Light Measurement and Display

Here we introduce various devices available in the lab, including spectroradiometers and photometers There is information about the different devices available, key points about how they should be used, and example code for automating the measurements in the context of displaying standardized colorspace-defined colors on an RGB display.

1. Prerequisites

MATLAB (tested on 2022b)

PsychToolbox-3 (tested on version 3.0.19)

Additional functions provided by CRS (available at <https://www.crsltd.com/tools-for-vision-science/light-measurement-display-calibation/colorcal-mkii-colorimeter/nest/product-support>)

1. Spectral Measurements

Relevant MATLAB file: **measureRadiance.m**

Spectral measurements are taken using **spectroradiometers**. There are two available in the lab - the **SpectraScan PR670** and the **CRS MKII SpectroCAL**. The following code is summarised from the ‘measureRadiance.m’ file available on the ICVS GitHub.

* 1. PR670

The PR670 can be used as a standalone device or plugged into a computer via the included USB cable and controlled using MATLAB. Measurements can be taken by pressing the ‘measurement’ button on top of the device. The device records a spatial average across the black circle visible through the aperture. There is a dial on the left side of the device, which closes a shutter over the aperture. Be sure to close this when the device is in use to avoid lens reflections affecting the measurement.

When using the PR670 with a computer, connect the devices via USB before turning on the PR670. Control of the device is enabled with PsychToolbox functions. First, a connection must be established between the computer and the device. Use the following function to do this.

>> PR670init();

In order to make a spectral measurement using the PR670, use the following function:

>> PR670measspd([380 5 81]);

Here, the input argument, [380 5 81], signified the bin size and wavelength range the device records over. In this case, 81 bins in 5-nm increments starting at 380nm. You should hear the sound of the shutter moving inside the device, before a beep. The function returns a one-dimensional array, where each array element corresponds to the energy recorded from each bin across the visible light spectrum.

* 1. SpectroCAL

The SpectroCAL works slightly differently to the PR670. It does not work as a standalone device, so can only be controlled by a computer. When plugged in, a red light should appear at the back of the device. Press down on this light to turn on the laser pointer. The device records a spatial average across the ring drawn by the laser, centred at the dot in the middle. To record a spectrum measurement, use the following function (there is no need to specifically initialise the device):

>> [CIEXY, CIEUV, Luminance, Lambda, Radiance] = SpectroCALMakeSPDMeasurement(port,380,780,5)

Here, the four input arguments define the port name (this will depend on your device), the start and end wavelengths, and the bin size between these. This function has many possible outputs. These range from CIE XYZ and Luv values, as well as luminance. However, we mainly use this device for making radiance measurements.

The MATLAB file ‘measureRadiance.m’ can be used to make radiance measurements using either device.

1. Luminance Measurements

Relevant MATLAB file: **measureLuminance.m**

**Photometers** are used to carry out luminance measurements. In the lab, we have two photometers – the **CRS ColorCal MKII** and the **UDT Photometer**.

* 1. ColorCAL

The ColorCAL uses functions freely available on the CRS website (see section 1). These come in the form of a single function structure with many in-built options. To see these options, use:

>> help ColorCal2

To measure luminance using the device, first make an XYZ recording, and then single out the Y value. In order for the recording to be accurate, you must correct the recording using a calibration matrix stored on the device. For best results, set the ColorCal up on its tripod and gently press the device to a display, so that any other light is kept out.

>> cMatrix = ColorCal2('ReadColorMatrix');

>> s = ColorCal2('MeasureXYZ');

>> correctedValues = cMatrix(1:3,:) \* [s.x s.y s.z]';

* 1. UDT Photometer

TBC

The MATLAB file ‘measureLuminance.m’ can be used to make luminance measurements using either device.

**If luminance devices are unavailable, spectroradiometers can be used to make luminance measurements, utilizing a luminous efficiency function.**

1. Gamma Measurement and Correction

Relevant MATLAB files: **measureGamma.m**, **exampleGammaCorrection.m**

* 1. Gamma Measurement

Most modern displays feature a ‘**gamma**’. This is where the input values to the display are not linearly related to the brightness of the pixels. This is in order to increase the contrast during typical use. However, when displaying colorimetrically defined stimuli, this feature is undesirable. Researchers typically linearize the display by correcting this gamma. In order to do this, they must first record how the display’s luminance changes as its input changes, to test whether this relationship is linear or not. These measurements are taken for the red, green and blue channels separately, as the gamma applied to each of these channels may differ.

**measureGamma.m** allows you to measure a display’s gamma using either luminance meter. Simply, this code loops through many luminance recordings, increasing the input to the display between each recording. Using the **‘tri’** mode, we will loop through the red, green and blue channels. Once all recordings for one channel are made, a power curve is fitted to the recorded data, as this reflects the kind of non-linearity of displays. This means that the data need to be normalized such that the maximum output, found at the maximum input, is (1,1). This is because, no matter the gamma, the maximum value should not be transformed, and 1 to any exponent is still 1.MATLAB’s **fit** function is used to fit a power law function in the format:

Where **L** is the output luminance, **I** is the input value, and **b**, **c** and **d** are values found by the **fit** function.

Check the figured generated by the code. A clear curve indicates a gamma, which should be reinforced by the fit statistics. A typical gamma is *c* = 2.2. The script will save the *c* values for each channel into a **.mat** file.

* 1. Gamma Correction

Once a gamma has been measured, a display’s output can be ‘linearized’ by using the gamma coefficients to correct input values. **exampleGammaCorrection.m** shows a short example of how input values are corrected using measured gamma values. Simply, input values are mapped to a power curve of **reciprocal exponent** to the recorded gamma. As such correcting input values looks like:

>> correctedValues = initialValues.^(1./gammaValues);

Where **initialValues** is the RGB triplet to be corrected, and **gammaValues** is the triplet of gamma exponents generated from **measureGamma.m**.

1. RGB Conversion

Relevant MATLAB files: **measurePrimaries.m**, **RGBconversion.m**, **exampleFindXYZ.m**

* 1. Measuring Primaries

In order to be able to precisely control the color output of a display, we need to know the spectral distribution of each of its primary lights. **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.

* 1. Conversion from RGB

It is often useful to be able to convert colors shown on a display into standardized tristimulus values such as CIE XYZ, CIE L\*u’v’, CIE L\*a\*b\*, xyY or LMS. A conversion matrix is required to make this calculation. This involves determining how much of each primary is ‘seen’ by each of the activating spectra for a given colorspace. **RGBconversion.**m contains procedures for converting RGB values to many colorspaces. For example, how much the red primary activates each of the L, M and S cones. Calculating each of these contingencies gives us a linear weighting on each cone class from each primary. Organizing these weightings as below allows us to convert linearized RGB values into relative LMS values. It is important to note that the RGB values used **must** be linearized, as the matrix multiplication uses linear multiplication across each channel/cone class. The same process is completed using color matching functions (CMFs) for conversion to the CIE XYZ colorspace. Other CIE colorspaces (such as L\*u’v’ and L\*a\*b\*) use XYZ values generated from CMFs, and then further manipulate these values.

>> Lr = redPrimary \* longCone

>> Lg = greenPrimary \* longCone

>> Lb = bluePrimary \* longCone

>> Mr = redPrimary \* mediumCone

>> Mg = greenPrimary \* mediumCone

>> Mb = bluePrimary \* mediumCone

>> Sr = redPrimary \* shortCone

>> Sg = greenPrimary \* shortCone

>> Sb = bluePrimary \* shortCone

>> matrix = [Lr, Lg, Lb;

Mr, Mg, Mb;

Sr, Sg, Sb]

>> LMS = matrix \* RGB

* 1. Example conversion

**exampleFindXYZ.m** shows a short example of how you may choose an RGB value, gamma correct it, and then convert it to XYZ using **RGBconversion.m**.

[Psychophysical Measurement and Display 1](#_Toc141299639)

[1. Prerequisites 1](#_Toc141299640)

[2. Spectral Measurements 1](#_Toc141299641)

[2.1. PR670 1](#_Toc141299642)

[2.2. SpectroCAL 2](#_Toc141299643)

[The MATLAB file ‘measureRadiance.m’ can be used to make radiance measurements using either device. 2](#_Toc141299644)

[3. Luminance Measurements 2](#_Toc141299645)

[3.1. ColorCAL 2](#_Toc141299646)

[3.2. UDT Photometer 3](#_Toc141299647)

[The MATLAB file ‘measureLuminance.m’ can be used to make luminance measurements using either device. 3](#_Toc141299648)

[4. Gamma Measurement and Correction 3](#_Toc141299649)

[4.1. Gamma Measurement 3](#_Toc141299650)

[4.2. Gamma Correction 4](#_Toc141299651)

[5. RGB Conversion 4](#_Toc141299652)

[5.1. Measuring Primaries 4](#_Toc141299653)

[5.2. Conversion from RGB 4](#_Toc141299654)

[5.3. Example conversion 5](#_Toc141299655)