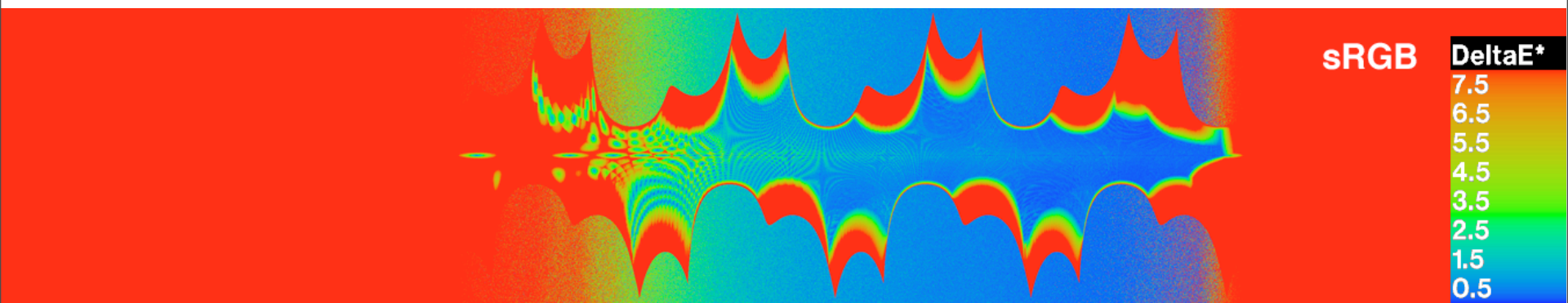


1:10^8 dynamic range, covering visible gamut

HDR Acid Test Image



1:10^8 dynamic range, covering visible gamut



Visible error for 24-bit/pixel sRGB encoding

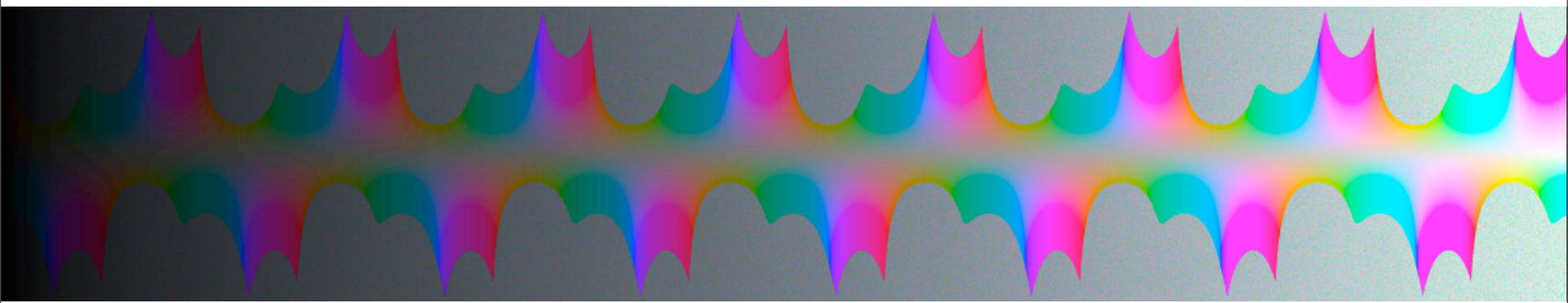
IEEE 96-bit TIFF

- Most accurate representation
- Support (with compression) in Photoshop CS2 & later
- Uncompressed files are enormous
  - 32-bit IEEE floats look like random bits

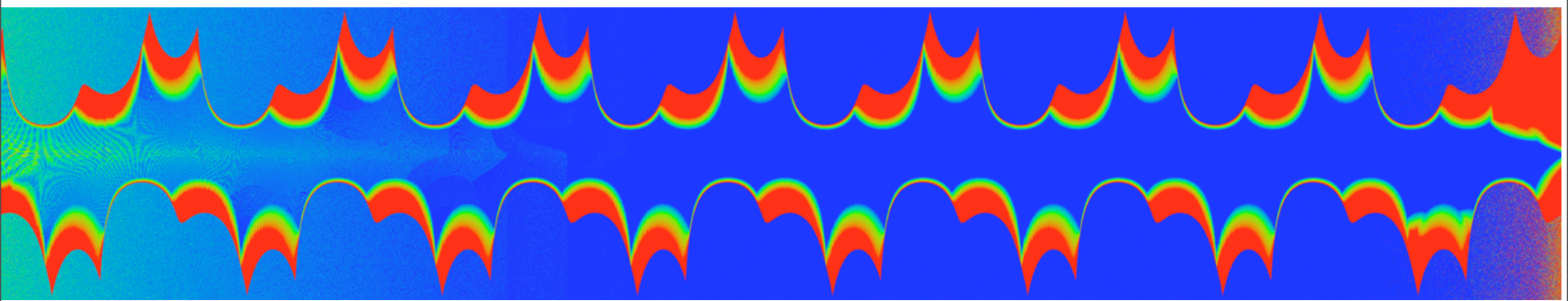
16-bit/sample TIFF (RGB48)

- Supported by Photoshop and TIFF library
- 16 bits each of log red, green, and blue
- 5.4 orders of magnitude in < 1% steps
- LZW lossless compression available
- Does not cover visible gamut
- Most applications think of max. value as “white”

48-bit RGB TIFF Accuracy



1:10^8 dynamic range, covering visible gamut



Visible error for 48-bit/pixel RGB encoding ($\gamma = 2.2$)

Radiance RGBE and XYZE

- Simple format with free source code
- 8 bits each for 3 mantissas and 1 exponent
- 76 orders of magnitude in 1% steps
- Run-length encoding (20% avg. compr.)
- RGBE format does not cover visible gamut
- Dynamic range at expense of accuracy
- Color quantization not perceptually uniform

Radiance Format (.pic, .hdr)

(145, 215, 87, 149) =

(145, 215, 87) * 2^(149-128) =

(1190000, 1760000, 713000)

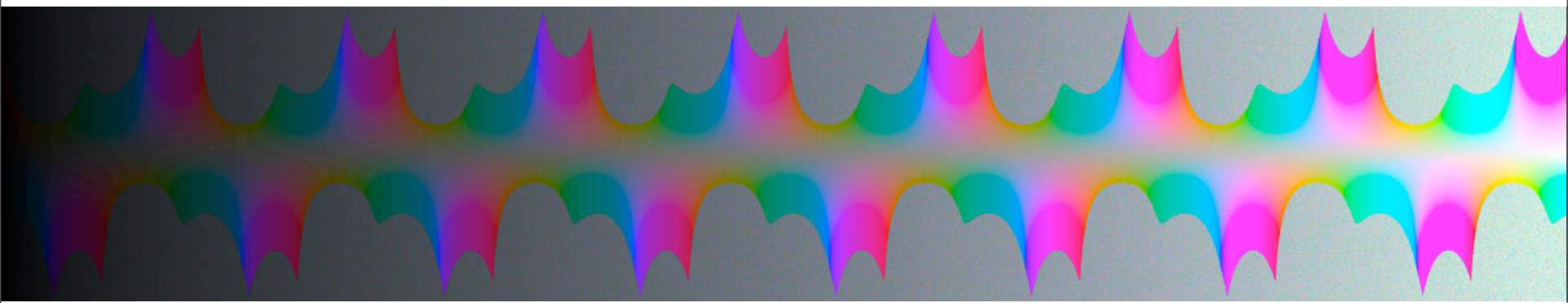
(145, 215, 87, 103) =

(145, 215, 87) * 2^(103-128) =

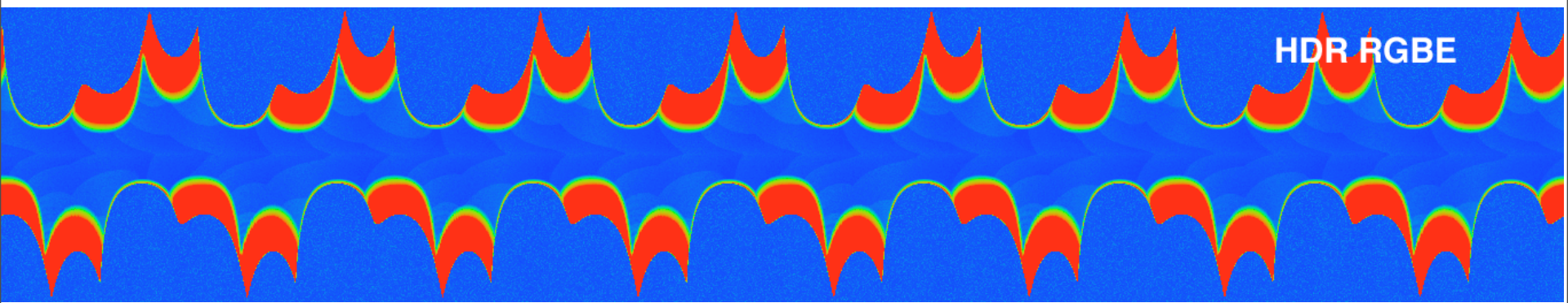
(0.00000432, 0.00000641, 0.00000259)

Ward, Greg. "Real Pixels," in Graphics Gems IV, James Arvo ed., Academic Press, 1994

Radiance RGBE Accuracy

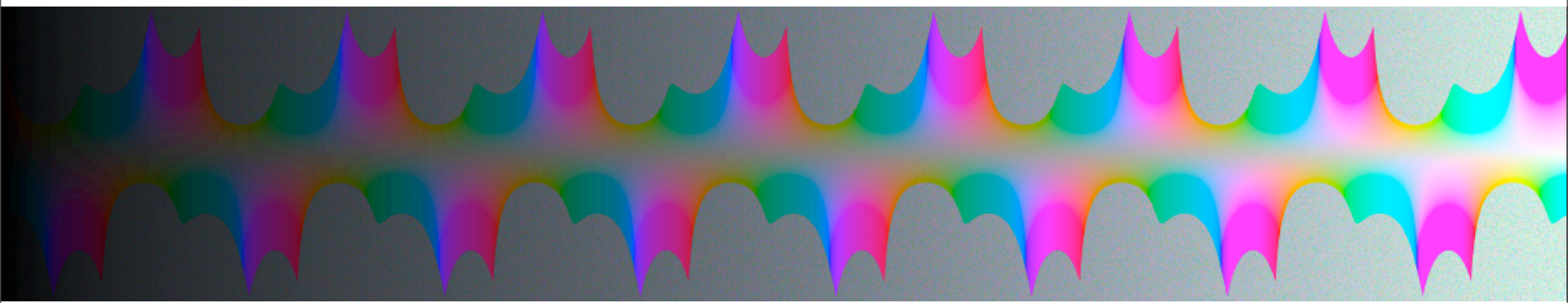


1:10^8 dynamic range, covering visible gamut



Visible error for 32-bit/pixel RGBE encoding

Radiance XYZE Accuracy



1:10^8 dynamic range, covering visible gamut

HDR XYZE

Visible error for 32-bit/pixel XYZE encoding

SGI 24-bit LogLuv TIFF Codec

- Implemented in Leffler's TIFF library
- 10-bit LogL + 14-bit CIE (u',v') lookup
- 4.8 orders of magnitude in 1.1% steps
- Just covers visible gamut and range
- Amenable to tone-mapping as look-up
- Dynamic range is less than we would like
- No compression

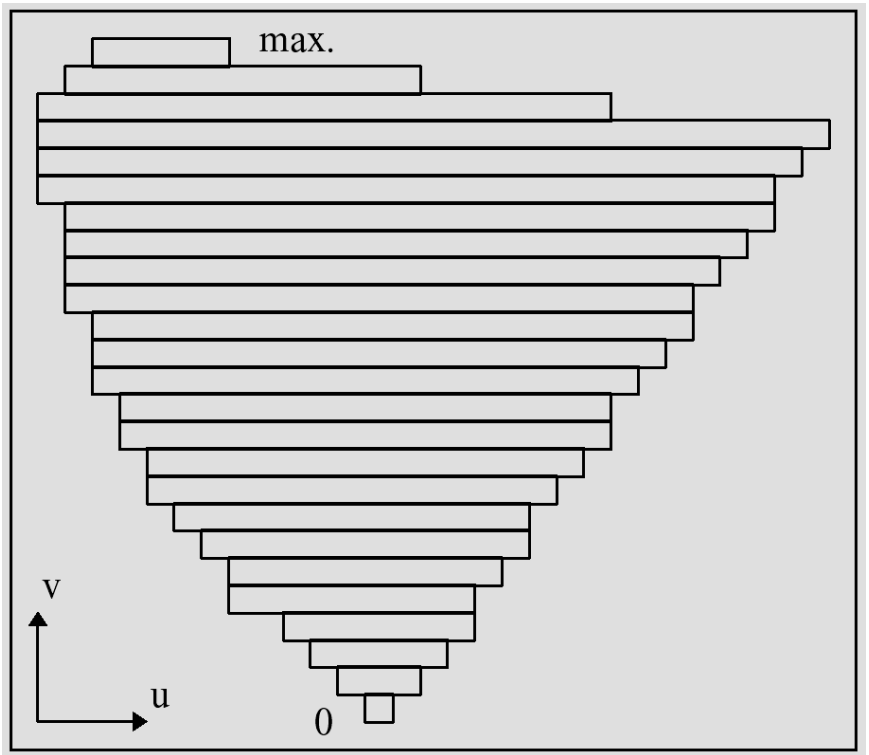
24 -bit LogLuv Pixel

From Larson, CIC '98

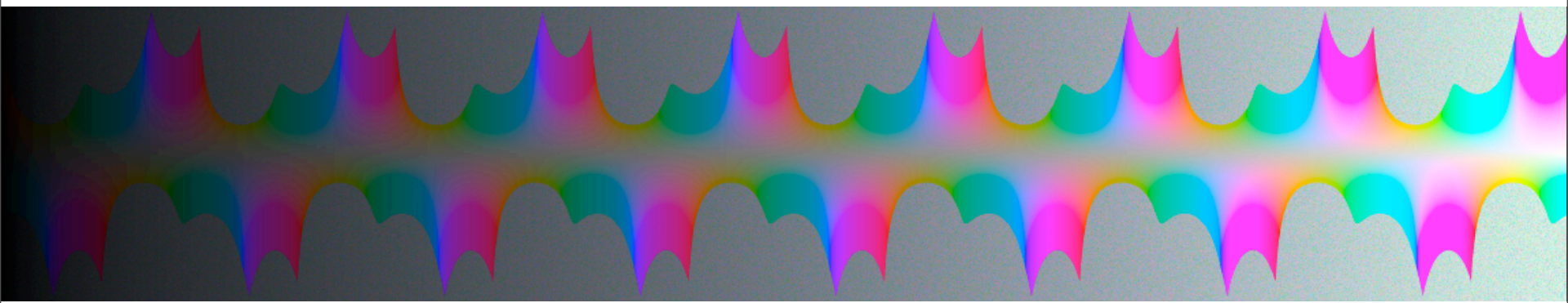
$$L_e = \lfloor 64(\log_2 L + 12) \rfloor$$

$$u' = \frac{4x}{-2x+12y+3}$$

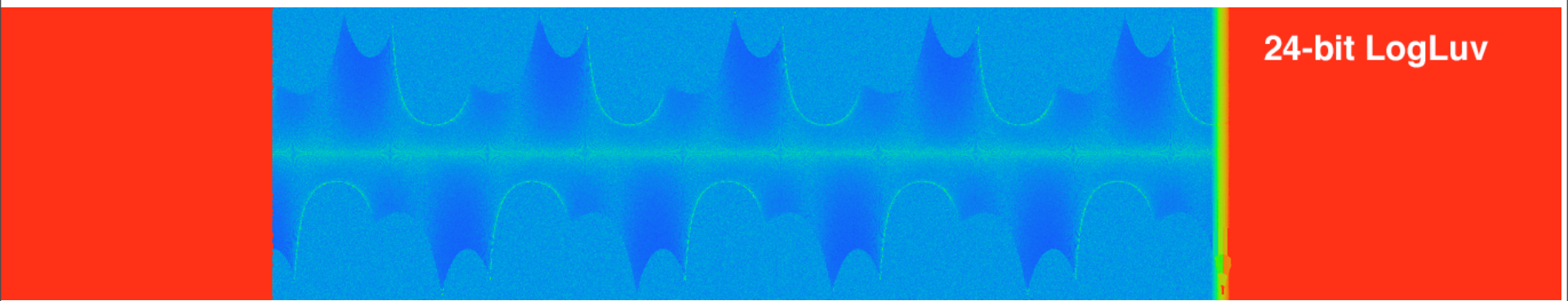
$$v' = \frac{9y}{-2x+12y+3}$$



TIFF LogLuv24 Accuracy



1:10^8 dynamic range, covering visible gamut



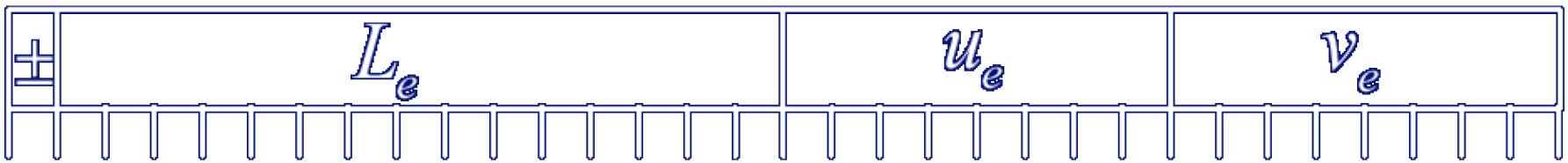
Visible error for 24-bit/pixel LogLuv encoding

SGI 32-bit LogLuv TIFF Codec

- Implemented in Leffler's TIFF library
- 16-bit LogL + 8 bits each for CIE (u',v')
- 38 orders of magnitude in 0.3% steps
- Run-length encoding (30% avg. compr.)
- Allows negative luminance values
- Amenable to tone-mapping as look-up

32-bit LogLuv Pixel

From Larson, JGT '98



$$L_e = \lfloor 256(\log_2 L + 64) \rfloor$$

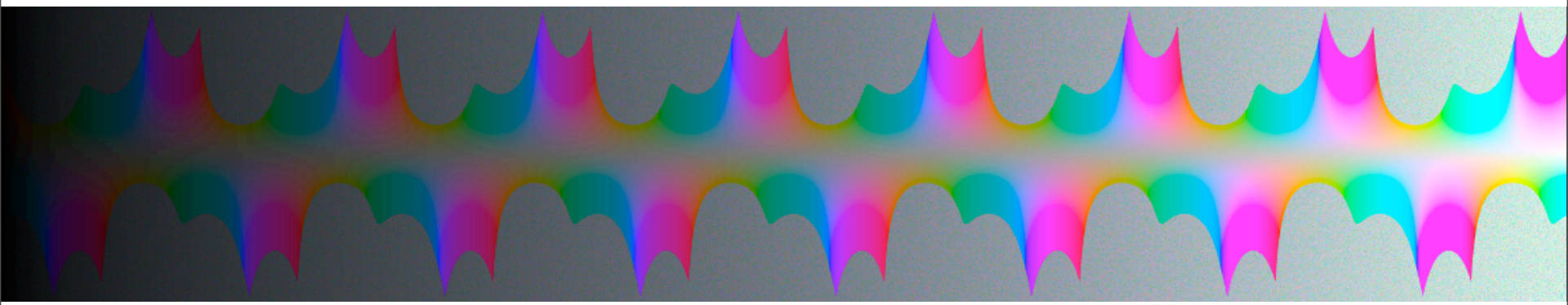
$$u_e = \lfloor 410u' \rfloor$$

$$v_e = \lfloor 410v' \rfloor$$

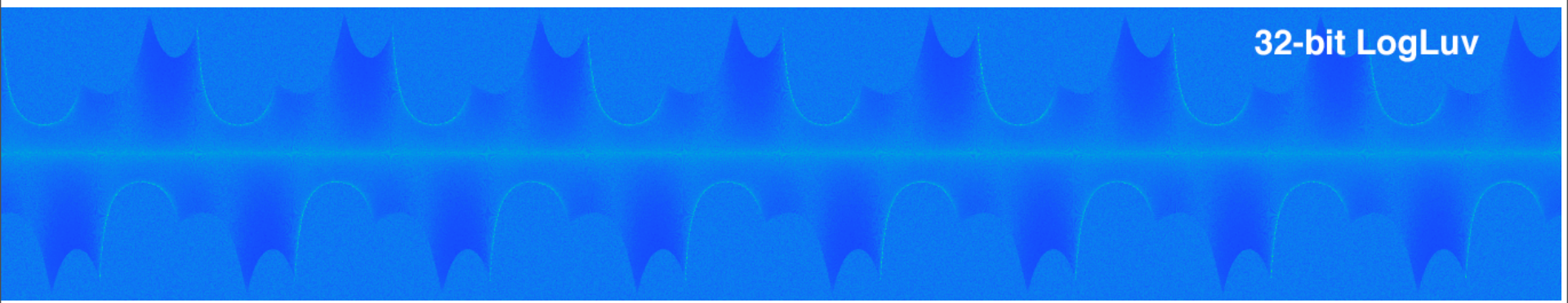
$$u' = \frac{4x}{-2x + 12y + 3}$$

$$v' = \frac{9y}{-2x + 12y + 3}$$

TIFF LogLuv32 Accuracy



1:10^8 dynamic range, covering visible gamut



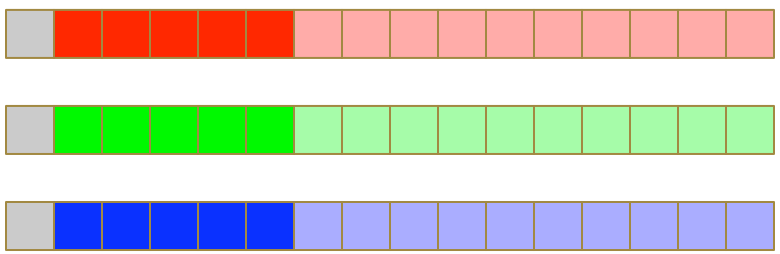
Visible error for 32-bit/pixel LogLuv encoding

ILM OpenEXR Format

- 16-bit/primary floating point (sign-e5-m10)
- 9.6 orders of magnitude in 0.1% steps
- Additional order of magnitude near black
- Wavelet compression of about 40%
- Negative colors and full gamut RGB
- Alpha and multichannel support
- Open Source I/O library released Fall 2002
- Slow to read and write

ILM's OpenEXR (.exr)

6 bytes per pixel, 2 for each channel, compressed

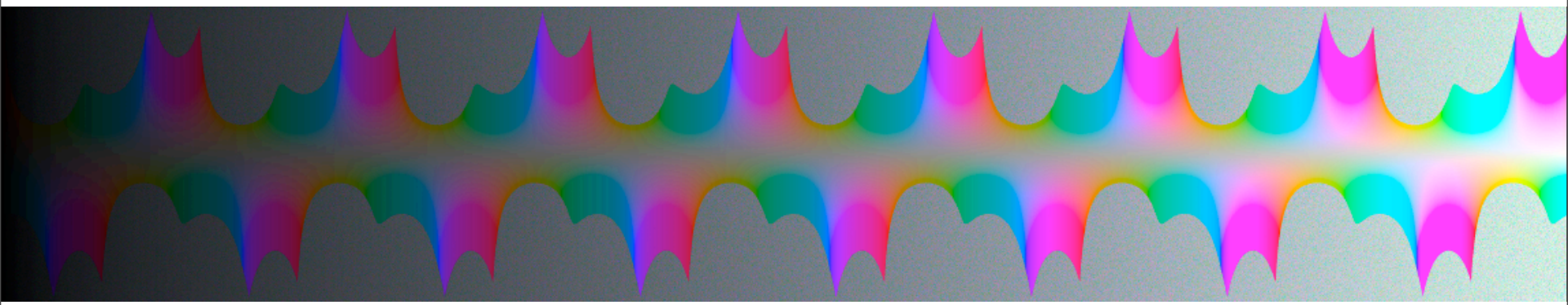


sign exponent mantissa

- Several lossless compression options, 2:1 typical
- Compatible with the "half" datatype in NVidia's Cg
- Supported natively on GeForce FX and Quadro FX

- Available at www.openexr.com

OpenEXR Accuracy



1:10^8 dynamic range, covering visible gamut

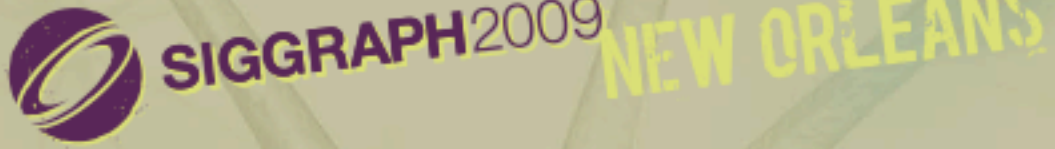
Visible error for 48-bit/pixel EXR `half` encoding

Summary

- Established encodings for HDR imagery exist to cover entire visible gamut
- Wide-gamut and HDR displays are arriving on the professional & consumer markets

Interesting Areas for Research

- Dynamic range and gamut expansion for legacy content
- How do we compress color saturation with DR to maintain color appearance?
- Who records the reference white for HDR imagery and how is it used?
- What is “scene-referred” color, really?



Applications: Color Transfer and Color Appearance Modeling

Erik Reinhard

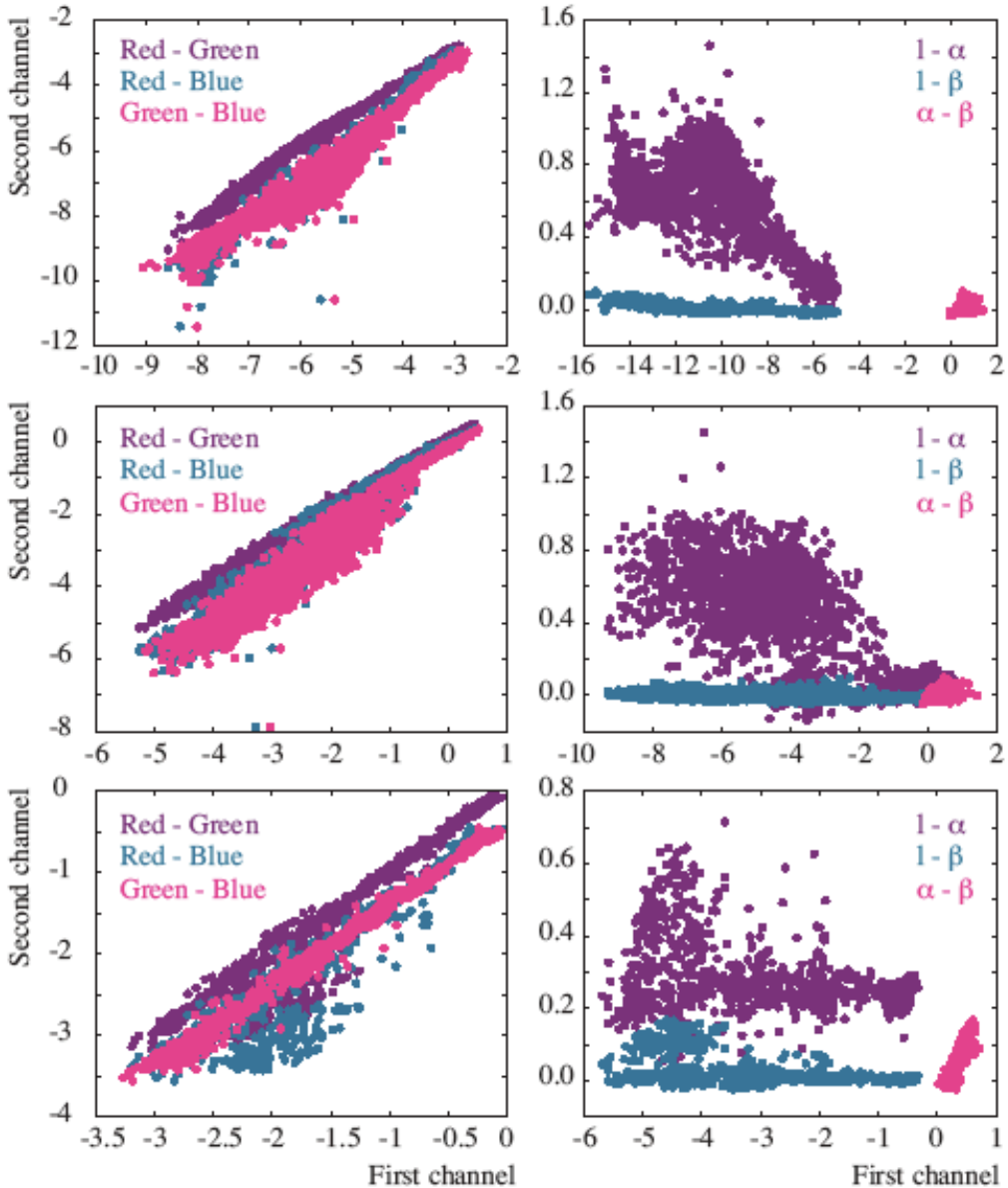
Natural Image Statistics on Color

- Take a set of spectral images of natural scenes
- Convert to LMS cone excitation space
- Run Principle Component Analysis
  - Decorrelates axes by linear transform
  - Can be encoded as a matrix multiplication

- Surprising result:
  - Emerging color space is a color opponent space
  - Human vision designed to decorrelate input
  - Can be used in image processing

Example

Results

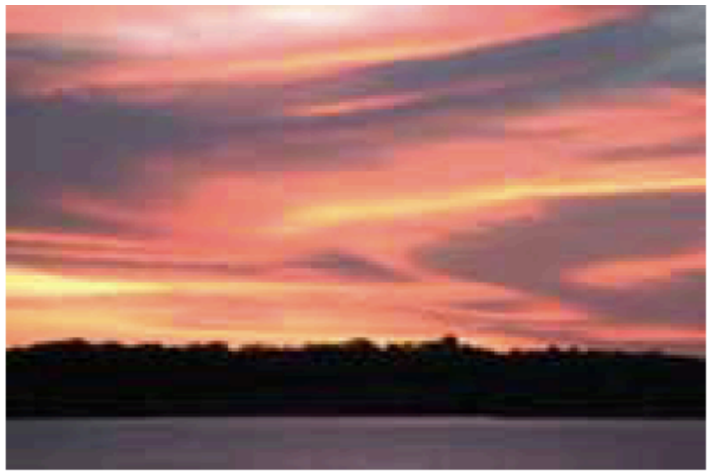

Red - Green
Red - Blue
Green - Blue
1 - α
1 - β
α - β
Second channel
First channel

Color Transfer

- Aim:
  - Take an exemplar image
  - Use its colors to recolor another image
- How:
  - Convert both images to a color opponent space
    - This allows us to treat each color axis independently
  - Compute means and standard deviation for each axis in each image
  - Shift and scale the pixel data so that image has the same mean and standard deviation as exemplar

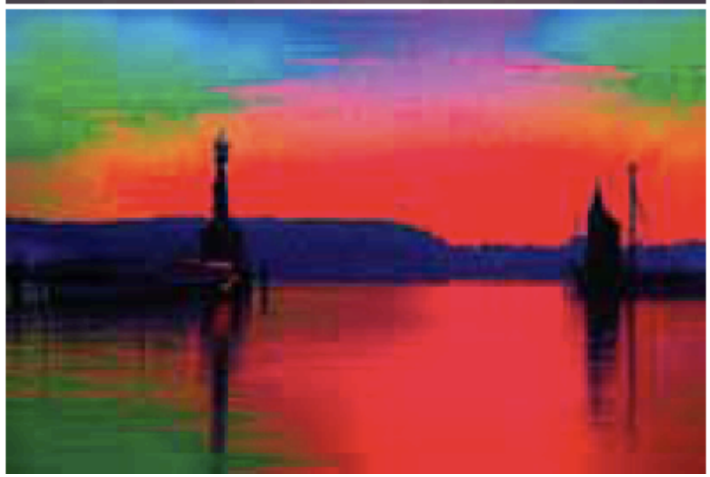
Results

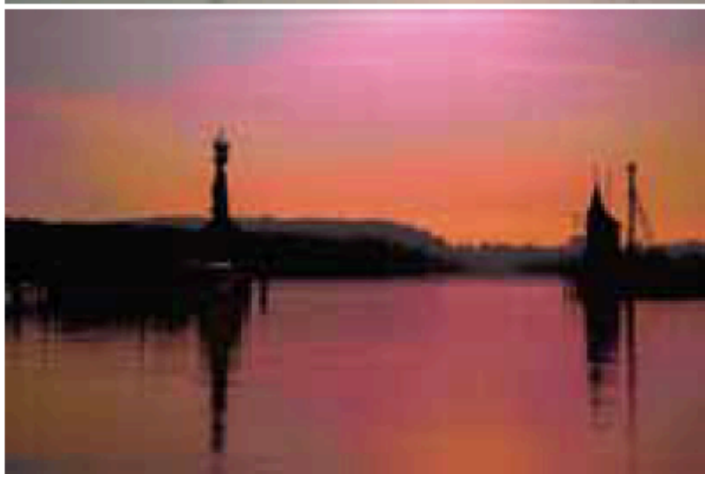
Input 1



Color Transfer

Input 2





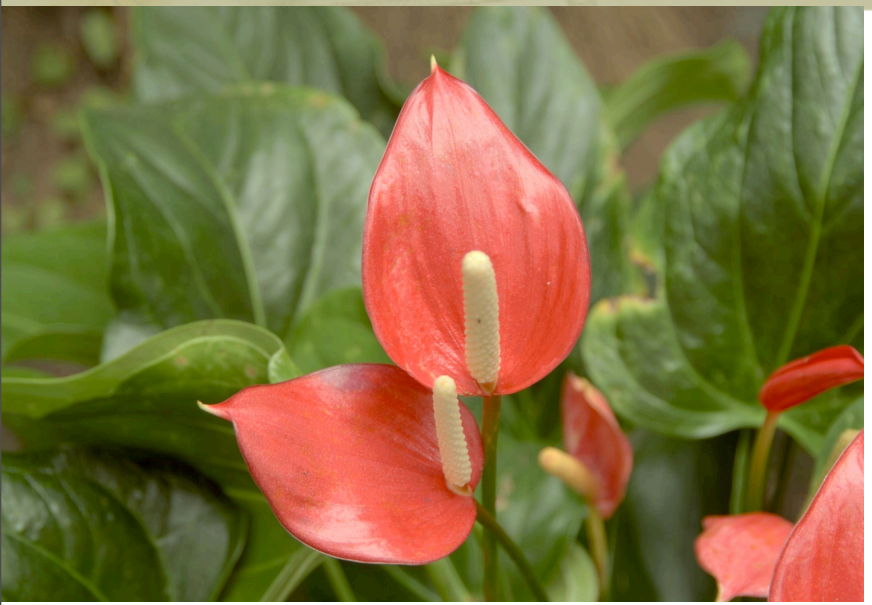
Histogram Matching

Compositing

Superimposed logo

Superimposed logo, with colors from background transferred to the logo

Further Results



Further Results

Conclusions

- Basic color transfer relies on a simple observation: color opponent space decorrelate the data
- This allows us to solve color transfer as three simple 1D problems rather than one complicated 3D problem

Simplified Color Appearance Modeling

- Current Color Appearance Models (CAMS) consist of the following steps:
  - Chromatic adaptation
  - Non-linear response compression
  - Computation of appearance correlates

- We observe
  - Response compression is essentially modeling photoreceptor behavior
  - Separate chromatic adaptation biologically unlikely

von Kries

- von Kries postulated that cones adapt independently
- In CAMs this is not strictly implemented, as chromatic adaptation occurs in a different color space from the non-linear response compression
- We can fix this

Current Linear Response Compression

- Conceptually, sigmoidal compression is applied three time independently

$$V_L = \frac{L^n}{L^n + \sigma}$$

$$V_M = \frac{M^n}{M^n + \sigma}$$

$$V_S = \frac{S^n}{S^n + \sigma}$$

- Note the shared semisaturation constant σ

New Model

- Use per-channel semi-saturation constants

$$\sigma_L = 27.13^{1/0.42}(D(L_W/100) + (1-D))$$
$$\sigma_M = 27.13^{1/0.42}(D(M_W/100) + (1-D))$$
$$\sigma_S = 27.13^{1/0.42}(D(S_W/100) + (1-D))$$

New Model

- Non-linear response compression

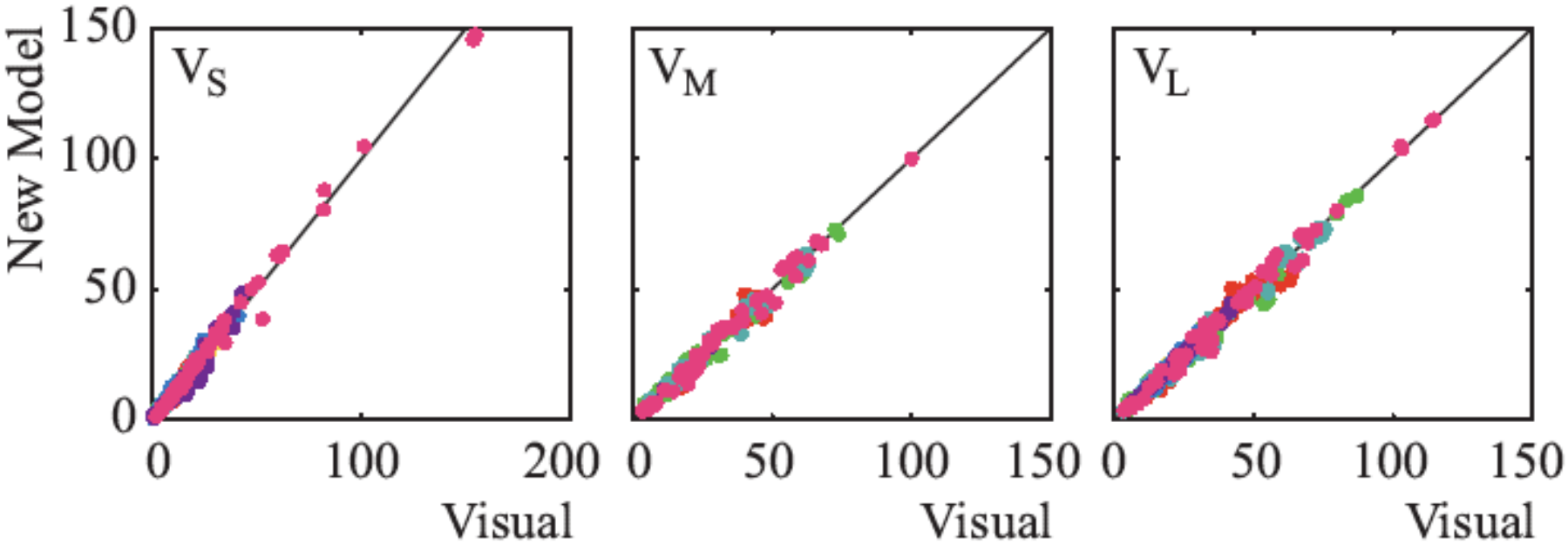
$$L' = 400 \frac{(F_L L/100)^{0.42}}{(F_L L/100)^{0.42} + \sigma_L^{0.42}} + 0.1$$

$$M' = 400 \frac{(F_L M/100)^{0.42}}{(F_L M/100)^{0.42} + \sigma_M^{0.42}} + 0.1$$

$$S' = 400 \frac{(F_L S/100)^{0.42}}{(F_L S/100)^{0.42} + \sigma_S^{0.42}} + 0.1$$

- Note: only difference with existing CAMs is in the semi-saturation constant.

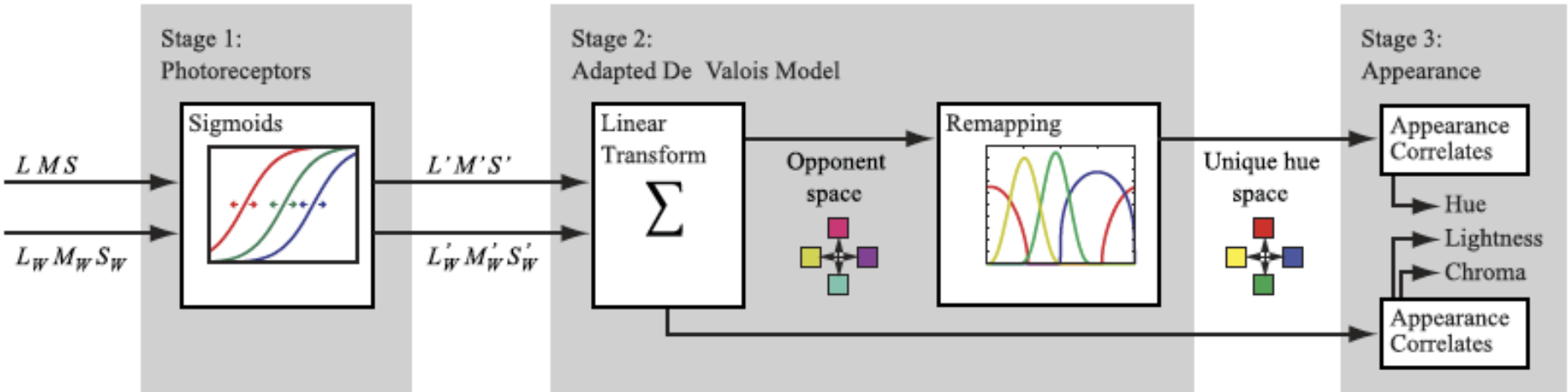
Comparison against Visual Data



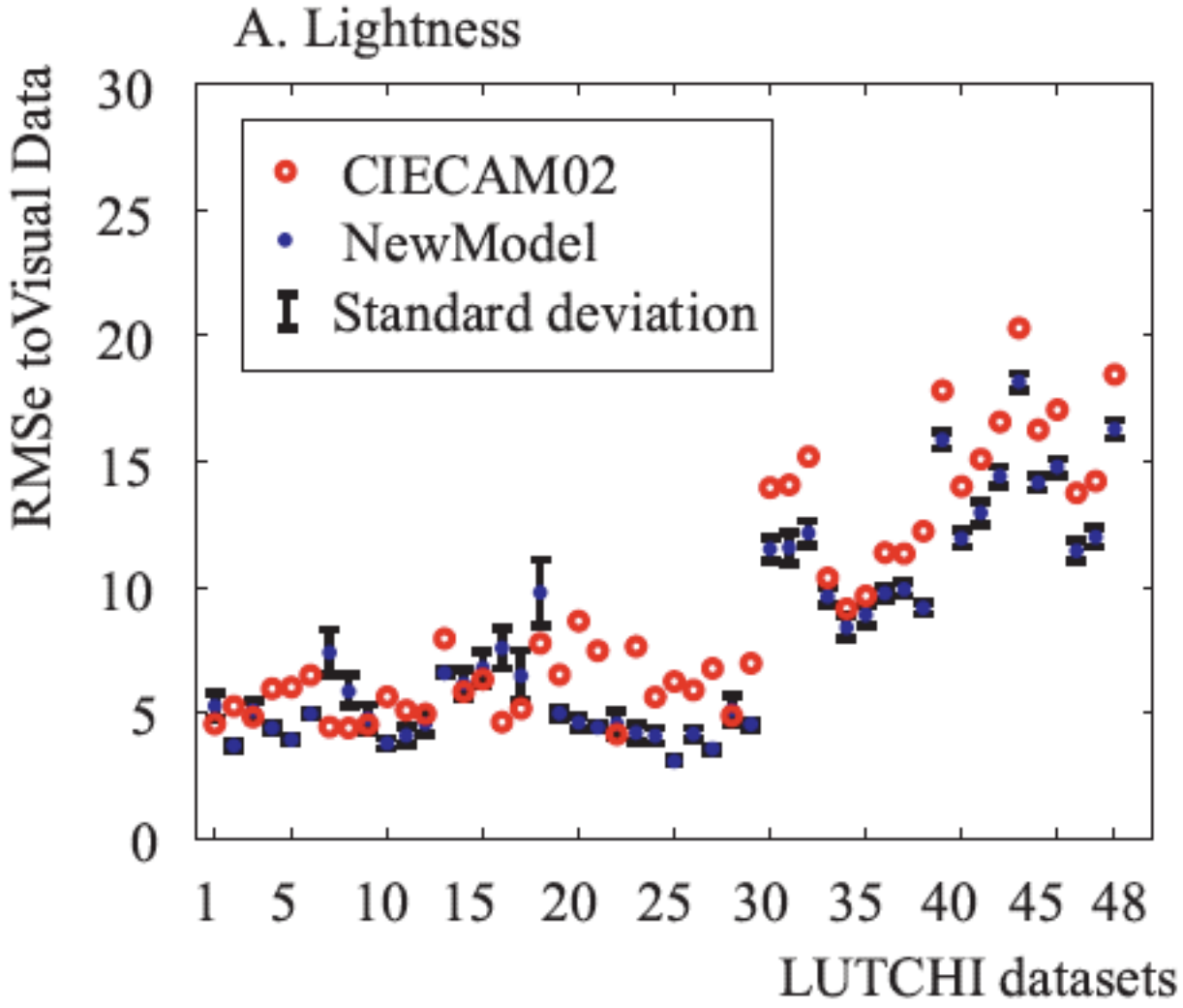
A Problem with Color Opponency

- We call opponent axes ‘red-green’ and ‘yellow-blue’
- However, they aren’t!
- Perceptual ‘red’, ‘green’, ‘yellow’ and ‘blue’ axes are subtly different.
- Can fix by applying a multi-stage color model (de Valois)

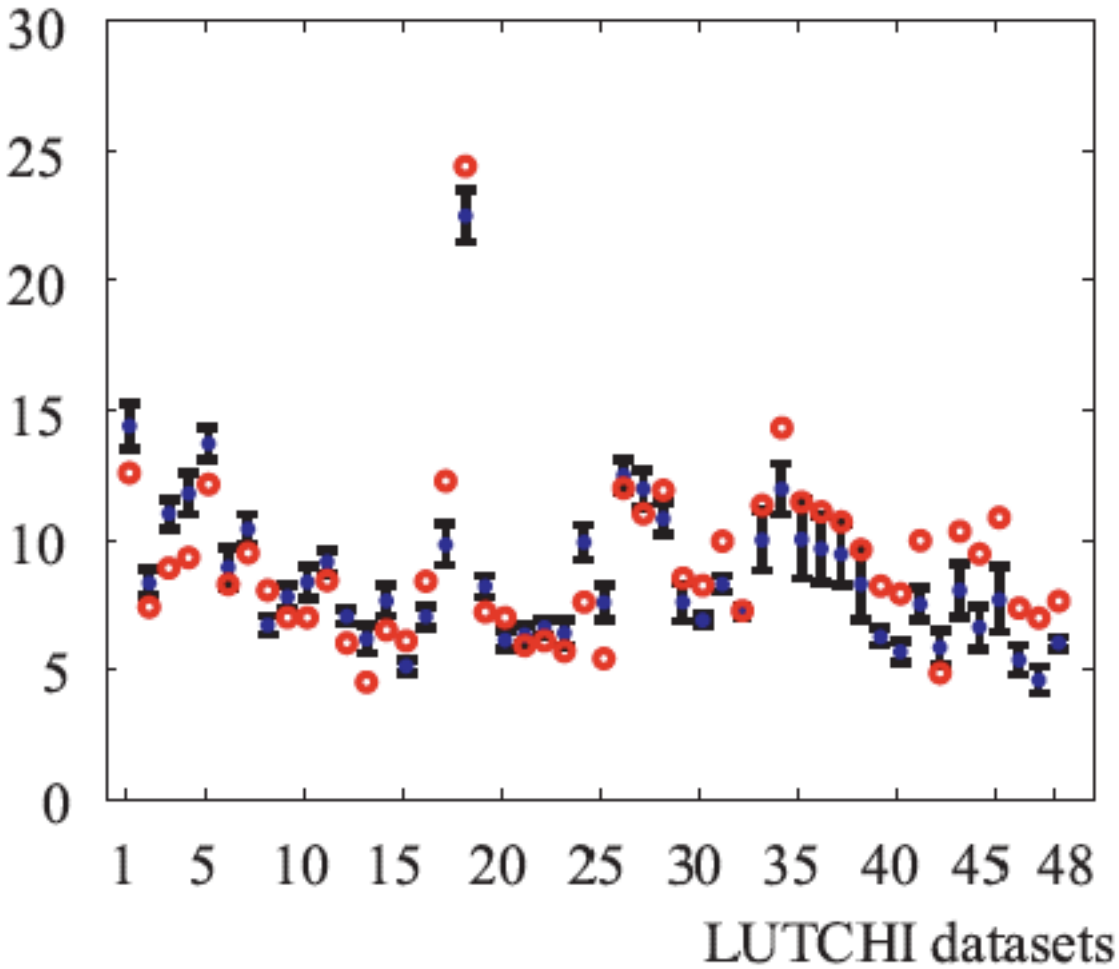
Full Model


Stage 1:
Photoreceptors
Sigmoids
L M S
$L_W M_W S_W$
L'M'S'
$L'_W M'_W S'_W$
Stage 2:
Adapted De Valois Model
Linear
Transform
Σ
Opponent
space
Remapping
Unique hue
space
Stage 3:
Appearance
Appearance
Correlates
Hue
Lightness
Chroma
Appearance
Correlates

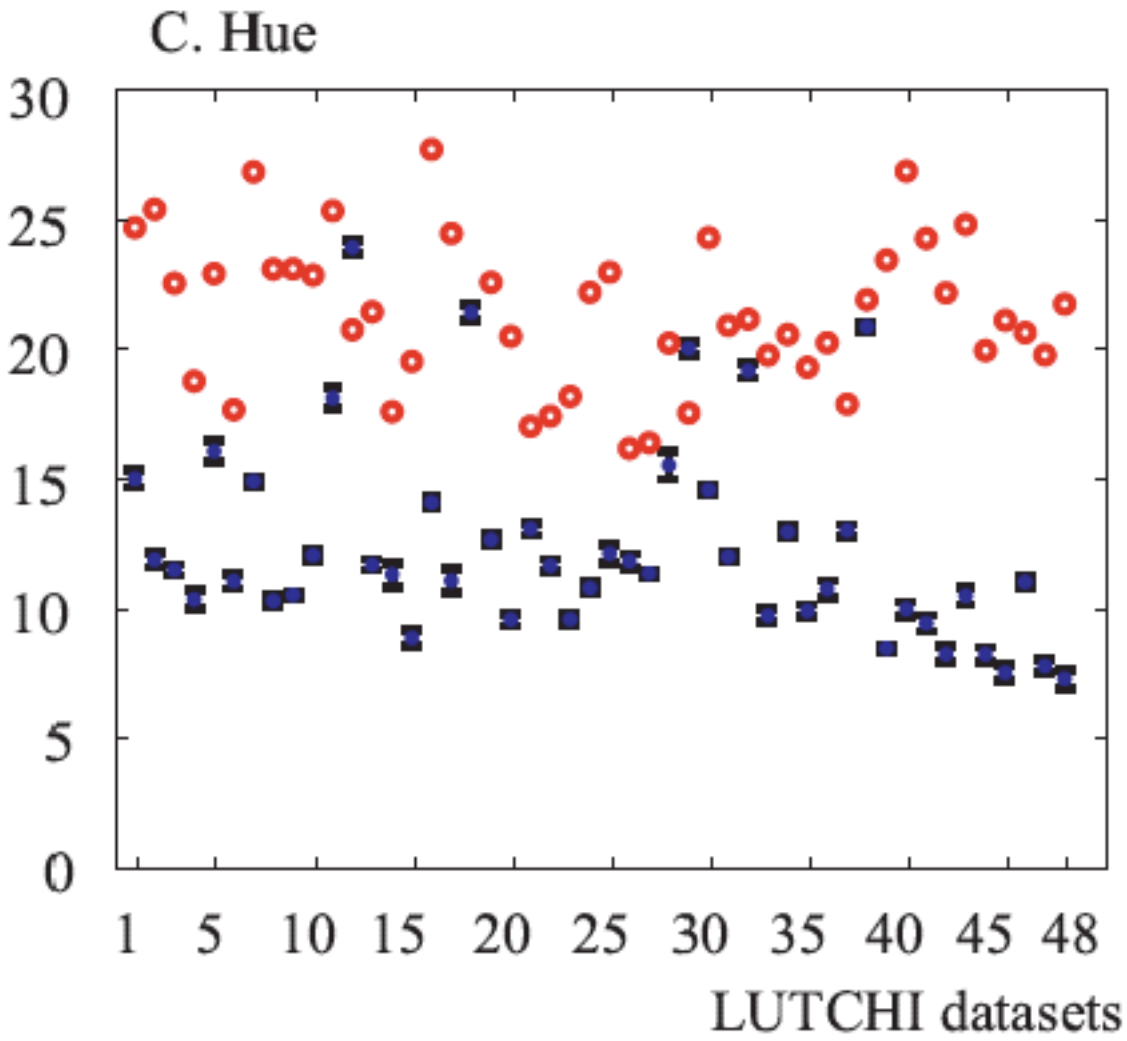
Lightness Results



Chroma Results



Hue Results

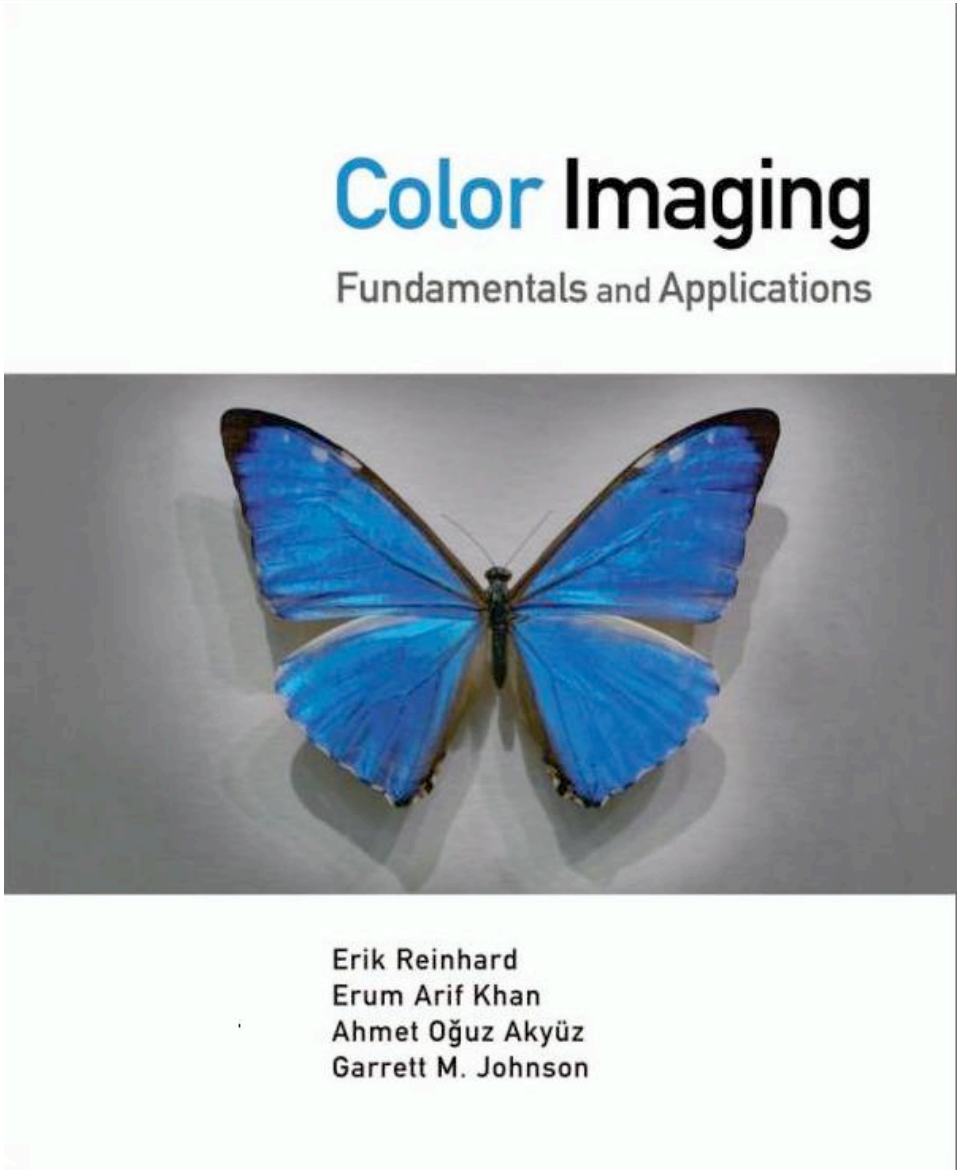

C. Hue
30
25
20
15
10
5
0
1
5
10
15
20
25
30
35
40
45
48
LUTCHI datasets

Conclusions

- Simplified color appearance modeling by merging chromatic adaptation and non-linear response compression
- Creates a true von Kries model

- Better hue predictor by using a multi-stage color model

Further Reading



Overcoming Gamut and Dynamic Range Limitations in Digital Images

Gregory Ward Larson
Silicon Graphics, Inc.
Mountain View, California

Abstract

The human eye can accommodate luminance in a single view over a range of about 10,000:1 and is capable of distinguishing about 10,000 colors at a given brightness. By comparison, typical CRT displays have a luminance range less than 100:1 and cover about half of the visible color gamut. Despite this difference, most digital image formats are geared to the capabilities of conventional displays, rather than the characteristics of human vision. In this paper, we propose two compact encodings suitable for the transfer, manipulation, and storage of full range color images. The first format is a replacement for conventional RGB images, and encodes color pixels as log luminance values and CIE (u',v') chromaticity coordinates. We have implemented and distributed this encoding as part of the standard TIFF I/O library on the net. The second format is proposed as an adjunct to conventional RGB data, and encodes out-of-gamut (and out-of-range) pixels in a supplemental image, suitable as a layer extension to the Flashpix standard. This data can then be recombined with the original RGB layer to obtain a high dynamic range image covering the full gamut of perceivable colors. Finally, we demonstrate the power and utility of full gamut imagery with example images and applications.

Introduction

What is the ultimate use of a digital image? How will it be presented? Will it be modified or adjusted? What kind of monitor will it be displayed on? What type of printer will it be sent to? How accurate do the colors need to be? More often than not, we don't know the answers to these questions a priori. More important, we don't know how these questions will be answered 10 or 100 years from now, when everything we know about digital imaging will have changed, but someone may still want to use our image. We should therefore endeavor to record image data that will be valuable under a broad range of foreseeable and postulated circumstances. Although this seems problematic, there is a simple solution. We may not be able to predict the technology, but we can predict that people will still be the primary consumers.

Most commonly used image standards based on current display technology, i.e., CRT monitors, rather than something less apt to change, i.e., human vision. All RGB standards are limited to a fraction of the visible gamut, since this gamut cannot be contained between any three *real* colors. Even Kodak's PhotoYCC encoding is ultimately geared for CRT display, and doesn't encompass the full gamut of colors or cover more than two orders of magnitude in brightness. The human eye is capable of perceiving at least four orders of magnitude in a daylit scene, and adapting more gradually over seven *additional* orders of magnitude, which means that most digital images encode only a small fraction of what a human observer can see.

In this sense, negative photography is superior to digital imaging in its ability to capture the dynamic range of a scene. A typical, consumer-grade color negative film has about 5-8 f-stops of *exposure latitude*, meaning that it can capture regions of a scene that are 2^5 to 2^8 times brighter than the camera's exposure setting (or dimmer if the image is overexposed), and still have enough range left over to reproduce each region*. Of course, most prints do not make use of the full range, unless a photographer picks up a wand or a cutout in the darkroom, but its presence permits re-exposure during the printing process to optimize the appearance of salient features, such as a person's face.

* To compute the latitude of a film or recording medium, take the log to the base 2 of the total usable dynamic range, from darkest unique value to brightest, and subtract 5 f-stops, which is the approximate range required for a usable image. There are about 3.3 f-stops per order of magnitude.

The question to ask is this: in 10 years or 100 years, what medium will be preferred for old photographs, a digital image, or a negative? Unless we change the way digital images are encoded, the answer in most cases will be a negative. Even considering aging and degradation (processes that can be partially compensated), a negative has both superior resolution and greater dynamic range than an RGB or YCC image. This needn't be the case.

In this paper, we present a compact pixel encoding using a log representation of luminance and a CIE (u',v') representation of color. We call this a *LogLuv* encoding. A log luminance representation means that at any exposure level, there will be equal brightness steps between values. This corresponds well with human visual response, whose contrast threshold is constant over a wide range of adaptation luminances (Weber's law). For color, the use of an approximately uniform perceptual space enables us to record the full gamut of visible colors using step sizes that are imperceptible to the eye. The combination of these two techniques permits us to make nearly optimal use of the bits available to record a given pixel, so that it may be reproduced over a broad range of viewing conditions. Also, since we are recording the full visible gamut and dynamic range, the output or display device can be *anything* and we won't be able to detect any errors or artifacts from our representation, simply because they will be reproduced below the visible threshold.

In this paper, we describe our LogLuv pixel encoding method, followed by a description of our extension to Sam Leffler's free TIFF library. We then put forth a proposal for extending the Flashpix format, and follow this with an example image to demonstrate the value of this encoding, ending with a brief conclusion.

Encoding Method

We have implemented two LogLuv pixel encodings, a 24-bit encoding and a 32-bit encoding. The 24-bit encoding breaks down into a 10-bit log luminance portion and a 14-bit, indexed uv coordinate mapping. Color indexing minimizes waste, allowing us to cover the irregular shape of the visible gamut in imperceptible steps. The 32-bit encoding uses 16 bits for luminance and 8 bits each for u' and v'. Compared to the 24-bit encoding, the 32-bit version provides greater dynamic range and precision at the cost of an extra byte per pixel. The exact interpretations of these two encodings are described below.

24-bit Encoding

In 24 bits, we can pack much more visible information than is commonly stored in three gamma-compressed 8-bit color primary values. By separating luminance and using a log encoding, we can use 10 bits to record nearly 5 orders of magnitude in 1.1% relative steps that will be imperceivable under most conditions. The remaining 14 bits will be used to store a color index corresponding to the smallest distinguishable patch size on a uv color chart. The bit allocation is shown graphically in Fig. 1.

Figure 1. 24-bit encoding. L_e is the encoded log luminance, and C_e is the encoded uv color index.

To compute the integer encoding L_e from real luminance, L, the we use the formula given in Eq. 1a. To compute real luminance from L_e, we use the inverse formula given in Eq. 1b.

$$L_e = \left\lfloor 64\left(\log_2 L + 12\right)\right\rfloor \qquad (1a)$$

$$L = \exp_2\left[\left(L_e + 0.5\right)/64 - 12\right] \qquad (1b)$$

In addition, an L_e value of 0 is taken to equal 0.0 exactly. An L_e value of 1 corresponds to a real luminance value of 0.000248 on an arbitrary scale, and the maximum L_e value of 1023 corresponds to a real value of 15.9 for a dynamic range of 65,000:1, or 4.8 orders of magnitude. It is difficult to compare this range to an 8-bit gamma-compressed encoding, because 1.1% accuracy is possible only near the very top of the 8-bit range. Allowing the luminance error to go as high as 5%, the dynamic range of an 8-bit encoding with a nominal gamma of 2.2 is 47:1, or 1.7 orders of magnitude. This leaves less than one f-stop of exposure latitude, compared to 11 f-stops for our 10-bit log encoding.

To capture full-gamut chrominance using only 14 bits, we cannot afford to waste codes on imaginary colors. We therefore divide our "perceptually uniform" (u',v') color space [8] into equal area regions using a scanline traversal over the visible gamut. This encoding concept is shown graphically in Fig. 2. The actual encoding has many more scanlines of course (163 to be exact), but the figure shows roughly how they are laid out. The minimum code value (0) is at the lower left, and codes are assigned left to right along each scanline until the maximum value (just less than 2^{14}) is assigned to the rightmost value on the top scanline.

$$u' = \frac{4x}{-2x+12y+3} \qquad (2a)$$

$$v' = \frac{9y}{-2x+12y+3} \qquad (2b)$$

To encode a given color, we start with the standard conversion from CIE (x,y) chromaticity to (u',v') shown in Eq. 2. We then look up the appropriate scanline for our *v'* value based on a uniform scanline height, and compute the position within the scanline using our uniform cell width. The index C_e is equal to the total of the scanlines below us plus the cells to the left in this

scanline. Cell width and height are both set to 0.0035 in our implementation, which corresponds to slightly less than the minimum perceptible step in this color space and uses up nearly all of the codes available in 14 bits.

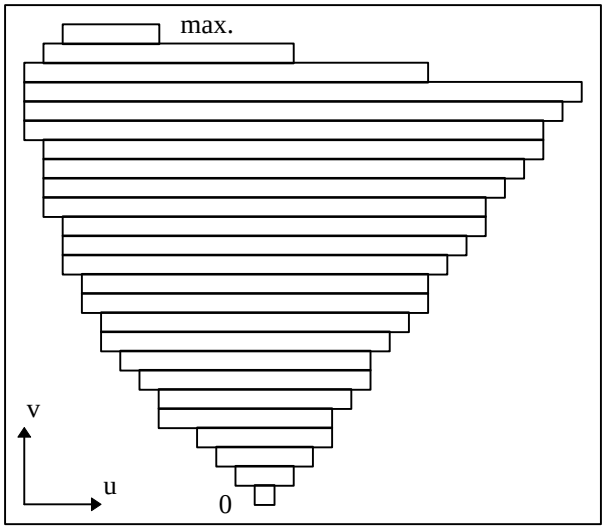


Figure 2. Scanline traversal of (u,v) coordinate space for 14-bit chromaticity encoding.

To get back the (x,y) chromaticity corresponding to a specific color index, we may either use a 16 Kentry look-up table, or apply a binary search to find the scanline containing corresponding to our C_e index. Once we have our original (u',v') coordinates back, we can apply the inverse conversion given in Eq. 3 to get the CIE chromaticity coordinates. (Note that this final computation may also be avoided using the same look-up table.)

$$x = \frac{9u'}{6u' - 16v' + 12} \qquad (3a)$$

$$y = \frac{4v'}{6u' - 16v' + 12} \qquad (3b)$$

32-bit Encoding

The 32-bit encoding is actually simpler, since we have 16 bits for (u',v'), which is more than enough that we can dispense with the complex color indexing scheme. The encoding of luminance is similar, with the addition of a sign bit so that negative luminances may also be encoded. In the remaining 15 bits, we can record over 38 orders of magnitude in 0.27% relative steps, covering the full range of perceivable world luminances in imperceptible steps. The bit breakdown is shown in Fig. 3.

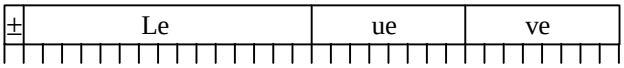


Figure 3. Bit allocation for 32-bit pixel encoding. MSB is a sign bit, and the next 15 bits are used for a log luminance encoding. The uv coordinates are separate 8-bit quantities.

The conversion to and from our log luminance encoding is given in Eq. 4. The maximum luminance using this encoding is 1.84×10^{19}, and the smallest magnitude is 5.44×10^{-20}. As in the 10-bit encoding, an L_e value of 0 is taken to be exactly 0.0. The sign bit is extracted before encoding and reapplied after the conversion back to real luminance.

$$L_e = \lfloor 256(\log_2 L + 64) \rfloor \qquad (4a)$$

$$L = \exp_2\left[(L_e + 0.5)/256 - 64\right] \qquad (4b)$$

As we mentioned, the encoding of chrominance is simplified because we have enough bits to record u_e and v_e separately. Since the gamut of u and v values is between 0 and 0.62, we chose a scale factor of 410 to go between our [0,255] integer range and real coordinates, as given in Eq. 5.

$$u_e = \lfloor 410u' \rfloor \qquad (5a)$$

$$v_e = \lfloor 410v' \rfloor \qquad (5b)$$

$$u' = (u_e + 0.5)/410 \qquad (5c)$$

$$v' = (v_e + 0.5)/410 \qquad (5d)$$

This encoding captures the full color gamut in 8 bits each for u_e and v_e. There will be some unused codes outside the visible gamut, but the tolerance this gives us of 0.0017 units in uv space is already well below the visible threshold. Conversions to and from CIE (x,y) chromaticities are the same as given earlier in Eqs. 2 and 3.

TIFF Input/Output Library

The LogLuv encodings described have been embedded as a new `SGILOG` compression type in Sam Leffler's popular TIFF I/O library. This library is freely distributed by anonymous ftp on ftp.sgi.com in the "/graphics/tiff/" directory.

When writing a high dynamic range (HDR) TIFF image, the LogLuv *codec* (compression/decompresson module) takes floating point CIE XYZ scanlines and writes out 24-bit or 32-bit compressed LogLuv-encoded values. When reading an HDR TIFF, the reverse conversion is performed to get back floating point XYZ values. (We also provide a simple conversion to 24-bit gamma-compressed RGB for the convenience of readers that do not know how to handle HDR pixels.)

An additional tag is provided for absolute luminance calibration, named `TIFFTAG_STONITS`. This is a single floating point value that may be used to convert Y values returned by the reader to absolute luminance in candelas per square meter. This tag may also be set by the application that writes out a HDR TIFF to permit calibrated scaling of values to a reasonable brightness range, where values of 1.0 will be displayed at the maximum output of the destination device. This scale factor may also necessary for calibration of the 24-bit format due to its more limited dynamic range.

Run-length Compression

Although at first it may appear that the 24-bit code is a more compact representation, the 32-bit encoding offers some advantages when it comes to applying nondestructive techniques to reduce storage requirements. By separating the bytes into four streams on each scanline, the 32-bit encoding can be efficiently compressed using an adaptive run-length encoding [3]. Since the top byte containing the sign bit and upper 7 log luminance bits changes very slowly, this byte-stream submits very well to run-length encoding. Likewise, the encoded u_e and v_e byte-streams compress well over areas of constant color. In contrast, the 24-bit encoding does not have a nice byte-stream breakup, so we do not attempt to run-length encode it, and the resulting files are quite often larger than the same data stored in the 32-bit format.

Grayscale Images

For maximum flexibility, a pure luminance mode is also provided by the codec, which stores and retrieves run-length encoded 16-bit log luminance values using the same scheme as applied in the 32-bit LogLuv encoding. There is no real space savings over a straight 32-bit encoding, since the u_e and v_e byte-streams compress to practically nothing for grayscale data, but this option provides an explicit way to specify floating point luminance images for TIFF readers that care.

Raw I/O

It is also possible to decode the raw 24-bit and 32-bit LogLuv data retrieved from an HDR TIFF directly, and this has some advantages for implementing fast tone mapping and display algorithms. In the case of the 24-bit format, one can simply multiply the output of a 1 Kentry L_e table and a 16 Kentry C_e table to get a tone-mapped and gamma-compressed RGB result. The 32-bit encoding requires a little more work, since its precomputed tables are 32 and 64 Kentries, but the same logic applies.

We have implemented this type of integer-math tone-mapping algorithm in an HDR image viewer, and it takes about a second to load and display a 512 by 512 picture on a 180 MHz processor.

Example TIFF Code and Images

Use of this encoding is demonstrated and sample images are provided on the following web site:

http://www.sgi.com/Technology/pixformat/

A converter has been written to and from the *Radiance* floating point picture format [6][7], and serves as an example of LogLuv codec usage. The web site itself also offers programming tips and example code segments.

Example TIFF images using the 32-bit LogLuv and 16-bit LogL encoding are provided on the web site. These images are either scanned from photographic negatives or rendered using *Radiance* and converted to the new TIFF format. Some images are rendered as 360° QuickTime VR panoramas suitable for experiments in HDR virtual reality.

Proposed Extension to Flashpix

The *Flashpix* format was originally developed by Kodak in collaboration with Hewlett-Packard, Live Picture and Microsoft. Its definition and maintenance has since been taken over by the Digital Imaging Group, a consortium of these and other companies. Flashpix is basically a multiresolution JPEG encoding, optimized for quick loading and editing at arbitrary pixel densities. It supports standard RGB as well as YCC color spaces with 8 bits/primary maximum resolution. For further information, see the DIG web site:

http://www.digitalimaging.org

Because Flashpix starts with 8-bit gamma-compressed color primaries, the dynamic range is limited to the same 1.7 orders of magnitude provided by other 24-bit RGB encodings. Furthermore, since JPEG encoding is applied, there will be additional losses and artifacts depending on the source image and the compression quality setting.

We cannot directly replace the JPEG-encoded Flashpix image with our own, alternate format, since this would violate standard compatibility as put forth by Kodak and enforced by the DIG. We must therefore provide any enhancement to the format as an optional extension, which results in a certain amount of redundancy in our case since the same pixels may be represented by two encodings. This is unavoidable.

For our extension, we need a second layer of "deeper" image data be provided for Flashpix users and applications that demand it. There are two ways we might go about this. The simplest method is to completely duplicate the source image in a 24 or 32-bit/pixel LogLuv encoding. On average, this will take roughly four to sixteen times as much space as the original JPEG encoding. A more sophisticated method is to replace only those pixels that are out of gamut or otherwise inadequate in the original encoding. We discuss this method below.

High Dynamic Range Extension Layer

Our proposed extension consists of a layer added to the standard Flashpix format. This layer contains two logical elements, a *presence map* of which pixels are included in the layer, and the list of corresponding 24-bit LogLuv pixels. The presence map may be represented by an entropy-encoded bitmap, which will typically take up 5% to 15% as much space as the JPEG layer. The extended pixels themselves will take between one half and four times as much space as the original JPEG layer, depending on the proportion of out-of-gamut pixels in the original image.

For an image that is entirely within gamut in the JPEG encoding, the presence map will compress to almost nothing, and there will be no LogLuv pixels, so the total overhead will be less than 1% of the original image. If the image is mostly out of the JPEG gamut, then the presence map might take half a bit per pixel, and the additional data will be the same size as a 24-bit RGB image. A typical high dynamic range image with 15% out-of-gamut pixels will take roughly the same space for the extension layer as the multiresolution JPEG layer, so the total image size will be about twice what it was originally. If the information is being accessed over the internet, the HDR layer may be loaded as an option, so it does not cost extra unless and until it is needed.

Example Results

Fig. 4a shows a scanned photograph as it might appear on a PhotoCD using a YCC encoding. Since YCC can capture up to "200% reflectance," we can apply a tone mapping operator to bring this extra dynamic range into our print, as shown in Fig. 5a. However, since many parts of the image were brighter than this 200% value, we still lose much of the sky and circumsolar region, and even the lighter asphalt in the foreground. In Fig. 4b, we see where 35% of the original pixels are outside the gamut of a YCC encoding.

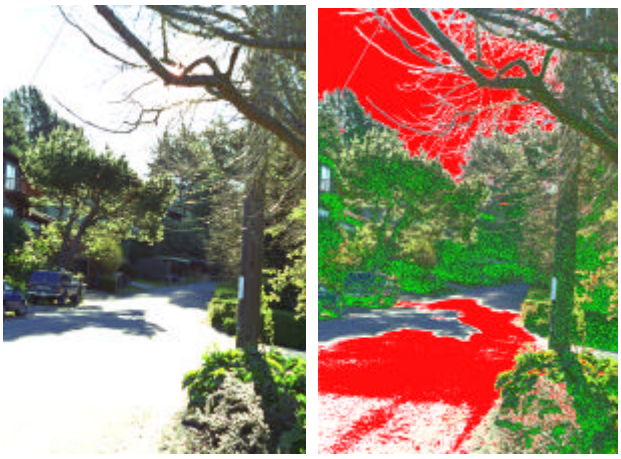


Figure 4. The left image (a) shows a PhotoYCC encoding of a color photograph tone-mapped with a linear operator. The right image (b) shows the out-of-gamut regions. Red areas are too bright or too dim, and green areas have inaccurate color.

Fig. 5b shows the same color negative scanned into our 32-bit/pixel high dynamic range TIFF format and tone mapped using a histogram compression technique [4]. Fig. 6c shows the same HDR TIFF remapped using the perceptual model of Pattanaik et al [5]. Figs. 6a and 6b show details of light and dark areas of the HDR image whose exposure has been adjusted to show the detail captured in the original negative. Without an HDR encoding, this information is either lost or unusable.

Figure 5. The left image(a) shows the YCC encoding after remapping with a high dynamic range tone operator [4]. Unfortunately, since YCC has so little dynamic range, most of the bright areas are lost. The right image (b) shows the same operator applied to a 32-bit HDR TIFF encoding, showing the full dynamic range of the negative.

Figure 6. The upper-left image (a) shows the circumsolar region reduced by 4 f-stops to show the image detail recorded on the negative. The lower-left image (b) shows house details boosted by 3 f-stops. The right image (c) shows our HDR TIFF mapped with the Pattanaik-Ferwerda tone operator [5].

Discussion

It is clear from looking at these images that current methods for tone-mapping HDR imagery, although better than a simple S-curve, are less than perfect. It would therefore be a mistake to store an image that has been irreversibly tone mapped in this fashion, as some scanner

software attempts to do. Storing an HDR image allows us to take full advantage of future improvements in tone mapping and display algorithms, at a nominal cost.

Besides professional photography, there are a number of application areas where HDR images are key. One is lighting simulation, where designers need to see an interior or exterior space as it would really appear, plus they need to evaluate things in terms absolute luminance and illuminance levels. Since an HDR image can store the real luminance in its full-gamut coverage, this information is readily accessible to the designer. Another application is image-based rendering, where a user is allowed to move about in a scene by warping captured or rendered images [1]. If these images have limited dynamic range, it is next to impossible to adapt the exposure based on the current view, and quality is compromised. Using HDR pixels, a natural view can be provided for any portion of the scene, no matter how bright or how dim. A fourth application area is digital archiving, where we are making a high-quality facsimile of a work of art for posterity. In this case, the pixels we record are precious, so we want to make sure they contain as much information as possible. At the same time, we have concerns about storage space and transmission costs, so keeping this data as compact as possible is important. Since our HDR format requires little more space than a standard 24-bit encoding to capture the full visible gamut, it is a clear winner for archiving applications.

Our essential argument is that we can make better use of the bits in each pixel by adopting a perceptual encoding of color and brightness. Although we don't know how a given image might be used or displayed in the future, we do know something about what a human can observe in a given scene. By faithfully recording this information, we ensure that our image will take full advantage of any future improvements in imaging technology, and our basic format will continue to find new uses.

Conclusion

We have presented a new method for encoding high dynamic range digital images using log luminance and uv chromaticity to capture the entire visible range of color and brightness. The proposed format requires little additional storage per pixel, while providing significant benefits to suppliers, caretakers and consumers of digital imagery.

Through the use of re-exposure and dynamic range compression, we have been able to show some of the benefits of HDR imagery. However, it is more difficult to illustrate the benefits of a larger color gamut without carefully comparing hard copy output of various multi-ink printers. Also, since we currently lack the ability to capture highly saturated scenes, our examples would have to be contrived from individual spectral measurements and hypothetical scenes. We therefore leave this as a future exercise.

Future work on the format itself should focus on the application of lossy compression methods (such as JPEG and fractal image encoding) for HDR images. Without such methods, the storage cost for a given resolution may hinder broad acceptance of this representation. Another extension we should look at is multispectral data, which is needed for remote imaging and some types of lighting simulation.

References

1. Paul Debevec, "Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography," *Computer Graphics (Proceedings of ACM Siggraph 98)*.
2. Paul Debevec, Jitendra Malik, "Recovering High Dynamic Range Radiance Maps from Photographs," *Computer Graphics (Proceedings of ACM Siggraph 97)*.
3. Andrew Glassner, "Adaptive Run-Length Encoding," in *Graphics Gems II*, edited by James Arvo, Academic Press, (1991).
4. Greg Larson, Holly Rushmeier, Christine Piatko, "A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes," *IEEE Transactions on Visualization and Computer Graphics*, **3**, 4, (1997).
5. Sumant Pattanaik, James Ferwerda, Mark Fairchild, Don Greenberg, "A Multiscale Model of Adaptation and Spatial Vision for Realistic Image Display," *Computer Graphics (Proceedings of Siggraph 98)*.
6. Greg Ward, "The RADIANCE Lighting Simulation and Rendering System," *Computer Graphics (Proceedings of Siggraph 94)*.
7. Greg Ward, "Real Pixels," in *Graphics Gems II*, edited by James Arvo, Academic Press, (1991).
8. Gunter Wyszecki, W.S. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae*, Second Edition, Wiley, (1982).

Biography

Gregory Ward Larson is a member of the technical staff in the engineering division of SGI. He graduated with an AB in Physics in 1983 from the UC Berkeley, and earned his Master's in CS from San Francisco State in 1985. Greg has done work in physically-based rendering, surface reflectance measurements, and electronic data standards. He is the developer of the widely-used *Radiance* synthetic imaging system and the MGF exchange standard for scene data.

Greg may be reached by e-mail at *gregl@sgi.com*.

Thirteenth Eurographics Workshop on Rendering (2002)
P. Debevec and S. Gibson (Editors)

Picture Perfect *RGB* Rendering Using Spectral Prefiltering and Sharp Color Primaries

Greg Ward[†]

Elena Eydelberg-Vileshin[‡]

Abstract

Accurate color rendering requires the consideration of many samples over the visible spectrum, and advanced rendering tools developed by the research community offer multispectral sampling towards this goal. However, for practical reasons including efficiency, white balance, and data demands, most commercial rendering packages still employ a naive RGB *model in their lighting calculations. This results in colors that are often qualitatively different from the correct ones. In this paper, we demonstrate two independent and complementary techniques for improving* RGB *rendering accuracy without impacting calculation time: spectral prefiltering and color space selection. Spectral prefiltering is an obvious but overlooked method of preparing input colors for a conventional* RGB *rendering calculation, which achieves exact results for the direct component, and very accurate results for the interreflected component when compared with full-spectral rendering. In an empirical error analysis of our method, we show how the choice of rendering color space also affects final image accuracy, independent of prefiltering. Specifically, we demonstrate the merits of a particular transform that has emerged from the color research community as the best performer in computing white point adaptation under changing illuminants: the Sharp* RGB *space.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Color, shading, shadowing, and texture

1. Introduction

It is well-known that the human eye perceives color in a three-dimensional space, owing to the presence of three types of color receptors. Early psychophysical research demonstrated conclusively that three component values are sufficient to represent any perceived color, and these values may be quantified using the CIE *XYZ* tristimulus space[20]. However, because the spectrum of light is continuous, the interaction between illumination and materials cannot be accurately simulated with only three samples. In fact, no finite number of fixed spectral samples is guaranteed to be sufficient — one can easily find pathological cases, for example, a pure spectral source mixed with a narrow band absorber, that require either component analysis or a ludicrous number of fixed samples to resolve. If the rendered spectrum is inaccurate, reducing it to a tristimulus value will usually not hide the problem.

Besides the open question of how many spectral samples to use, there are other practical barriers to applying full spectral rendering in commercial software. First, there is the general dearth of spectral reflectance data on which to base a spectral simulation. This is consistent with the lack of any kind of reflectance data for rendering. We are grateful to the researchers who are hard at work making spectral data available[3, 19], but the ultimate solution may be to put the necessary measurement tools in the hands of people who care about accurate color rendering. Hand-held spectrophotometers exist and may be purchased for the cost of a good laser printer, but few people apply them in a rendering context, and to our knowledge, no commercial rendering application takes spectrophotometer data as input.

The second practical barrier to spectral rendering is white balance. This is actually a minor issue once you know how to address it, but the first time you render with the correct

[†] Exponent – Failure Analysis Associates, Menlo Park, California
[‡] Department of Computer Science, Stanford University, Palo Alto, California

source and reflectance spectra, you are likely to be disappointed by the strong color cast in your output. This is due to the change in illuminant from the simulated scene to the viewing condition, and there is a well-known method to correct for this, which we will cover in Section 2.

The third practical barrier to the widespread acceptance of spectral rendering is what we call the "data mixing problem." What if the user goes to the trouble of acquiring spectral reflectances for a set of surfaces, but they also want to include materials that are characterized in terms of *RGB* color, or light sources that are specified to a different spectral resolution? One may interpolate and extrapolate to some extent, but in the end, it may be necessary to either synthesize a spectrum from *RGB* triples a la Smits' method[14], or reduce all the spectral data to *RGB* values and fall back on three component rendering again.

The fourth practical barrier to full spectral rendering is cost. In many renderings, shading calculations dominate the computation, even in *RGB*. If all of these calculations must be carried out at the maximum spectral resolution of the input, the added cost may not be worth the added benefit.

Many researchers in computer graphics and color science have addressed the problem of efficient spectral sampling[8, 7]. Meyer suggested a point-sampling method based on Gaussian quadrature and a preferred color space, which requires only 4 spectral samples and is thus very efficient[11]. Like other point sampling techniques, however, Meyer's method is prone to problems when the source spectrum has significant spikes in it, as in the case of common fluorescent lighting. A more sophisticated approach employing orthonormal basis functions was presented by Peercy, who uses characteristic vector analysis on combinations of light source and reflectance spectra to find an optimal, orthonormal basis set[13]. Peercy's method has the advantage of handling spiked and smooth spectra with equal efficiency, and he demonstrated accurate results with as few as three orthonormal bases. The additional cost is comparable to spectral sampling, replacing N multiplies in an N-sample spectral model with $M \times M$ multiplies in an M-basis vector model. Examples in his paper showed the method significantly out-performing uniform spectral sampling for the same number of operations. The cost for a 3-basis simulation, the minimum for acceptable accuracy in Peercy's technique, is roughly three times that of a standard *RGB* shading calculation.

In this paper, we present a method that has the same overall accuracy as Peercy's technique, but without the computational overhead. In fact, no modification at all is required to a conventional *RGB* rendering engine, which multiplies and sums its three color components separately throughout the calculation. Our method is not subject to point sampling problems in spiked source or absorption spectra, and the use of an *RGB* rendering space all but eliminates the data mixing problem mentioned earlier. White adaptation is also accounted for by our technique, since we ask the user to identify a dominant source spectrum for their scene. This avoids the dreaded color cast in the final image.

We start with a few simple observations:

1. The direct lighting component is the first order in any rendering calculation, and its accuracy determines the accuracy of what follows.
2. Most scenes contain a single dominant illuminant; there may be many light sources, but they tend to all have the same spectral power distribution, and spectrally differentiated sources make a negligible contribution to illumination.
3. Exceptional scenes, where spectrally distinct sources make roughly equal contributions, cannot be "white balanced," and will look wrong no matter how accurately the colors are simulated. We can be satisfied if our color accuracy is no worse on average than standard methods in the mixed illuminant case.

The spectral prefiltering method we propose is quite simple. We apply a standard CIE formula to compute the reflected *XYZ* color of each surface under the dominant illuminant, then transform this to a white-balanced *RGB* color space for rendering and display. The dominant sources are then replaced by white sources of equal intensity, and other source colors are modified to account for this adaptation. By construction, the renderer gets the exact answer for the dominant direct component, and a reasonably close approximation for other sources and higher order components.

The accuracy of indirect contributions and spectrally distinct illumination will depend on the sources, materials, and geometry in the scene, as well as the color space chosen for rendering. We show by empirical example how a sharpened *RGB* color space seems to perform particularly well in simulation, and offer some speculation as to why this might be the case.

Section 2 details the equations and steps needed for spectral filtering and white point adjustment. Section 3 shows an example scene with three combinations of two spectrally distinct light sources, and we compare the color accuracy of naive *RGB* rendering to our prefiltering approach, each measured against a full spectral reference solution. We also look at three different color spaces for rendering: CIE *XYZ*, linear *sRGB*, and the Sharp *RGB* space. Finally, we conclude with a summary discussion and suggestions for future work.

2. Method

The spectral prefiltering method we propose is a straightforward transformation from measured source and reflectance spectra to three separate color channels for rendering. These input colors are then used in a conventional rendering process, followed by a final transformation into the display *RGB* space. Chromatic adaptation (i.e., white balancing) may take place either before or after rendering, as a matter of convenience and efficiency.

2.1. Color Transformation

Given a source $I(\lambda)$ and a material $\rho_m(\lambda)$ with arbitrary spectral distributions, the CIE describes a standard method for deriving a tristimulus value that quantifies the average person's color response. The *XYZ* tristimulus color space is computed from the CIE "standard observer" response functions, $\bar{x}$, $\bar{y}$, and $\bar{z}$, which are integrated with an arbitrary source illuminant spectrum and surface reflectance spectrum as shown in Eq. (1), below:

$$\begin{aligned} X_m &= \int I(\lambda)\,\rho_m(\lambda)\,\bar{x}(\lambda)\,d\lambda \\ Y_m &= \int I(\lambda)\,\rho_m(\lambda)\,\bar{y}(\lambda)\,d\lambda \\ Z_m &= \int I(\lambda)\,\rho_m(\lambda)\,\bar{z}(\lambda)\,d\lambda \end{aligned} \tag{1}$$

For most applications, the 1971 2° standard observer curves are used, and these may be found in Wyszecki and Stiles[20].

Eq. (1) is very useful for determining metameric color matches, but it does not give us an absolute scale for color appearance. For example, there is a strong tendency for viewers to discount the illuminant in their observations, and the color one sees depends strongly on the ambient lighting and the surround. For example, Eq. (1) might compute a yellow-orange color for a white patch under a tungsten illuminant, while a human observer would still call it "white" if they were in a room lit by the same tungsten source. In fact, a standard photograph of the patch would show its true yellow-orange color, and most novice photographers have the experience of being startled when the colors they get back from their indoor snapshots are not as they remembered them.

To provide for the viewer's chromatic adaptation and thus avoid a color cast in our image after all our hard work, we apply a von Kries style linear transform to our values prior to display[17]. This transform takes an *XYZ* material color computed under our scene illuminant, and shifts it to the equivalent, apparent color XYZ' under a different illuminant that corresponds to our display viewing condition. All we need are the *XYZ* colors for white under the two illuminants as computed by Eq. (1) with $\rho_m(\lambda) = 1$, and a 3×3 transformation matrix, M_C, that takes us from *XYZ* to an appropriate color space for chromatic adaptation. (We will discuss the choice of M_C shortly.) The combined adaptation and display transform is given in Eq. (2), below:

$$\begin{bmatrix} R'_m \\ G'_m \\ B'_m \end{bmatrix} = M_D\, M_C^{-1} \begin{bmatrix} \frac{R'_w}{R_w} & 0 & 0 \\ 0 & \frac{G'_w}{G_w} & 0 \\ 0 & 0 & \frac{B'_w}{B_w} \end{bmatrix} M_C \begin{bmatrix} X_m \\ Y_m \\ Z_m \end{bmatrix}, \tag{2}$$

where

$$\begin{bmatrix} R_w \\ G_w \\ B_w \end{bmatrix} = M_C \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix}$$

for the scene illuminant, and similarly for the display white point, (X'_w, Y'_w, Z'_w).

The display matrix, M_D, that we added to the standard von Kries transform, takes us from CIE *XYZ* coordinates to our display color space. For an *sRGB* image or monitor with D65 white point[15], one would use the following matrix, followed by a gamma correction of $1/2.2$:

$$M_{sRGB} = \begin{bmatrix} 3.2410 & -1.5374 & -0.4986 \\ -0.9692 & 1.8760 & 0.0416 \\ 0.0556 & -0.2040 & 1.0570 \end{bmatrix}$$

If we are rendering a high dynamic-range scene, we may need to apply a tone-mapping operator such as Larson et al[6] to compress our values into a displayable range. The tone operator of Pattanaik et al even incorporates a partial chromatic adaptation model[12].

The choice of which matrix to use for chromatic adaptation, M_C, is an interesting one. Much debate has gone on in the color science community over the past few years as to which space is most appropriate, and several contenders seem to perform equally well in side-by-side experiments[2]. However, it seems clear that *RGB* primary sets that are "sharper" (more saturated) tend to be more plausible than primaries that are inward of the spectral locus[4]. In this paper, we have selected the *Sharp* adaptation matrix for M_C, which was proposed based on spectral sharpening of color-matching data[17]:

$$M_{\text{Sharp}} = \begin{bmatrix} 1.2694 & -0.0988 & -0.1706 \\ -0.8364 & 1.8006 & 0.0357 \\ 0.0297 & -0.0315 & 1.0018 \end{bmatrix}$$

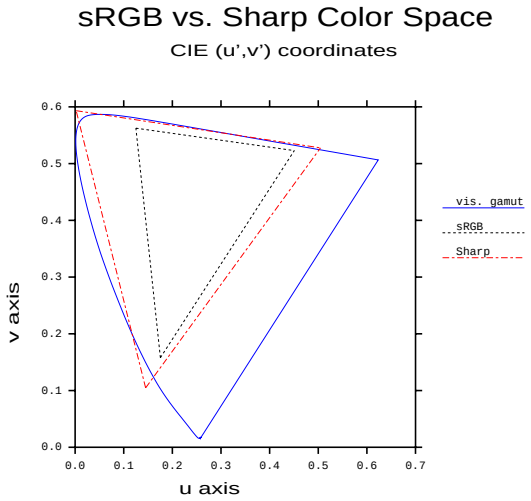


Figure 1: *A plot showing the relative gamuts of the* sRGB *and Sharp color spaces.*

Figure 1 shows a CIE (u',v') plot with the locations of the *sRGB* and Sharp color primaries relative to the visible gamut. Clearly, one could not manufacture a color monitor with Sharp primaries, as they lie just outside the spectral locus. However, this poses no problem for a color transform or a rendering calculation, since we can always transform back to a displayable color space.

In fact, the Sharp primaries may be preferred for rendering and *RGB* image representation simply because they include a larger gamut than the standard *sRGB* primaries. This is not an issue if one can represent color values less than zero and greater than one, but most image formats and some rendering frameworks do not permit this. As we will see in Section 3, the choice of color space plays a significant role in the final image accuracy, even when gamut is not an issue.

2.2. Application to Rendering

We begin with the assumption that the direct-diffuse component is most important to color and overall rendering accuracy. Inside the shader of a conventional *RGB* rendering system, the direct-diffuse component is computed by multiplying the light source color by the diffuse material color, where color multiplication happens separately for each of the three *RGB* values. If this calculation is accurate, it must give the same result one would get using Eq. (1) followed by conversion to the rendering color space. In general, this will not be the case, because the diffuse *RGB* for the surface will be based on some other illuminant whose spectrum does not match the one in the model.

For example, the CIE (x,y) chromaticities and Y-reflectances published on the back of the Macbeth ColorChecker chart[9] are measured under standard illuminant *C*, which is a simulated overcast sky. If a user wants to use the Macbeth color Purple in his *RGB* rendering of an interior space with an incandescent (tungsten) light source, he might convert the published (Y,x,y) reflectances directly to *RGB* values using the inverse of M_{sRGB} given earlier. Unfortunately, he makes at least three mistakes in doing so. First, he is forgetting to perform a white point transform, so there is a slight red shift as he converts from (Y,x,y) under the bluish illuminant *C* to the more neutral D65 white point of *sRGB*. Second, the tungsten source in his model has a slight orange hue he forgets to account for, and there should be a general darkening of the surface under this illuminant, which he fails to simulate. Finally, the weak output at the blue end of a tungsten spectrum makes purple very difficult to distinguish from blue, and he has failed to simulate this metameric effect in his rendering. In the end, the rendering shows something more like violet than the dark blue one would actually witness for this color in such a scene.

If the spectra of all the light sources are equivalent, we can precompute the correct result for the direct-diffuse component and replace the light sources with neutral (white) emitters, inserting our spectrally prefiltered *RGB* values as the diffuse reflectances in each material. We need not worry about how many spectral samples we can afford, since we only have to perform the calculation once for each material in a preprocess. If we intend to render in our display color space, we may even perform the white balance transform ahead of time, saving ourselves the final 3×3 matrix transform at each pixel.

In Section 3, we analyze the error associated with three different color spaces using our spectral prefiltering method, and compare it statistically to the error from naive rendering. The first color space we apply is CIE *XYZ* space, as recommended by Borges[1]. The second color space we use is linear *sRGB*, which has the CCIR-709 *RGB* color primaries that correspond to nominal CRT display phosphors[15]. The third color space is the same one we apply in our white point transformation, the Sharp *RGB* space. We look at cases of direct lighting under a single illuminant, where we expect our technique to perform well, and mixed illuminants with indirect diffuse and specular reflections, where we expect prefiltering to work less effectively.

When we render in CIE *XYZ* space, it makes the most sense to go directly from the prefiltered result of Eq. (1) to *XYZ* colors divided by white under the same illuminant:

$$X_m* = \frac{X_m}{X_w} \qquad Y_m* = \frac{Y_m}{Y_w} \qquad Z_m* = \frac{Z_m}{Z_w}$$

We may then render with light sources using their absolute *XYZ* emissions, and the resulting *XYZ* direct diffuse component will be correct in absolute terms, since they will be remultiplied by the source colors. The final white point adjustment may then be combined with the display color transform exactly as shown in Eq. (2).

When we render in *sRGB* space, it is more convenient to perform white balancing ahead of time, applying both Eq. (1) and Eq. (2) prior to rendering. All light sources that match the spectrum of the dominant illuminant will be modeled as neutral, and spectrally distinct light sources will be modeled as having their *sRGB* color divided by that of the dominant illuminant.

When we render in the Sharp *RGB* space, we can eliminate the transformation into another color space by applying just the right half of Eq. (2) to the surface colors calculated by Eq. (1):

$$\begin{bmatrix} R_m* \\ G_m* \\ B_m* \end{bmatrix} = \begin{bmatrix} \frac{1}{R_w} & 0 & 0 \\ 0 & \frac{1}{G_w} & 0 \\ 0 & 0 & \frac{1}{B_w} \end{bmatrix} M_{\text{Sharp}} \begin{bmatrix} X_m \\ Y_m \\ Z_m \end{bmatrix},$$

Dominant illuminants will again be modeled as neutral, and spectrally distinct illuminants will use:

$$R_s* = \frac{R_s}{R_w} \qquad G_s* = \frac{G_s}{G_w} \qquad B_s* = \frac{B_s}{B_w}$$

The final transformation to the display space will apply the

remaining part of Eq. (2):

$$\begin{bmatrix} R_d \\ G_d \\ B_d \end{bmatrix} = M_D M_{\text{Sharp}}^{-1} \begin{bmatrix} R'_w & 0 & 0 \\ 0 & G'_w & 0 \\ 0 & 0 & B'_w \end{bmatrix} \begin{bmatrix} R'_m \\ G'_m \\ B'_m \end{bmatrix}.$$

3. Results

Our test scene was constructed using published spectral data and simple geometry. It consists of a square room with two light sources and two spheres. One sphere is made of a smooth plastic with a 5% specular component, and the other sphere is made of pure, polished gold (24 carat). The diffuse color of the plastic ball is Macbeth Green[9]. The color of elemental gold is computed from its complex index of refraction as a function of wavelength. The ceiling, floor, and far wall are made of the Macbeth Neutral.8 material. The left wall is Macbeth Red, and the right wall is Macbeth Blue. The near wall, seen in the reflection of the spheres, is the Macbeth BlueFlower color. The left light source is a $2856^\circ K$ tungsten source (i.e., Standard Illuminant A). The right light source is a cool white fluorescent.

All spectral data for our scene were taken from the material tables in Appendix G of Glassner's *Principles of Digital Image Synthesis*[5], and these are also available in the Materials and Geometry Format (MGF)[18]. For convenience, the model used in this paper has been prepared as a set of MGF files and included with our image comparisons in the supplemental materials.

Figure 2 shows a Monte Carlo path tracing of this environment with fluorescent lighting using 69 evenly spaced spectral samples from 380 to 720 nm, which is the resolution of our input data. Using our spectral prefiltering method with the cool white illuminant, we recomputed the image using only three *sRGB* components, taking care to retrace exactly the same ray paths. The result shown in Figure 3 is nearly indistinguishable from the original, with the possible exception of the reflection of the blue wall in the gold sphere. This can be seen graphically in Figure 5, which plots the CIE 1994 Lab $\Delta E*$ color difference[10] in false color. A $\Delta E*$ value of one is just noticeable if the colors are adjacent, and we have found values above five or so to be visible in side-by-side image comparisons.

Using a naive assumption of an equal-energy illuminant, we recomputed the *sRGB* material colors from their reflectance spectra and rendered the scene again, arriving at Figure 4. The rendering took the same time to finish, about a third as long as the full-spectral rendering, and the results are quite different. Both the red wall and the green sphere have changed lightness and saturation from the reference image, the blue wall is reflected as purple in the gold sphere, and the $\Delta E*$ errors shown in Figure 6 are over 20 in large regions. Clearly, this level of accuracy is unacceptable for critical color evaluations, such as selecting a color to repaint the living room.

Illum	Method	*XYZ*		*sRGB*		Sharp	
		50%	98%	50%	98%	50%	98%
tung	naive	10.4	45.9	4.8	15.4	0.8	5.1
	prefilt	2.3	5.7	0.6	1.5	0.5	0.9
fluor	naive	6.1	32.0	5.8	39.2	1.1	6.0
	prefilt	2.0	6.6	0.4	1.2	0.4	0.8
both	naive	5.6	31.6	4.5	21.5	0.6	2.8
	prefilt tung	4.9	15.1	0.5	2.0	0.7	2.2
	prefilt fluor	4.8	55.1	0.6	6.5	0.7	8.6
	Average	5.7	27.4	2.8	12.5	0.7	3.8

Table 1: *CIE 1994 Lab $\Delta E*$ percentiles for our example scene.*

We repeated the same comparisons in CIE *XYZ* and Sharp *RGB* color spaces, then changed the lighting configuration and ran them again. Besides the fluorescent-only lighting condition, we looked at tungsten-only and both sources together. Since the lumen output of the two sources is equal, it was not clear which one to choose as the dominant illuminant, so we applied our prefiltering technique first to one source then to the other. Altogether, we compared 21 combinations of light sources, color spaces, and rendering methods to our multispectral reference solution. The false color images showing the $\Delta E*$ for each comparison are included in the supplemental materials, and we summarize the results statistically in Table 1 and Figure 7.

Table 1 gives the 50th percentile (median) and 98th percentile $\Delta E*$ statistics for each combination of method, lighting, and color space. These columns are averaged to show the relative performance of the three rendering color spaces at the bottom. Figure 7 plots the errors in Table 1 as a bar chart. The 50th percentile errors are coupled with the 98th percentile errors in each bar. In all but one simulation, the Sharp *RGB* color space keeps the median error below the detectable threshold, and the majority of the Sharp renderings have 98% of their pixels below a $\Delta E*$ of five relative to the reference solution, a level at which it is difficult to tell the images apart in side-by-side comparisons. The smallest errors are associated with the Sharp color space and spectral prefiltering with a single illuminant, where 98% of the pixels have errors below the detectable threshold. In the mixed illuminant condition, spectral prefiltering using tungsten as the dominant illuminant performs slightly better than a naive assumption, and prefiltering using cool white as the dominant illuminant performs slightly worse. The worst performance by far is seen when we use CIE *XYZ* as the rendering space, which produces noticeable differences above five for over 2% of the pixels in every simulation, and a *median* $\Delta E*$ over five in each naive simulation.

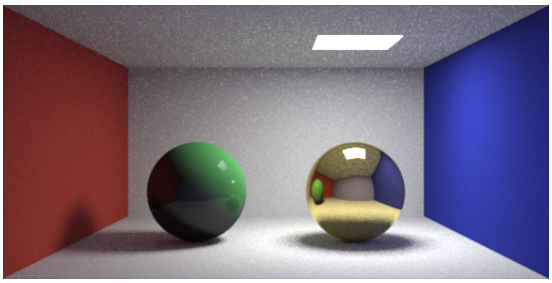


Figure 2: *Our reference multi-spectral solution for the fluorescent-only scene.*

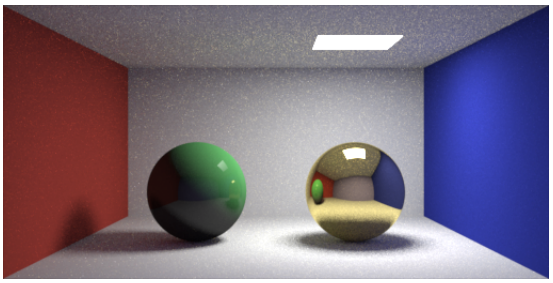


Figure 3: *Our prefiltered* sRGB *solution for the fluorescent-only scene.*

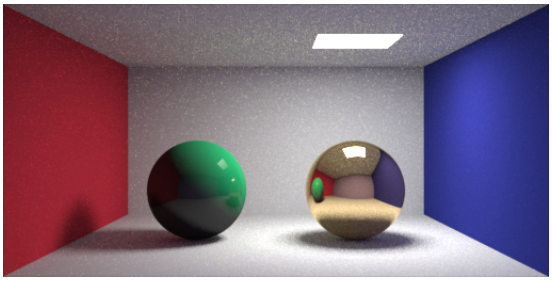


Figure 4: *Our naive* sRGB *solution for the fluorescent-only scene.*

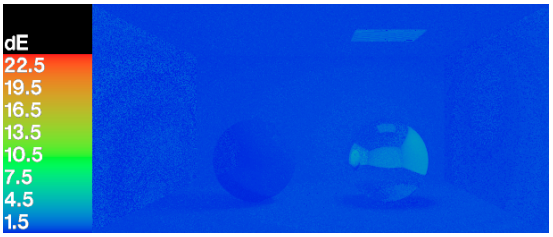


Figure 5: *The* $\Delta E*$ *error for the prefiltered* sRGB *solution.*

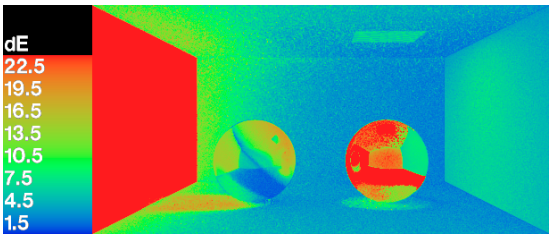


Figure 6: *The* $\Delta E*$ *error for the naive* sRGB *solution.*

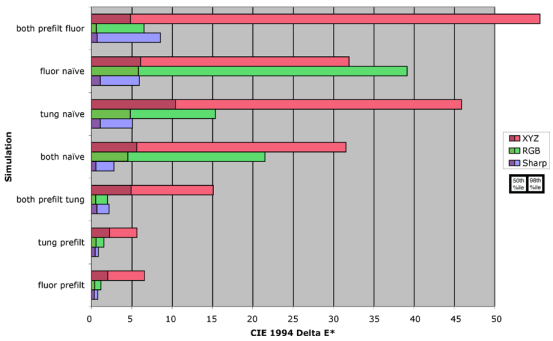


Figure 7: *Error statistics for all solutions and color spaces.*

4. Conclusions

In our experiments, we found spectral prefiltering to minimize color errors in scenes with a single dominant illuminant spectrum, regardless of the rendering color space. The median CIE 1994 Lab $\Delta E*$ values were reduced by a factor of six on average, to levels that were below the detectable threshold when using the *sRGB* and Sharp color spaces. Of the three color spaces we used for rendering, the CIE *XYZ* performed the worst, generating median errors that were above the detectable threshold even with prefiltering, and five times the threshold without prefiltering, meaning the difference was clearly visible over most of the image in side-by-side comparisons to the reference solution. In contrast, the Sharp *RGB* color space, favored by the color science community for chromatic adaptation transforms, performed exceptionally well in a rendering context, producing median error levels that were at or below the detectable threshold both with and without prefiltering.

We believe the Sharp *RGB* space works especially well for rendering by minimizing the representation error for tristimulus values with axes that are aligned along the densest regions of *XYZ* space, perceptually. This property is held in common with the AC_1C_2 color space recommended by Meyer for rendering for this reason[11]. In fact, the AC_1C_2 space has also been favored for chromatic adaptation, indicating the strong connection between rendering calculations and von Kries style transforms. This is evident in the diagonal matrix of Eq. (2), where white point primaries are mul-

tiplied in separate channels, analogous to the color calculations inside a three-component shader. Just as a white point shifts in a von Kries calculation, so do colors shift as they are reflected by a material.

The combination of spectral prefiltering and the Sharp *RGB* space is particularly effective. With prefiltering under a single illuminant, 98% of the pixels were below the detectable error threshold using the Sharp *RGB* space, and only a single highlight in the gold sphere was distinguishable in our side-by-side comparisons. We included a polished gold sphere because we knew its strong spectral selectivity and specularity violated one of our key assumptions, which is that the direct-diffuse component dominates the rendering. We saw in our results that the errors using prefiltering for the gold sphere are no worse than without, and it probably does not matter whether we apply our prefiltering method to specular colors or not, since specular materials tend to reflect other surfaces more than light sources in the final image, anyway. However, rendering in a sharpened *RGB* space always seems to help.

We also tested the performance of prefiltering when we violated our second assumption of a single, dominant illuminant spectrum. When both sources were present and equally bright, the median error was still below the visible threshold using prefiltering in either the *sRGB* or Sharp color space. Without prefiltering, the median jumped significantly for the *sRGB* space, but was still below threshold for Sharp *RGB* rendering. Thus, prefiltering performed no worse on average than the naive approach for mixed illuminants, which was our goal as stated in the introduction.

In conclusion, we have presented an approach to *RGB* rendering that works within any standard framework, adding virtually nothing to the computation time while reducing color difference errors to below the detectable threshold in typical environments. The spectral prefiltering technique accommodates sharp peaks and valleys in the source and reflectance spectra, and user-selection of a dominant illuminant avoids most white balance problems in the output. Rendering in a sharpened *RGB* space also greatly improves color accuracy, independent of prefiltering. Work still needs to be done in the areas of mixed illuminants and colored specular reflections, and we would like to test our method on a greater variety of example scenes.

Acknowledgments

The authors would like to thank Maryann Simmons for providing timely reviews of the paper in progress, and Albert Meltzer for critical editing and LaTeX formatting assistance. We also wish to thank the anonymous reviewers, who we hope will make themselves nonymous at the workshop so we may discuss their points in person, as there was not room to include such discussion in a short paper.

References

1. C. Borges. Trichromatic Approximation for Computer Graphics Illumination Models. *Proc. Siggraph '91*.
2. Anthony J. Calabria and Mark D. Fairchild. Herding CATs: A Comparison of Linear Chromatic-Adaptation Transforms for CIECAM97s. *Proc. 9th Color Imaging Conf.*, pp. 174–178, 2001.
3. Kristin J. Dana, Bram van Ginneken, Shree K. Nayar and Jan J. Koenderink. Reflectance and Texture of Real World Surfaces. *ACM TOG*, **15**(1):1–34, 1999.
4. G. D. Finlayson and P. Morovic. Is the Sharp Adaptation Transform more plausible than CMCCAT2000? *Proc. 9th Color Imaging Conf.*, pp. 310–315, 2001.
5. Andrew S. Glassner. Principles of Digital Image Synthesis. Morgan Kaufmann, 1995.
6. G. W. Larson, H. Rushmeier and C. Piatko. A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes. *IEEE Transactions on Visualization and Computer Graphics*, **3**(4) (December 1997).
7. Laurence T. Maloney. Evaluation of Linear Models of Surface Spectral Reflectance with Small Numbers of Parameters. *J. Optical Society of America A*, **3**(10):1673–1683 (October 1986).
8. David Marimont and Brian Wandell. Linear Models of Surface and Illuminant Spectra. *J. Optical Society of America A*, **9**(11):1905–1913 (November 1992).
9. C. S. McCamy, H. Marcus and J. G. Davidson. A color-rendition chart. *J. Applied Photographic Engineering*, **2**(3):95–99 (summer 1976).
10. R. McDonald and K. J. Smith. CIE94 - a new colur-difference formula. *Soc. Dyers Col.*, **111**:376-9, Dec 1995.
11. Gary Meyer. Wavelength Selection for Synthetic Image Generation. *Computer Vision, Graphics and Image Processing*, **41**:57–79, 1988.
12. Sumanta N. Pattanaik, James A. Ferwerda, Mark D. Fairchild and Donald P. Greenberg. A multiscale model of adaptation and spatial vision for realistic image display. *Proc. Siggraph '98*.
13. Mark S. Peercy. Linear color representations for full speed spectral rendering. *Proc. Siggraph '93*.
14. Brian Smits. An RGB to Spectrum Conversion for Reflectances. *J. Graphics Tools*, **4**(4):11–22, 1999.
15. Michael Stokes et al. A Standard Default Color Space for the Internet – sRGB. Ver. 1.10, November 1996. `http://www.w3.org/Graphics/Color/sRGB`.
16. S. Sueeprasan and R. Luo. Incomplete Chromatic Adaptation under Mixed Illuminations. *Proc. 9th Color Imaging Conf.*, pp. 316–320, 2001.

17. S. Süsstrunk, J. Holm and G. D. Finlayson. Chromatic Adaptation Performance of Different *RGB* Sensors. *IS&T/SPIE Electronic Imaging*, SPIE **4300**, Jan. 2001.

18. Greg Ward et al. Materials and Geometry Format. `http://radsite.lbl.gov/mgf`.

19. Harold B. Westlund and Gary W. Meyer. A BRDF Database Employing the Beard-Maxwell Reflection Model. *Graphics Interface 2002*.

20. Günter Wyszecki and W. S. Stiles. Color Science: Concepts and Methods, Quantitative Data and Formulae. John Wiley & Sons, New York, 2nd ed., 1982.

The LogLuv Encoding for Full Gamut, High Dynamic Range Images

Gregory Ward Larson
Silicon Graphics, Inc.
Mountain View, California

Abstract

The human eye can accommodate luminance in a single view over a range of about 10,000:1 and is capable of distinguishing about 10,000 colors at a given brightness. By comparison, typical computer monitors have a luminance range less than 100:1 and cover about half of the visible color gamut. Despite this difference, most digital image formats are geared to the capabilities of conventional displays, rather than the characteristics of human vision. In this paper, we propose a compact encoding suitable for the transfer, manipulation, and storage of high dynamic range color images. This format is a replacement for conventional RGB images, and encodes color pixels as log luminance values and CIE (u',v') chromaticity coordinates. We have implemented and distributed this encoding as part of the standard TIFF I/O library available by anonymous ftp. After explaining our encoding, we describe its use within TIFF and present some techniques for handling high dynamic range pixels, and demonstrate with an example image.

1. Introduction

Recently, there has been increased interest in high dynamic range (HDR) images, both captured and synthetic [Debevec 97] [Ward 94], which permit extended processing and higher fidelity display methods [Debevec 98] [Larson 97] [Pattanaik 98]. Conventional 24-bit RGB formats cannot encode this additional information, and simple floating point extensions require too much storage space. Some formats, used in the digital film industry, extend the dynamic range slightly using a logarithmic RGB space. Pixar has been using a 33-bit/pixel log format for years, which covers 3.5 orders of magnitude with 0.4% accuracy. Cineon has a 30-bit/pixel log format, but it only covers about 2 orders of magnitude, depending on the type of film being scanned. Another solution, employed in the *Radiance* rendering system, is to append 8 bits to each pixel to represent a common exponent for three 8-bit RGB mantissas [Ward 91]. This provides over 77 orders of magnitude in dynamic range, and works well for the majority of images, but can produce visible quantization artifacts and gamut clamping for some highly saturated colors. This is due to the imperfect separation of luminance and chrominance, and the limited gamut of a non-negative RGB color space.

In this paper, we present a new pixel encoding that uses a log representation of luminance and a CIE (u',v') representation of chrominance.* We call this a *LogLuv* encoding. This encoding has the following desirable properties. It:

- Covers the entire visible color gamut.
- Covers the full range of perceivable luminances (over 38 orders of magnitude).
- Uses imperceptible step sizes in a perceptually uniform space.
- May be calibrated to absolute luminance and color.
- Enables optimal visual fidelity on any output device.

In addition to these inherent features, we have the following technical goals for our format. We would like it to be:

- Compact.
- Compressible.
- Easy to convert to XYZ and RGB formats.
- Incorporated into the TIFF standard.

In this paper, we describe our LogLuv pixel encoding method, followed by a description of our extension to Sam Leffler's free TIFF library. We then discuss some practical considerations, and give an example application to HDR tone mapping, ending with a brief conclusion.

2. Encoding Method

We have actually implemented two LogLuv pixel encodings, a 24-bit encoding and a 32-bit encoding. The 24-bit encoding breaks down into a 10-bit log luminance portion and a 14-bit, indexed uv coordinate mapping. The 32-bit

*Chrominance is the quantity equivalent to hue plus saturation. We follow conventional CIE usage by using "color" and "chrominance" interchangeably, though the latter is properly a subset of the former [Wyszecki82].

encoding uses 16 bits for luminance and 8 bits each for u' and v'. Compared to the 24-bit encoding, the 32-bit version provides greater dynamic range and precision at the cost of an extra byte per pixel. Also, the 32-bit format is simpler and compresses better, such that most 32-bit LogLuv images end up smaller than their 24-bit counterparts. Therefore, we will not discuss the 24-bit encoding in this article, but refer the reader to the original paper for details [Larson 98].

2.1 32-bit LogLuv Pixel Encoding

The 32-bit LogLuv encoding uses 16 bits for luminance information and 16 bits for chrominance. The MSB is used to flag negative luminances, and the next 15 bits record up to 38 orders of magnitude in 0.27% relative steps, covering the full range of perceivable world luminances in imperceptible steps. The lower two bytes encode u' and v', respectively. The bit breakdown is shown in Fig. 1.

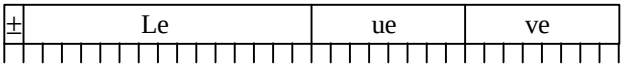


Figure 1. Bit allocation for 32-bit pixel encoding. MSB is a sign bit, and the next 15 bits are used for a log luminance encoding. The uv coordinates are separate 8-bit quantities.

The conversion to and from our log luminance encoding is given in Eq. 1. The maximum luminance using this encoding is 1.84×10^{19}, and the smallest magnitude is 5.44×10^{-20}. An L_e value of 0 is taken to be exactly 0.0. The sign bit is extracted before encoding and reapplied after the conversion back to real luminance.

$$L_e = \lfloor 256(\log_2 Y + 64) \rfloor \qquad (1a)$$

$$Y = \exp_2\left[\left(L_e + 0.5\right)/256 - 64\right] \qquad (1b)$$

Since the gamut of perceivable u and v values is between 0 and 0.62, we chose a scale factor of 410 to go between our [0,255] integer range and real coordinates, as given in Eq. 2.

$$u_e = \lfloor 410u' \rfloor \qquad (2a)$$

$$v_e = \lfloor 410v' \rfloor \qquad (2b)$$

$$u' = \left(u_e + 0.5\right)/410 \qquad (2c)$$

$$v' = \left(v_e + 0.5\right)/410 \qquad (2d)$$

This encoding captures the full color gamut in 8 bits each for u_e and v_e. There will be some unused codes outside the visible gamut, but the tolerance this gives us of 0.0017 units in uv space is already well below the visible threshold. Conversions to and from 1931 CIE (x,y) chromaticities are given in Eqs. 3 and 4.

$$u' = \frac{4x}{-2x+12y+3} \qquad (3a)$$

$$v' = \frac{9y}{-2x+12y+3} \qquad (3b)$$

$$x = \frac{9u'}{6u'-16v'+12} \qquad (4a)$$

$$y = \frac{4v'}{6u'-16v'+12} \qquad (4b)$$

where:

$x = X/(X+Y+Z)$

$y = Y/(X+Y+Z)$

3. TIFF Input/Output Library

The LogLuv encoding described has been embedded as a new `SGILOG` compression type in Sam Leffler's popular TIFF I/O library. This library is freely distributed by anonymous ftp at the site given at the end of this article.

When writing a high dynamic range TIFF image, the LogLuv *codec* (compression/decompresson module) takes floating point CIE XYZ scanlines and writes out 24-bit or 32-bit compressed LogLuv-encoded values. When reading an HDR TIFF, the reverse conversion is performed to get back floating point XYZ values. (We also provide a simple conversion to 24-bit gamma-compressed RGB for the convenience of readers that do not know how to handle HDR pixels.)

An additional tag is provided for absolute luminance calibration, named `TIFFTAG_STONITS`.* This is a single floating point value that may be used to convert Y values returned by the reader to absolute luminance in candelas per square meter. This tag is set by the application that writes out a HDR TIFF to permit prescaling of values to a reasonable brightness range for display, where values of 1.0 will be displayed at the maximum output of the destination device. This avoids the image reader having to figure out a good exposure level for absolute luminances. If the input data is uncalibrated (i.e., the absolute luminances are unknown), then there is no need to store this tag, whether the values are scaled or not.

3.1 Run-length Compression

By separating the bytes into four streams on each scanline, the 32-bit encoding can be efficiently compressed using an adaptive run-length encoding scheme [Glassner 91]. Since the top byte containing the sign bit and upper 7 log luminance bits changes very slowly, this byte-stream submits very well to run-length encoding. Likewise, the encoded u_e and v_e byte-streams compress well over areas of constant chrominance.

* STONITS stands for "samples to Nits," where "Nits" is the photometric unit for luminance, also written candelas/meter^2. Use of this tag will be discussed later in section 5.2.

3.2 Grayscale Images

For maximum flexibility, a pure luminance mode is also provided by the codec, which stores and retrieves run-length encoded 16-bit log luminance values using the same scheme as applied in the 32-bit LogLuv encoding. There is no real space savings over a straight 32-bit encoding, since the u_e and v_e byte-streams compress to practically nothing for grayscale data, but this option provides an explicit way to specify floating point luminance images for TIFF readers that care.

3.3 Raw I/O

It is also possible to decode the raw 32-bit LogLuv data retrieved from an HDR TIFF directly, and this has some advantages for implementing fast tone mapping and display algorithms. For the 32-bit format, one can simply multiply the output of a 32 Kentry L_e table and a 64 Kentry uv table to get a tone-mapped and gamma-compressed RGB result, provided the tone mapping algorithm can separate chrominance from luminance. A explanation of how this is done is given in Section 5.1.

3.4 Example TIFF Code and Images

Use of the LogLuv encoding is demonstrated and sample images are provided on our web site, which is given at the end of this article. A converter has been written to and from the *Radiance* floating point picture format [Ward 94] [Ward 91], and serves as an example of LogLuv codec usage. The web site itself also offers programming tips and example code segments.

Example TIFF images using the 32-bit LogLuv and 16-bit LogL encoding are provided on the web site. These images are either scanned from photographic negatives or rendered using *Radiance* and converted to the new TIFF format. Some images are rendered as 360° QuickTime VR panoramas suitable for experiments in HDR virtual reality.

4. Practical Considerations

There are several important considerations in applying the LogLuv image format, which we describe in this section. Issues such as conversion to and from RGB space require thought about dynamic range and gamut limitations, and calibrated versus uncalibrated color spaces. We also discuss speed and space efficiency issues, and appropriate filtering and image manipulation techniques.

4.1 Converting from RGB to XYZ

Ideally, the original image data is available in a real, XYZ color space or a spectrally sampled color space that can be converted to XYZ using standard CIE techniques [Wyszecki 82]. However, most imaging devices and renderers are based on some RGB standard. To get from RGB into XYZ space, a simple 3×3 matrix transformation may be computed from the CIE (x,y) chromaticities for the three RGB primaries and a white point [Rogers 85]. Using the standard CCIR 709 RGB primaries for computer displays and a neutral white point for optimal color balance, we derive the color transformation shown in Eq. 5.

	R	G	B
x	0.640	0.300	0.150
y	0.330	0.600	0.060

Table 1. CIE (x,y) chromaticities for CCIR 709 RGB primaries.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.497 & 0.339 & 0.164 \\ 0.256 & 0.678 & 0.066 \\ 0.023 & 0.113 & 0.864 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (5)$$

4.2 Luminance Calibration

If the image corresponds to a luminous scene (as opposed to a painting or other reflective media), it may be possible to calibrate the recorded information using the `TIFFTAG_STONITS` field mentioned earlier. This is a real multiplier that is stored with the image by whoever creates it. When reading an image, an application can retrieve this value and multiply it by each pixel's Y value to get an absolute luminance in cd/m^2. If the absolute luminance for a given pixel or subimage is known by the TIFF writer, this multiplier is also known, since it equals the absolute luminance divided by the output Y value. If, on the other hand, we are working from a scan of a photographic negative or transparency, it may be possible to approximate this multiplier from the image exposure time, f-stop and film speed. For 35mm photography, we can use the formula given in Eq. 6, borrowed from the IES Handbook [IES 93].

$$m = \frac{200}{\pi S} \cdot f^2 / t \quad (6)$$

where:

S = film speed (ISO ASA)
f = aperture (f-stop)
t = exposure time (seconds)

Eq. 6 assumes that the maximum image brightness corresponds to a Y value of 1.0. For photographic negatives, which hold about 2 orders of magnitude of extended dynamic range beyond white, we assume this "maximum" corresponds to a density of about 1.5 (3% negative transmittance).

4.3 Converting back to RGB

For the reverse conversion, we invert the matrix from Eq. 5 as shown in Eq. 7.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 2.690 & -1.276 & -0.414 \\ -1.022 & 1.978 & 0.044 \\ 0.061 & -0.224 & 1.163 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (7)$$

Since some of the matrix coefficients are negative due to the larger gamut of the imaginary CIE primaries, the formula given in Eq. 7 may result in negative RGB values for some highly saturated colors. If the image processing software can cope with negative values, it is better to leave them that way, otherwise a gamut mapping operation may be performed to bring them back into the legal range. Much

research has been devoted to this problem [Stone 88], but clamping is the most often applied solution. We have also found desaturating to a face of the RGB cube to be a simple and effective solution to limiting the color gamut.

If floating point RGB colors are maintained without gamut limiting (i.e., negative values allowed), it is possible to go from LogLuv→RGB→LogLuv without losing data. However, the opposite is not true, since the LogLuv encoding quantizes data into perceptual bins. Even in the case of integer RGB data, there will be some differences due to the binning used by the two encodings.

4.4 RGB→LogLuv→RGB Information Loss

Even though the gamut and dynamic range of the 32-bit LogLuv format is superior to that of 24-bit RGB, we will not get the exact pixel values again if we go through this format and back to RGB. This is because the quantization size of a LogLuv pixel is matched to human perception, whereas 24-bit RGB is not. In places where human error tolerance is greater, the LogLuv encoding will have larger quantization volumes than RGB and therefore may not reproduce exactly the same 24-bit values when going back and forth. Over most of the gamut, the LogLuv tolerances will be tighter, and RGB will represent lower resolution information. (This is especially true for dark areas.) In other words, the losses incurred going through the LogLuv format may be measurable in absolute terms, but they should not be visible to a human observer since they are below the threshold of perception.

We performed two tests to study the effects of going between RGB and LogLuv formats, one quantitative test and one qualitative test. In the quantitative test, we went through all 16.7 million 24-bit RGB colors and converted to 32-bit LogLuv and back, then measured the difference between the input and output RGB colors using the CIE E* perceptual error metric. We found that 17% of the colors were translated exactly, 80% were below the detectable threshold and 99.75% were less than twice the threshold, where differences may become noticeable. In our qualitative test, we examined a dozen or so images from different sources, performing the RGB→LogLuv→RGB mapping in interleaved 16 scan-line stripes that roughly corresponded to the maximum visible angular frequency. In this way, we hoped to notice the underlying pattern in cases where the translation resulted in visible differences. In all of the captured images we looked at, the details present completely obscured any differences caused by the translation, even in sky regions that were relatively smooth. Only in one synthesized image, which we designed very carefully to have smooth gradients for our tests, were we just barely able to discern the pattern in large, low detail regions. Even so, we had to really be looking for it to see it, and it sort of faded in and out, as if the eye could not decide if it was really there or not.

From these tests, we concluded that the differences caused by taking RGB data through the LogLuv format will not be visible in side-by-side comparisons.

4.5 LogLuv Image Compression

The simple adaptive run-length encoding scheme used to compress our 32-bit/pixel LogLuv format performs about as well on average as the LZW scheme used on standard TIFF RGB images. Since luminance and chrominance are placed in separate byte streams, regions where one or the other are relatively constant compress very well. This is often the case in computer graphics imagery, which will compress between 15% and 60% for reasonably complex scenes, and will often outperform LZW compression so much that the resulting file size is actually smaller than the LZW-compressed 24-bit RGB version. For scanned imagery, the performance is usually not as good using run-length encoding due to the increased complexity and image noise. However, we never grow the data, as can happen with the LZW compression algorithm, which is sometimes worse than no compression at all.

For example, compression performance for the scene shown in Fig. 2 is -8% for LZW compression, compared to 13% for our run-length encoding. The LZW file still ends up slightly smaller since it is starting from 24-bit pixels, but our compressed result has an additional advantage, which is lower entropy. Specifically, we can apply an entropy encoding scheme to our run-length encoded result, and pick up some additional savings. For the same figure, applying the gzip program to the result gains an additional 17% compression, making the resulting file smaller than the straight LZW encoding. (Applying gzip to the LZW file only reduces its size by 4%, which is not enough to make up the difference.) In most cases, applying gzip to our run-length encoded 32-bit LogLuv TIFF yields a file that is within a 10% of a gzip'ped 24-bit RGB file, which is almost always smaller than TIFF's blockwise LZW compression.

4.6 Image Processing

There are two basic approaches for processing LogLuv encoded images. One method is to convert to a floating-point RGB or XYZ color space and perform image manipulations on these quantities. This is probably the simplest and most convenient for general purposes. The other method is to work directly in a separate luminance space such as Yuv or Yxy. For certain types of image processing, such as contrast and brightness adjustment, it may actually be faster since we can work on the Y channel alone, plus we save a little time on our conversions.

In general, it is not a good idea to convert to an 8-bit/primary integer space to manipulate high dynamic range images, because the dynamic range is lost in the process. However, since most existing hardware and software is geared to work with 24-bit RGB data, it may be easiest to use this format for interactive feedback, then apply the manipulation sequence to the floating point data when writing the file. Better still, one could use the approach adopted by some software packages and store the sequence of image processing commands together with the original data. These commands may be recorded as additional TIFF tags and attached to the file without even touching the data. Besides saving time, this approach also preserves our absolute luminance calibration, if `TIFFTAG_STONITS` is present.

Compositing high dynamic range images using an alpha channel is also possible, though one must consider the effect of multiplying a floating point value by an integer value. When one is small and the other is large, the result may either be wrong or show severe quantization artifacts. Therefore, it is best to use a high dynamic range alpha

channel, which may be stored as a separate 16-bit LogL TIFF layer.

An additional benefit of the 32-bit LogLuv and 16-bit LogL formats is that pixels may take on negative values, which are useful for arbitrary image math and general masking and filtering. For example, an image may be broken into separate 2D Fourier frequency layers, which when summed together yield back the original image. The coefficients for these layers may then be tuned independently or manipulated in arbitrary ways. This is usually done in memory, such that the written file format pays little part, but being able to write out the intermediate pixel data without losing information has potential advantages for subsequent processing.

5. Motivating Application: Tone Mapping

Tone mapping is the process of mapping captured image values to displayed or printed colors [Tumblin93]. With the added dynamic range and gamut provided by our new encoding, we can get a much better mapping than we could with the limited range of conventional digital images. If our data is also calibrated (i.e., `STONITS` has been set by the creating application), then we can go further to simulate visibility using what we know about human color and contrast sensitivity. Recently, a number of tone mapping techniques have been developed for images with high dynamic range [Pattanaik 98] [Spencer 95] [Chiu 93] [Jobson 97]. Most of these methods introduce image-geometric dependencies, so they cannot be applied independently to each pixel, or computed using integer math. In this section, we present an efficient tone-mapping technique for HDR images that can be applied on a pixel-by-pixel basis using integer operations [Larson97].

5.1 Efficient Tone Mapping

As we mentioned earlier in Section 3.3, we can go directly from the luminance and uv image data to an RGB tone-mapped result by multiplying the output of two lookup tables. To get the math to work out right, both the luminance and uv table values must be in the monitor's gamma-compressed space, and the RGB values must be premultiplied by their corresponding luminance coefficients. The computation for luminance and RGB table entries corresponding to specific tone-mapped display colors is given in Eq. 8.

$$L_t(L_e) = \lfloor 256\, L_d(L_e)^{1/g} \rfloor \qquad (8a)$$

$$R_t(C_e) = \left\lfloor 256\left(S_r R_1(C_e)\right)^{1/g} \right\rfloor \qquad (8b)$$

$$G_t(C_e) = \left\lfloor 256\left(S_g G_1(C_e)\right)^{1/g} \right\rfloor \qquad (8c)$$

$$B_t(C_e) = \left\lfloor 256\left(S_b B_1(C_e)\right)^{1/g} \right\rfloor \qquad (8d)$$

where:

L_t, R_t, G_t, B_t	= lookup table values
L_e	= encoded log luminance
C_e	= encoded uv color
$L_d(L_e)$	= mapped display luminance
R_1, G_1, B_1	= normalized color mapping
γ	= monitor response gamma
S_r, S_g, S_b	= red, green, blue coefficients from middle row of Eq. 5 matrix

$S_r \times R_1(C_e) + S_g \times G_1(C_e) + S_b \times B_1(C_e) = 1$

The color coefficients guarantee that all tabulated chrominance values will be between 0 and 255. To get from the separately tabulated luminance and chrominance to display values in the 0-255 range, we apply the formulas given in Eq. 9.

$$R_d = L_t(L_e)\, R_t(C_e) / \lfloor 256\, S_r^{1/g} \rfloor \qquad (9a)$$

$$G_d = L_t(L_e)\, G_t(C_e) / \lfloor 256\, S_g^{1/g} \rfloor \qquad (9b)$$

$$B_d = L_t(L_e)\, B_t(C_e) / \lfloor 256\, S_b^{1/g} \rfloor \qquad (9c)$$

Since the denominators in Eq. 9 are constant, they can be precomputed, leaving only four table lookups, three integer multiplies and three integer divides to map each pixel.

We have implemented this type of integer-math tone-mapping algorithm in an HDR image viewer, and it takes less than a second to convert and display a 512×512 picture on a 180 MHz processor. The L_e table size is determined by the range of luminances present in the image, and only the colors needed are actually translated to RGB and stored in the uv lookup table. We used the high dynamic range operator described in [Larson 97], which requires a modification to our algorithm to include mesopic color correction, since it relates the luminance and color mappings. For pixels below the upper mesopic limit, we determine the color shift in uv coordinates, then do our table lookup on the result.

5.2 Example Results

Fig. 2a shows a scanned photograph as it might appear on a PhotoCD using a YCC encoding. Since YCC can capture up to "200% reflectance," we can apply a tone mapping operator to bring this extra dynamic range into our print, as shown in Fig. 3a. However, since many parts of the image were brighter than this 200% value, we still lose much of the sky and circumsolar region, and even the lighter asphalt in the foreground. In Fig. 2b, we see where 35% of the original pixels are outside the gamut of a YCC encoding.

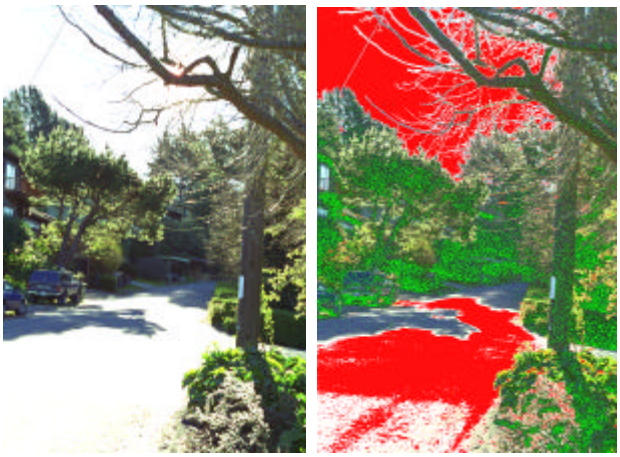


Figure 2. The left image (a) shows a PhotoYCC encoding of a color photograph tone-mapped with a linear operator. The right image (b) shows the out-of-gamut regions. Red areas are too bright or too dim, and green areas have inaccurate color.

Fig. 3b shows the same color negative scanned into our 32-bit/pixel high dynamic range TIFF format and tone mapped using a histogram compression technique [Larson 97]. Fig. 4c shows the same HDR TIFF remapped using the perceptual model of Pattanaik et al [Pattanaik 98]. Figs. 4a and 4b show details of light and dark areas of the HDR image whose exposure has been adjusted to show the detail captured in the original negative. Without an HDR encoding, this information would be lost.

Figure 3. The left image(a) shows the YCC encoding after remapping with a histogram compression tone operator. Unfortunately, since YCC has so little dynamic range, most of the bright areas are lost. The right image (b) shows the same operator applied to a 32-bit HDR TIFF encoding, showing the full dynamic range of the negative.

Figure 4. The upper-left image (a) shows the circumsolar region reduced by 4 f-stops to show the image detail recorded on the negative. The lower-left image (b) shows house details boosted by 3 f-stops. The right image (c) shows our HDR TIFF mapped with the Pattanaik-Ferwerda tone operator.

5.3 Discussion and Other Applications

It is clear from our example that current methods for tone-mapping HDR imagery, although better than a simple S-curve, are less than perfect. It would therefore be a mistake to store an image that has been irreversibly tone mapped in this fashion, as some film scanner software attempts to do. Storing an HDR image allows us to take full advantage of future improvements in tone mapping and display algorithms, at a nominal cost.

Besides professional film and photography, there are a number of application areas where HDR images are key. One is lighting simulation, where designers need to see an interior or exterior space as it would really appear, and evaluate things in terms of absolute luminance and illuminance levels. Since an HDR image can store the real luminance in its full-gamut coverage, this information is readily accessible to the designer. Another application is image-based rendering, where a user is allowed to move about in a scene by warping captured or rendered images [Debevec 96]. If these images have limited dynamic range, it is next to impossible to adapt the exposure based on the current view, and impossible to use the image data for local lighting. Using HDR pixels, a natural view can be provided for any portion of the scene, and the images may also be used to illuminate local objects [Debevec 98]. A fourth application area is digital archiving, where we are making a high-quality facsimile of a work of art for posterity. In this case, the pixels we record are precious, so we want to make sure they contain as much information as possible. At the same time, we have concerns about storage space and transmission costs, so keeping this data as compact as possible is important. Since our HDR format requires little more space than a standard 24-bit encoding to capture the full visible gamut, it is a clear winner for archiving applications.

Our essential argument is that we can make better use of the bits in each pixel by adopting a perceptual encoding of color and brightness. Although we don't know how a given image might be used or displayed in the future, we do know something about what a human can observe in a given scene.

By faithfully recording this information, we ensure that our image will take full advantage of any future improvements in imaging technology, and our basic format will continue to find new uses.

6. Conclusion

We have presented a new method for encoding high dynamic range digital images using log luminance and uv chromaticity to capture the entire visible range of color and brightness. The proposed format requires little additional storage per pixel, while providing significant benefits to suppliers, caretakers and consumers of digital imagery.

This format is most appropriate for recording the output of computer graphics rendering systems and images captured from film, scanners and digital photography. Virtually any image meant for human consumption may be better represented with a perceptual encoding. Such encodings are less appropriate when we either wish to store non-visual (or extravisual) information as found in satellite imagery, or output to a specific device with known characteristics such as NTSC or PAL video.

Through the use of re-exposure and dynamic range compression, we have been able to show some of the benefits of HDR imagery. However, it is more difficult to illustrate the benefits of a larger color gamut without carefully comparing hard copy output of various multi-ink printers. Also, since we currently lack the ability to capture highly saturated scenes, our examples would have to be contrived from individual spectral measurements and hypothetical scenes. We therefore leave gamut validation as a future exercise.

Future work on the format itself should focus on the application of lossy compression methods for HDR images and animations. A JPEG-like cosine transform should work very well, since LogLuv looks almost the same perceptually as the gamma-compressed YCbCr coordinates that JPEG uses. Efficient compression is needed for broad acceptance of HDR imagery and extensions for digital cinema.

7. Acknowledgments

The author would like to thank Sam Leffler for his help and cooperation in adding these routines to the TIFF i/o library, Dan Baum for his support and encouragement, Paul Haeberli, Christopher Walker and Sabine Susstrunk for their advice, and Alexander Wilkie, Philippe Bekaert and Jack Tumblin for testing the software. Finally, thanks to Ronen Barzel for his many helpful comments and suggestions on the paper itself.

8. References

[Chiu 93] K. Chiu, M. Herf, P. Shirley, S. Swamy, C. Wang and K. Zimmerman "Spatially nonuniform scaling functions for high contrast images," *Proceedings of Graphics Interface '93*, Toronto, Canada, (May 1993).

[Debevec 98] Paul Debevec, "Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography," *Computer Graphics (Proceedings of ACM Siggraph 98)*.

[Debevec 97] Paul Debevec, Jitendra Malik, "Recovering High Dynamic Range Radiance Maps from Photographs," *Computer Graphics (Proceedings of ACM Siggraph 97)*.

[Debevec 96] Paul Debevec, Camillo Taylor, Jitendra Malik, "Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach," *Computer Graphics (Proceedings of ACM Siggraph 96)*.

[Glassner 91] Andrew Glassner, "Adaptive Run-Length Encoding," in *Graphics Gems II*, edited by James Arvo, Academic Press, (1991).

[IES 93] Illuminating Engineering Society of North America, *IES Lighting Handbook, Reference Volume*, IESNA (1993).

[Jobson 97] Daniel Jobson, Zia-ur Rahman, and Glenn A. Woodell. "Properties and Performance of a Center/Surround Retinex," *IEEE Transactions on Image Processing*, Vol. 6, No. 3 (March 1997).

[Larson 98] Greg Larson, "Overcoming Gamut and Dynamic Range Limitations in Digital Images," Color Imaging Conference, Scottsdale, Arizona (1998).

[Larson 97] Greg Larson, Holly Rushmeier, Christine Piatko, "A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes," *IEEE Transactions on Visualization and Computer Graphics*, **3**, 4, (1997).

[Pattanaik 98] Sumant Pattanaik, James Ferwerda, Mark Fairchild, Don Greenberg, "A Multiscale Model of Adaptation and Spatial Vision for Realistic Image Display," *Computer Graphics (Proceedings of Siggraph 98)*.

[Rogers 85] David Rogers, *Procedural Elements for Computer Graphics*, McGraw-Hill, (1985).

[Spencer 95] G. Spencer, P. Shirley, K. Zimmerman, and D. Greenberg, "Physically-based glare effects for computer generated images," *Computer Graphics* (*Proceedings of Siggraph 95*).

[Stone 88] Maureen Stone, William Cowan, John Beatty, "Color Gamut Mapping and the Printing of Digital Color Images," *ACM Transactions on Graphics*, 7(3):249-292, (October 1988).

[Tumblin93] Tumblin, Jack and Holly Rushmeier. "Tone Reproduction for Realistic Images," *IEEE Computer Graphics and Applications*, November 1993, 13(6).

[Ward 94] Greg Ward, "The RADIANCE Lighting Simulation and Rendering System," *Computer Graphics (Proceedings of Siggraph 94)*.

[Ward 91] Greg Ward, "Real Pixels," in *Graphics Gems II*, edited by James Arvo, Academic Press, (1991).

[Wyszecki 82] Gunter Wyszecki, W.S. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae*, Second Edition, Wiley, (1982).

9. Web Information

Links to Sam Leffler's TIFF library, example LogLuv images and other information may be found at: *http://www.acm.org/jgt/papers/Larson99*

Greg may be reached by e-mail at *gregl@sgi.com*.