# Colors in DKL Space

## Introduction

This folder contains programs to do the following:

1. Convert colors from one space to another. We consider four spaces – RGB, XYZ, LMS and DKL.
2. Generate color patches that lie on an equi-luminant space and form an ellipse (or circle) on the DKL space (generateDKLColors and generateDKLColorsOnCircle).
3. Generate color patches that lie on an equi-luminant plane and span different saturation levels going from the white point towards the primaries (generateCIEpointsOnLine).

It is assumed that the monitor has been linearized already.

Functions are used from the following two resources:

1. Psychtoolbox (PTB) (<http://www.psychtoolbox.org/>), a freely available software in Matlab.
2. Stephen Westland’s toolbox (<https://stephenwestland.co.uk/matlab/index.html>)

## Conversion between different color spaces

1. **RGB to XYZ space**

For this, we just need the CIE (x,y) coordinates of the three primaries and the white point. The matrix that does the conversion can be found using RGBToXYZMatrix program in PTB.

1. **XYZ to LMS space**

LMS values are obtained from the spectra by taking the dot product with the cone fundamentals (CFs). However, note that the CFs are simply scaled to have the maximum value of 1, and therefore the LMS values need to be “scaled” properly. In particular, in order for L+M to be equal to the luminance, L and M need to be in “luminance units”. The scaling factors depend on the CFs that we use, as well as the luminosity function (which should be a linear combination of the CFs of L and M). In fact, the command computeDKL\_M in PTB takes the CFs and luminosity function as inputs and finds out the appropriate weights (LMLumWeights). So, in order to get LMS in “luminance units”, we can do one of the following:

a) Obtain LMS using the cone fundamentals and then apply an appropriate scaling factor. For example, the lms2rgbMB program in Stephen Westland’s book assumes a default scaling factor of [0.689903 0.348322 0.0371597] for L, M and S. These scaling factors correspond to Stockman&Sharpe 2000 CF and CIE 2006 XYZ CMF (which can be found on cvrl.org).

The matrix that converts the CFs to XYZ CMFs is given by the following (see Stockman, 2019, Current Opinion in Behavioral Sciences):

M\_LMSToXYZ =

Note that the second row that the scaling values for L and M.

Therefore, the inverse of this matrix gives the transformation from XYZ to LMS. However, for those LMS values, we have 0.68990272L + 0.34832189M = Y. To convert the L and M values to luminance units so that L+M=Y, we need to pre-multiply by the scaling matrix diag([0.689903 0.348322 0.0371597]). The final matrix: diag([0.689903 0.348322 0.0371597])\*inv(M\_LMSToXYZ) is

M\_XYZToLMS\_SS =

b) We can use CFs that directly yield LMS values in luminance units. For example, the one used by Smith and Pokorny, 1975. Macleod and Boynton, 1978, used the following matrix to get LMS values directly from XYZ values (obtained using CMFs of Judd-Vos, 1978)

M\_XYZToLMS\_SP =

Working backwards, this corresponds to a scaling factor of [0.6372 0.3920 0.0260].

The program XYZToLMSMatrix provides this matrix.

1. **LMS to DKL space**

We can get the DKL coordinates from LMS (expressed in luminance units) by postmutiplying LMS with a simple matrix transformation B given by:

B =

and subsequently scaling each entry. Here LMS\_background = [b1 b2 b3].

This matrix essentially describes each mechanism. If we post-multiply by a 3x1 vector [L M S]’, we get a 3x1 vector with the first entry being luminance (L+M), second entry as L – b1\*M/b2 and the third entry as -L -M + (b1+b2)S/b3. These are the luminance (L+M), chromatic L-M and chromatic S-(L+M) mechanisms, scaled appropriately by their baseline values to yield the chromatic contrast. The subsequent scaling of each mechanism is a bit tricky. Scaling is done such that each mechanism-isolating stimulus with unit pooled contrast produces unit responses in the three DKL mechanisms. This is done in computeDKL\_M in PTB as well as lms2dkl.m in Stephen Westland’s Matlab toolbox (both of which basically followed from Brainard, 1996). However, we can show that the same scaling operation can be achieved by a simple matrix multiplication, as explained in LMS2DKLMatrix\_Explanation.pdf.

The program LMSToDKLMatrix provides the required matrix.

## Generate colors that fall on an ellipse in the DKL space on the equi-luminance plane (theta=0)

The logic described below is taken from Lablib, written by Dr. John Maunsell in Objective C (<https://github.com/MaunsellLab/Lablib-Public-05-July-2016>). The color conversion idea was developed by Dr. Maunsell and Dr. Geoff Ghose.

The DKL space is essentially an extension of the Macleod Boynton (MB) space. MB space is a 2D space defined by variables r and b. It is a projection of the LMS space (called RGB by MB – not to be confused by the RGB color space) on the L+M=1 plane. Along the x-axis, we get both r and g = 1 – r. Moving along the x-axis does not change the blue cone activity, so it is a constant blue axis. Moving along the y-axis only changes b but not r or g, so it is a constant R&G axis. People who do not have blue cones cannot distinguish colors that are parallel to the y-axis, so this is also the tritanopic confusion line. We refer the x and y axes as cb (constant blue) and tc (tritanopic confusion) axes, following the convention used in Lablib.

From the LMS values, we can get the rb coordinates of the MB space. The DKL space simply involves another luminance axis passing through the white point orthogonal to the MB space. In our case, we wish to stay in the iso-luminant plane, so we need not worry about the third axis for now. In the 2D MB plane, we can simply find the largest ellipse that can fit in the monitor’s Gamut. So, we first find out the points on the MB plane (rb) values, from which we can get the xy coordinates. Then, given a luminance value (in our case it is 0.5), we get our xyY coordinate, from which we get XYZ and use the XYZ2RGB transform to get the desired RGB values.

The program generateDKLColors generates these colors, while displayColors displays these colors in various spaces.