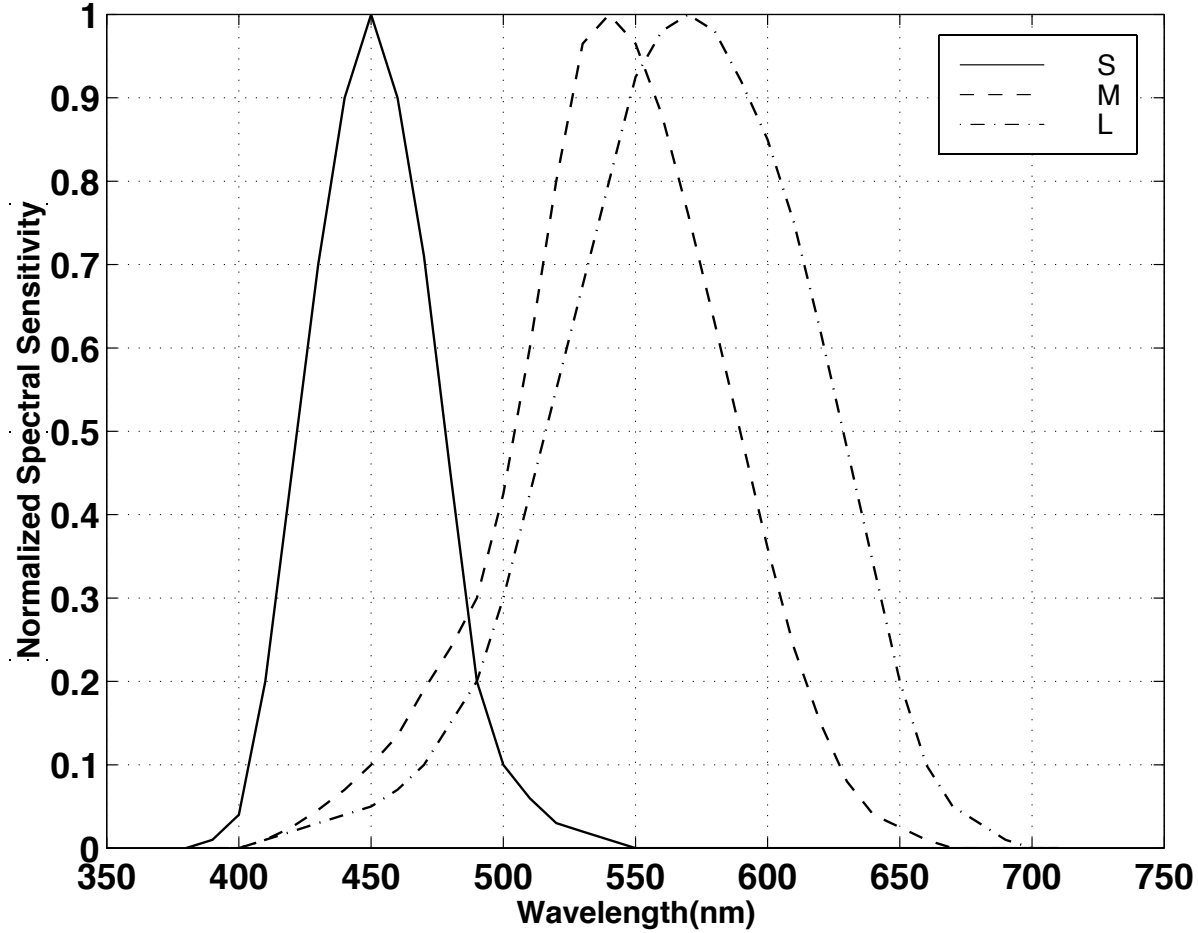
Lecture Notes 11
Introduction to Color Imaging

- Color filter options
- Color processing
- Color interpolation (demozaicing)
- White balancing
- Color correction

Preliminaries

- Up till now we have been only discussing gray scale image capture
  - If the incident photon flux density at a pixel is $f_0(\lambda)$ ph/cm^3.s, for $400 \leq \lambda \leq 700$ nm, then the resulting photocurrent density
    $$j_{ph} = q \int f_0(\lambda) \mathrm{QE}(\lambda) d\lambda \text{ A/cm}^2,$$
    where $\mathrm{QE}(\lambda)$ e-/ph is the photodetector QE, which is a function of the technology parameters
  - Assuming constant j_{ph} over pixel area and over time (and ignoring dark current and noise) we get a pixel output (voltage)
    $$v_o \propto \int f_0(\lambda) \mathrm{QE}(\lambda) d\lambda$$
- To capture color images, each pixel must output more information about the spectral distribution of the incident photon flux ($f_0(\lambda)$)
- A key fact from *color science* is that we do not need to completely know the incident photon flux spectral distribution to faithfully reproduce color – in fact only *three values* per pixel can be sufficient

- Reason: the human eye has three types of photodetectors (*cones*) L, M, and S with different spectral sensitivities



- So under uniform illumination (photon flux density $f_0(\lambda)$) the color we see can be represented by a 3-dimensional vector

$$\mathbf{C} = \begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} \int f_0(\lambda) l(\lambda) d\lambda \\ \int f_0(\lambda) m(\lambda) d\lambda \\ \int f_0(\lambda) s(\lambda) d\lambda \end{bmatrix}$$

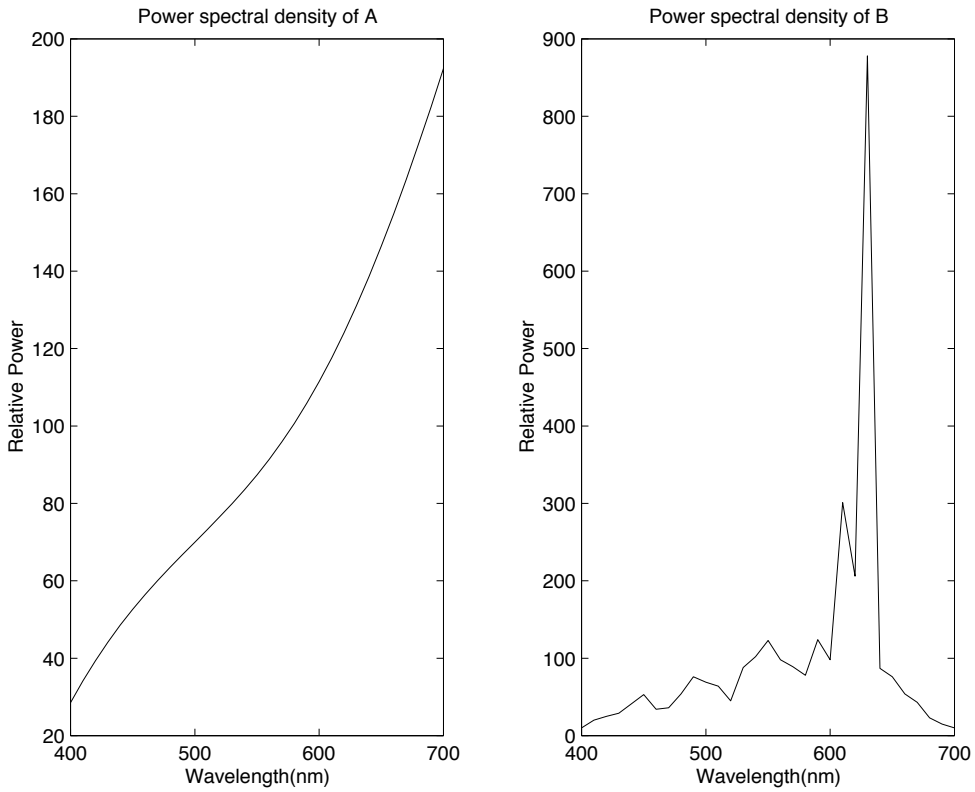
Or using discrete λ values as

$$\mathbf{C} = \begin{bmatrix} \\ \mathbf{l}^T(\lambda) \\ \mathbf{m}^T(\lambda) \\ \mathbf{s}^T(\lambda) \\ \\ \end{bmatrix} \begin{bmatrix} \\ \mathbf{F_0}(\lambda) \\ \\ \\ \\ \end{bmatrix}$$

Thus $\mathbf{C}$ can be expressed as a linear combination of three *basis vectors*

$$\mathbf{C} = L \cdot \mathbf{l} + M \cdot \mathbf{m} + S \cdot \mathbf{s}$$

- Note: photon flux densities with different spectral distributions can produce the same perceived color (these are called *metamers*), e.g.,



- The color basis vectors are not unique — we can use different basis vectors to represent color, e.g., RGB (but we must be careful in selecting the spectral responses for the basis vectors), so for example we can write

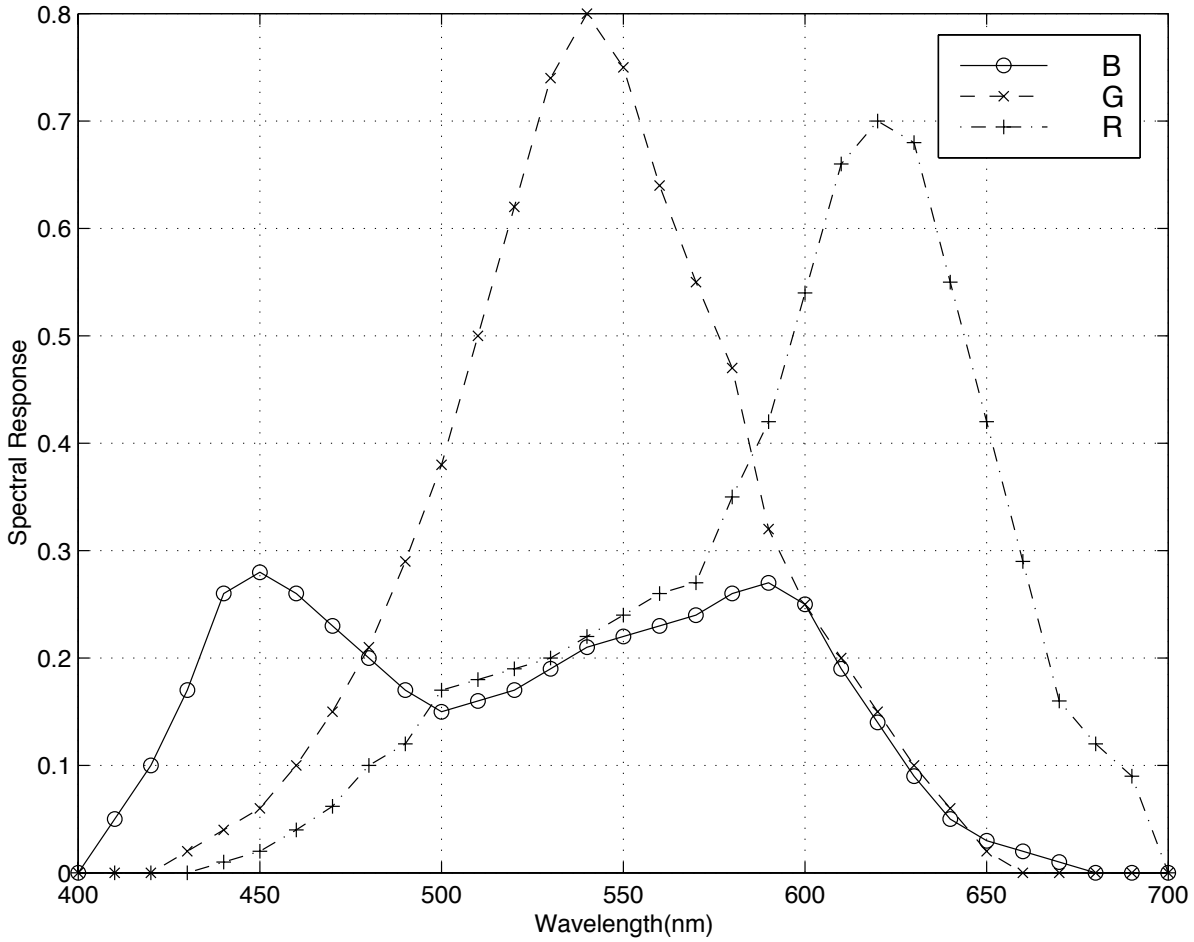
$$\mathbf{C} = R \cdot \mathbf{r} + G \cdot \mathbf{g} + B \cdot \mathbf{b}$$

- $\mathbf{C}$ can be transformed from one basis vector representation to another (or from one color space to another) using a 3×3 matrix (more on this later)
- To get the three values from a pixel, color filters with different spectral responses are used, e.g., R, G, B filters
- So if we denote the R filter response by $\phi_R(\lambda)$, the R output from a pixel with photon flux density $f_0(\lambda)$ is

$$v_{oR} \propto \int f_0(\lambda)\eta(\lambda)\phi_R(\lambda)d\lambda$$

  and similarly for the other filters
- The "camera" RGB spectral responses are the products of each filter's response and the photodetector spectral response, i.e., $\phi_R(\lambda)\eta(\lambda)$, ... etc.

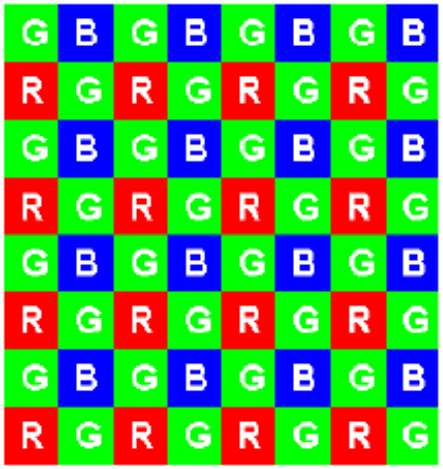
- Example: RGB spectral responses for a Kodak digital camera



Color filter options

- Use three image sensors and a beam splitter (prism)
  - + Every photon finds its way to a sensor
  - + High spatial resolution
  - – High cost, nonoverlapping color filter spectra not desirable
- Use time-switched color filter
  - + High spatial resolution, each color can have different exposure time
  - – Longer exposure time — motion blur can be a problem
  - – Optical loss due to filters, high cost (rarely used)
- Use color filter array (CFA) or mosaic deposited on top of the pixel array, so each pixel outputs only one color component, *e.g.*, R, G, or B
  - + Lowest cost option
  - – Lower spatial resolution, optical loss due to filters
  - – Processing (demozaicing) needed to reconstruct the missing color components for each pixel

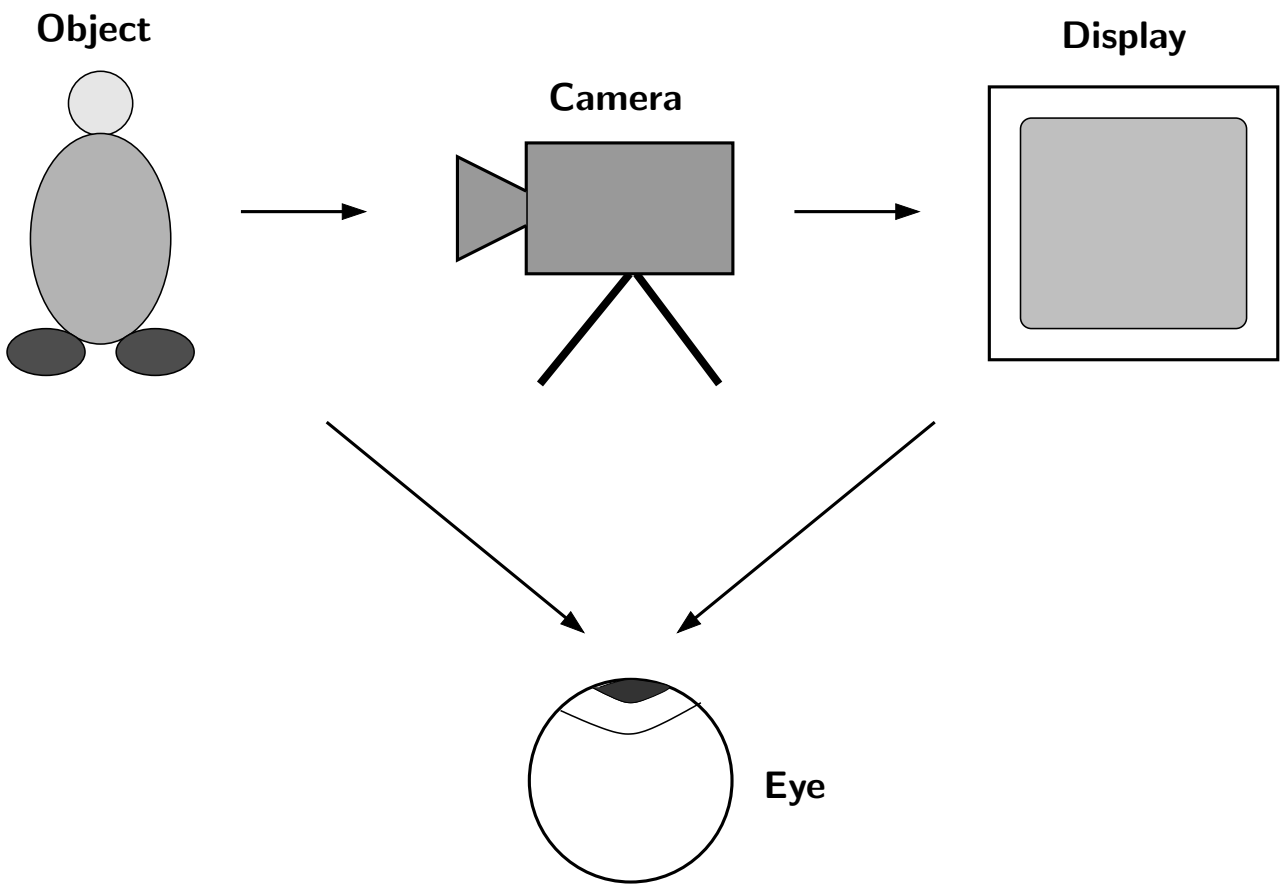
Color Filter Arrays



Bayer

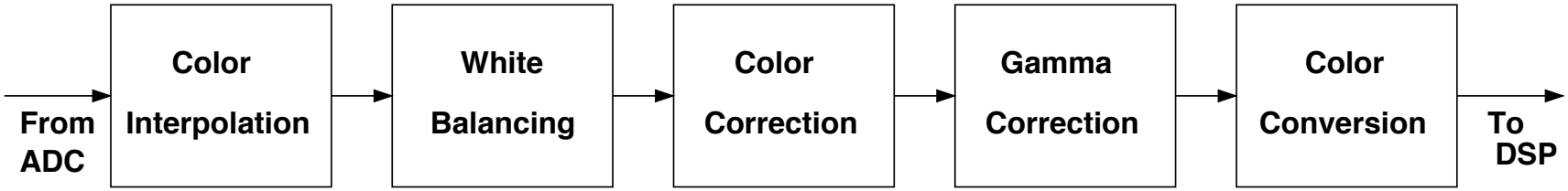
Stripe

Color Processing



- Color processing is needed to (i) reconstruct missing pixel color components and (ii) to produce color (on a display device) that is close to what the eye would perceive

- Typical color processing steps in a digital camera



  - **White balance:** used to adjust for illuminant so that, for example, a white background appears white (the eye does this adaptively)
  - **Color correction:** transforms the camera output to the color space of the display, or to a standard color space
  - **Gamma correction:** corrects for display nonlinearity, also needed before image processing/compression
  - **Color conversion:** needed before image processing/compression

- Color processing is performed mostly in the digital domain (but sometimes in analog, *e.g.*, white balancing)
- It is computationally very demanding (about 70% of processing in a digital camera is related to color)

Color Interpolation (Demozaicing)

- Used to reconstruct the missing pixel color components (when a CFA is used)
- Interpolation method must
  - Reduce artifacts such as aliasing and color fringing (false colors)
  - Have reasonable computational complexity

Interpolation algorithms:

- Nearest neighbor replication
  - To reconstruct a missing color component of a pixel, simply set it equal to the value of its nearest pixel with that color
  - Simple and fast, but results in large artifacts especially at edges
- Bilinear interpolation
  - Perform bilinear interpolation in each color plane

- relatively simple, but still suffers from some edge artifacts (may not be visible in a video sequence)

- 2-D filtering
  - This is a generalization of bilinear interpolation
  - The filter window size and coefficients for each color plane are designed to reduce artifacts
  - Artifacts will still exist around edges

- Adaptive algorithms
  - Since most artifacts occur around edges, change (adapt) the interpolation method when edges are present
  - Yields better performance but requires more computations (for edge detection)

Interpolation Algorithms — Examples

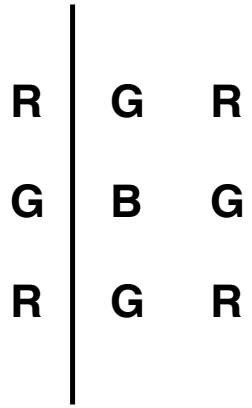
- Consider the Bayer pattern

B1	**G2**	**B3**	**G**
G4	**R5**	**G6**	**R**
B7	**G8**	**B9**	**G**
G	**R**	**G**	**R**

- Bilinear interpolation
  - $G5 = \frac{G2+G4+G6+G8}{4}$
  - $B5 = \frac{B1+B3+B7+B9}{4}$
  - $B2 = \frac{B1+B3}{2}$
- Adaptive algorithm
  - interpolation results in color fringing and zipper effects along edges — most significant for luminance (green), since the eye is more

sensitive to spatial variation in luminance than chrominance

- For each pixel (with missing green) perform edge detection before interpolation, and only use pixels *along* edges
- For example, assume that the pixels to the left of the edge have larger pixel values than the ones on the right



Instead of using the four greens to estimate the missing green value of the blue pixel, which would result in color fringing, we only use the two greens along the edge

What if the edge is diagonal?

Use larger region for interpolation . . .

White Balancing

- Different light sources (illuminants) have different power spectral densities (psd)
- The psd of color reflected from an object is a function of both the object's surface reflectance and the illuminant psd

  more specifically the photon flux density at a pixel is proportional to the product of the object surface reflectance $S(\lambda)$ and the illuminant psd $E(\lambda)$, i.e., $f_0(\lambda) \propto E(\lambda)S(\lambda)$
- So, for example, a raw image taken by a camera of a white piece of paper will look yellowish under incandescent lighting and greenish under fluorescent lighting compared to under day light
- The eye, by comparison, would see the white paper as white almost independent of the illuminant, and in a scene with white background it adjusts the colors in the scene so that the background looks white
- Captured images must also be processed so that a white background looks white — this is called white (color) balancing

Two Approaches to White Balancing

- *Fixed* white balance, i.e., with known illuminant
  - Capture images of a white piece of paper under each potential illuminant (the first illuminant being the standard one where the image looks white), for each illuminant
    - Compute the average value for each color channel (R_i, G_i, B_i)
    - Compute the ratio between each color channel and the green channel (luminance), i.e., $\frac{R_i}{G_i}$ and $\frac{B_i}{G_i}$
    - Normalize each ratio by the corresponding ratio of the first illuminant to get $\left(\frac{R_i}{G_i}\right) / \left(\frac{R_1}{G_1}\right)$ ...
  - To perform white balancing for a captured image with known illuminant, divide the red and blue values by the appropriate normalized ratios

- *Automatic* white balance is used if we do not know the illuminant
  - Most algorithms used in cameras are proprietary, most use some variation of the Gray World assumption
  - The *Gray World* assumption is that over all scenes $R_{avg} = G_{avg} = B_{avg}$
  - Simple GrayWorld Algorithm: equalize the averages for the three color channels by dividing each red value by $(\frac{R_{avg}}{G_{avg}})$ and each blue by $(\frac{B_{avg}}{G_{avg}})$ (this works ok except when we have "atypical" scenes, e.g., a forest with mostly bright green leaves — the image will look grayish after white balancing)
  - Another approach is to use the color information to estimate (or decide on) the illuminant

Color Correction

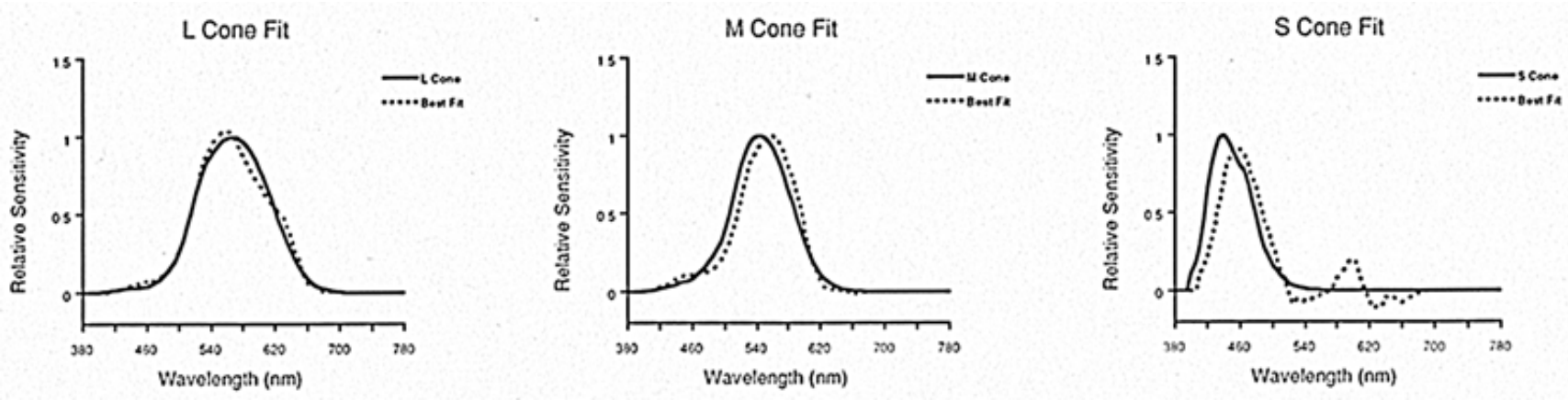
- Color filter technology and photodetector spectral response determine the camera color space
- To ensure that color from the camera looks the same on a display, the camera output must be transformed to the display color spectral response space
- Since there may be *many* display types used to render a captured image, it is customary to transform the camera output to a standard color space, *e.g.*, corresponding to the LMS spectral responses, in the camera — correction for each display type is performed outside the camera
- To transform the camera output to a standard color space we use a 3×3 matrix D, thus if $\mathbf{C}$ is the color from a pixel, the corrected color

$$\mathbf{C}_o = D\mathbf{C}$$

- So how do we find D ?

  If A_1 is the camera spectral response matrix $(3 \times n)$ and A_2 is the LMS spectral response matrix $(3 \times n)$, then we can select D such that DA_1 is as close to A_2 as possible, which can be done, for example, using least squares

- This seems to work well, the following is the corrected RGB (of the Kodak camera) compared to LMS spectral responses



Gamma Correction

- Intensity of the light generated by a display device Z is not linear in its input Y, e.g., $Z \propto Y^{2.22}$
- Must prewarp the image sensor output X so that the output of the display is linear in the illumination at the camera — done using a companding function, e.g., $Y \propto X^{0.45}$
- Also needed prior to image enhancement and compression
    - Most image processing algorithms assume pixel values proportional to perceptual brightness, which is close to the prewarped value Y
- Typically implemented using 3 lookup tables, one for each color component

Color Space Conversion

- Transform RGB to YCbCr, or to YUV using 3×3 matrix

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- Most image enhancement and compression are performed on luminance and chrominace values separately
  - Eye is more sensitive to luminance than to chrominance
  - Preserve color before and after processing