INTERNATIONAL COLOUR VISION SOCIETY

Chromaticity Conversions

Here we introduce the maths behind colour space conversions along with some example code to perform these conversions.

1. Prerequisites

MATLAB (tested on 2022b)

PsychToolbox-3 (tested on version 3.0.19)

2. Theory

You will be familiar with the idea that displays produce colours as an RGB triplet. However, RGB coordinates are device dependent and thus not appropriate for describing colour between different displays. One must therefore convert from RGB coordinates to a device independent colour space. Typical device independent colour spaces used are LMS and XYZ. Here we introduce these key colour spaces and how to convert RGB coordinates to them.

3. LMS

One of the most useful spaces for the colour vision scientist is the LMS space. This is the activation in each cone class. Each cone class has a known spectral sensitivity, $T_L(\lambda)$, $T_M(\lambda)$, and $T_S(\lambda)$. To calculate the activation in each cone class to a given radiance spectrum of physical light, $P(\lambda)$, one simply multiplies and integrates the spectral sensitivities with the spectrum i.e. L = $\int T_L(\lambda)P(\lambda)\,d\lambda$, M = $\int T_M(\lambda)P(\lambda)\,d\lambda$, $S = \int T_S(\lambda)P(\lambda)\,d\lambda$. This can be implemented in MATLAB using the * operator (though be careful to make sure you have the dimensions of your matrices correct):

```
>> LMS_activations = lms_spectral_sensitvities * spectrum;
```

The radiance spectrum can be obtained by measuring the physical light (see the Light Measurement Worksheet). Also, as noted in the worksheet, this formulation assumes that power is expressed as power per wavelength bin, not power per nm. And that is how spectra are returned by the Psychtoolbox interface to the PR-670 and SpectroCAL. Note as well that the convention in Psychtoolbox is for the spectral sensitivities/sensors to occupy the rows in a matrix and the primaries/spectrum to occupy the columns.

The cone spectral sensitivities are often taken from a standard observer model, such as the Stockman-Sharpe standard observer (which can be downloaded from **cvrl.org**). You need to consider which set of spectral sensitivities best suit your purpose. For example, the effective absorbance of each cone class is different depending on whether stimuli are presented locally to the fovea, or more eccentrically. This is owing to different density of macular pigment throughout the eye. This change in ‘field size’ is accounted for by many models of the cone spectral sensitivities – discuss in your groups how you decide which field size to choose.

3.1. RGB -> LMS

One common problem encountered is the need to reproduce a desired LMS triplet on a display. To do so one needs to know which RGB triplet will generate a spectral output of the display that induces the desired LMS triplet. This can be easily solved as the display primaries and cone spectral sensitivities are simply a linear transformation of each other (assuming the display has been linearized first – see Gamma Correction Worksheet). Thus, one needs to solve for the matrix N in the following equation:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = N \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

In MATLAB:

```
>> LMS = N * RGB;
```

So how do we find matrix N? Matrix N is simply given as below:

$$N = \begin{bmatrix} L_R & L_G & L_B \\ M_R & M_G & M_B \\ S_R & S_G & S_B \end{bmatrix}$$

Where L_R is the L cone activation in response to the R primary etc. Work through the maths in your group to derive why this is the case. Hint: consider what happens when you think about the RGB triplets [1, 0, 0], [0, 1, 0], and [0, 0, 1]. Ask the faculty for help if needed.

Thus one needs now to find L_R, L_G, L_B etc. We have already seen above how to find the cone activation for a given spectrum! Thus we simply must have a measurement of each of the primaries, $P_R(\lambda), P_G(\lambda), P_B(\lambda)$, and the LMS spectral sensitivities and we can multiple and integrate each in turn to find the elements of matrix N as below:

$$\begin{bmatrix} L_R = \int P_R(\lambda)T_L(\lambda)d\lambda & L_G = \int P_G(\lambda)T_L(\lambda)d\lambda & L_B = \int P_B(\lambda)T_L(\lambda)d\lambda \\ M_R = \int P_R(\lambda)T_M(\lambda)d\lambda & M_G = \int P_G(\lambda)T_M(\lambda)d\lambda & M_B = \int P_B(\lambda)T_M(\lambda)d\lambda \\ S_R = \int P_R(\lambda)T_S(\lambda)d\lambda & S_G = \int P_G(\lambda)T_S(\lambda)d\lambda & S_B = \int P_B(\lambda)T_S(\lambda)d\lambda \end{bmatrix}$$

We thus need accurate measurements of the primaries of our display. The provided MATLAB script **measurePrimaries.m** contains this procedure, using either the **PR670** or **SpectroCAL**. Simply, a spectral recording of each primary light is taken, and then saved into a **.mat** file.

We can then use those primaries, and load in our model of the spectral sensitivities (e.g. as downloaded from **cvrl.org**). We can then find the matrix N in MATLAB as:

```
>> N = LMS_spectral_sensitivities * primaries;
```

3.2. LMS -> RGB

This process gives us the RGB -> LMS conversion, but often we want to convert LMS->RGB. To do this we must simply invert the matrix N:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = N \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$

This can be done in MATLAB using:

```
>> inv(N);
```

4. CIE XYZ

Perhaps one of the most ubiquitous tools in colorimetry, the CIE 1931 XYZ colour matching functions (CMFs) was the first device-independent colour space that does not depend on the characteristics of the display system on which colour is being rendered. Colour matching data from Guild, 1931, and Wright 1929 & 1930, were used to derive the CIE 1931 XYZ CMFs. The CIE 1931 XYZ CMFs are the first notion of a standard observer: where a single set of functions, derived from the average of, in this case, 18 observers, can be used to describe the "average" observer.

As the real RGB primary lights lead to $P_R(\lambda)$ going negative, i.e. the R primary needed to be added to the target stimulus, in order for a colour match to be made, the CIE proposed a set of imaginary primaries, XYZ, with associated CMFs, $x(\lambda), y(\lambda), z(\lambda)$ which are always positive, and which can be derived from the real $r(\lambda),\ g(\lambda),\ b(\lambda)$, CMFs by a simple linear transformation. It is these, $x(\lambda), y(\lambda), z(\lambda)$ CMFs for the imaginary XYZ primaries that are most commonly used in colorimetry. Note that (when scaled appropriately) $y(\lambda)$ is the luminous efficiency function, and thus the Y value indicates the luminance of the stimulus (in cd/m^2).

Nowadays, the CIE 1931 XYZ functions are considered out-dated as they are not a linear transformation of the CIE adopted Stockman-Sharpe cone spectral sensitivities (CIE 2006), and an updated set of CIE XYZ functions which are a linear transformation of the adopted fundamentals are used instead. However, many hardware devices may still use the CIE 1931 XYZ functions depending on when they were made, so check which CIE XYZ space you are working in.

Similarly to how one calculates the cone activations, one can calculate the XYZ tristimulus value of a given radiance spectrum of physical light, $P(\lambda)$, by multiplying and integrating the colour matching functions with the spectrum i.e. X = $\int x(\lambda)\, P(\lambda)\, d\lambda$, Y = $\int y(\lambda)\, P(\lambda)\, d\lambda$, $Z = \int z(\lambda)\, P(\lambda)\, d\lambda$. This can be implemented in MATLAB using the * operator (though be careful to make sure you have the dimensions of your matrices correct):

```
>> XYZ_tristimulus_values = xyz_functions * spectrum;
```

4.1. RGB -> XYZ

RGB to XYZ works almost identically as RGB to LMS:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = N \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The only difference is that the matrix N is defined in terms of XYZ not LMS as below:

$$N = \begin{bmatrix} X_R & X_G & X_B \\ Y_R & Y_G & Y_B \\ Z_R & Z_G & Z_B \end{bmatrix}$$

Can you work through in your group how to find the elements of matrix N and implement this conversion in MATLAB given what we have demonstrated above for the LMS example?

4.2. RGB -> XYZ

Again, here we simply invert the matrix N:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = N \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

This can be done in MATLAB using:

```
>> inv(N);
```

4.3. Example conversion

exampleFindXYZ.m shows a short example of how you may choose an RGB value, gamma correct it (discuss in your groups why this is necessary), and then convert it to XYZ using **RGBconversion.m**.

5. Other CIE colour spaces

There are other colour spaces one may encounter, all of which are transforms of the CIE XYZ space. Here we introduce those colour spaces, and the provided MATLAB code **RGBconversion.m** demonstrates how to convert from RGB to those colour spaces (via the XYZ space). Additionally the **ColorSpaces.m** script demonstrates how to convert the chromaticity of D65 between some of these colour spaces.

5.1 CIE xyY

Another useful tool in colorimetry is the chromaticity diagram. Chromaticity coordinates are defined independently of intensity i.e. such that black and white at the intensity extremes of the achromatic axis would have the same chromaticity coordinates. By removing intensity information, chromaticity coordinates need only capture a two-dimensional representation of the human colour percept, which is thus easier to depict diagrammatically. The CIE 1931 xy chromaticity diagram is commonly used in industrial applications.

The CIE 1931 XYZ CMFs translate to the CIE 1931 xy chromaticity coordinates as:

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

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

$$z = \frac{Z}{X + Y + Z} = 1 - x - y$$

This can be implemented in MATLAB with the Psychtoolbox function:

```
>> xyY = XYZToxyY(XYZ);
```

5.2. CIE Luv

A major drawback of the CIE 1931 xyY chromaticity diagram is that it is not perceptually uniform. That is to say there are larger colour appearance changes associated with a given separation of points in some regions of the chromaticity diagram than in others. The CIE 1976 Lu^*v^*, often abbreviated to CIELUV, colour space was the first attempt to design a perceptually uniform colour space. The Lu^*v^* values are derived from a non-linear combination of the CIE 1931 XYZ tristimulus values, and require a white point as reference. This conversion can be implemented in MATLAB via the Psychtoolbox function:

```
>> LUV = XYZToLuv(XYZ,whiteXYZ);
```

The CIELUV u^*v^* coordinates are based on the CIE u'v' chromaticity diagram. The CIE u'v' chromaticity coordinates can be implemented in MATLAB via the Psychtoolbox function:

```
>> uv = XYZTouv(XYZ);
```

5.3. CIELAB

1976 also saw the development of the CIELAB, or CIE 1976 L*a*b* colour space. Like the CIELUV space, CIELAB was designed to be perceptually uniform. The L* value is common between the two colour spaces, and is an expression of perceptual lightness, where 0 is black and 100 is diffuse white. Crucially, this lightness value is relative to a predefined white point to which the observer is adapted (and in fact, so are the u*v* and a*b* values). The CIELAB space differs from the CIELUV space as the associated chromaticity coordinates for the CIELAB space, a*b*, are designed to represent the unique hues of colour vision. The a* axis is intended to represent the red-green opponent colours, while the b* axis represents blue-yellow.

To convert from XYZ to LAB in MATLAB one can use the Psychtoolbox function:

```
>> LAB = XYZToLab(XYZ,whiteXYZ);
```

Like CIELUV, CIELAB has been widely adopted by colour vision scientists, engineers, and the display industry. Of particular prevalence is the use of CIELAB to compute the colour difference, ΔE, between two colours as defined in CIELAB coordinates. In MATLAB (via Psychtoolbox):

```
>> dE = ComputerDE2000_Lab(LAB_1,LAB_2);
```

6. MacLeod-Boynton Chromaticity Diagram

The colour spaces described above show a range of attempts to quantify the perceptual dimensions of colour. An alternative method is to quantify the physiological dimensions of colour directly i.e. the relative cone photoreceptor signals. One common way to do this is using the MacLeod-Boynton chromaticity diagram. In the MacLeod-Boynton chromaticity diagram the relative S cone signals compared to the summed L and M cone signal, $s_{MB}(\lambda)$, is plotted against the L cone signals relative to the summed L and M signals, $l_{MB}(\lambda)$, defined as below:

$$s_{MB}(\lambda) = \frac{a_s T_S(\lambda)}{V(\lambda)}$$

$$l_{MB}(\lambda) = \frac{a_L T_L(\lambda)}{V(\lambda)}$$

Where

$$V(\lambda) = a_L T_L(\lambda) + \ a_M T_M(\lambda)$$

And $V(\lambda)$ and $s_{MB}(\lambda)$ are scaled to peak at unity.

The MacLeod-Boynton chromaticity diagram is equiluminant by definition. One must decide what set of cone spectral sensitivity functions to use to calculate their MacLeod-Boynton chromaticity coordinates. When the 10° Stockman Sharpe cone spectral sensitivities (energy) are used the scaling constants a_L, a_M, a_S, required such that $V(\lambda)$ and $s_{MB}(\lambda)$ peak at unity are: $a_L = 0.692839, a_M = 0.349676, a_S = 0.0554786.$ More generally it can be specified using any set of cone spectral sensitivities and scaling of the luminous efficiency function, for which a different set of scaling constants must be derived.

Chromaticity Conversions