Emulation of one Camera by another Camera

Glenn Davis <gdavis@gluonics.com>

January 27, 2024

Introduction

The goal of this package **colorSpec** vignette is to reproduce the findings of [1] and [2], where a camera with Foveon X3 sensor is modified to closely emulate the spectral responses of the human eye. The two modifications are:

- a prefilter in front of the camera; this modification is optical and classical, see [3]

- a 3x3 matrix applied to the camera output; this modification is in hardware or in software

The figures below are best viewed on a display calibrated for sRGB. Featured functions in this vignette are: `emulate()`.

```
library( colorSpec )
library( spacesRGB )     # for function plotPatchesRGB()
```

1 BT.709.RGB and Foveon X3

The camera `BT.709.RGB` is not real; it is a theoretical camera whose spectral responses are linear combinations of the responses of the human eye (the *standard observer*). See the man page of `BT.709.RGB` for details. Create a fixed wavelength vector, and resample both the Foveon camera, and the reference (the ideal) camera to the same wavelengths. Calibrate and plot both cameras.

```
wave = 380:720
# read the Macbeth ColorCheck target
path = system.file( 'extdata/cameras/FoveonX3.txt', package='colorSpec')
foveon = radiometric( readSpectra( path, wave=wave ) )
reference = resample( BT.709.RGB, wave=wave )
# calibrate so that both have the same response RGB=(1,1,1) to Illuminant E
illum = illuminantE(wave=wave)
foveon = calibrate( foveon, stimulus=illum )
reference = calibrate( reference, stimulus=illum )
# plot both for comparison
par( omi=c(0,0,0,0), mai=c(0.5,0.9,0.1,0) )
plot( reference, main='' )
plot( foveon, lty=2, add=TRUE, legend=FALSE, color=c('red','green','blue') )
```

See Figure 1. These spectral responses are obviously quite different; although the area under all 6 curves is 1. To visualize the difference we will use the ever-popular *ColorChecker* target. The data for this target has been kindly provided in CGATS format by [4]. *ColorChecker* is a Registered Trademark of X-Rite, and X-Rite is a Trademark.
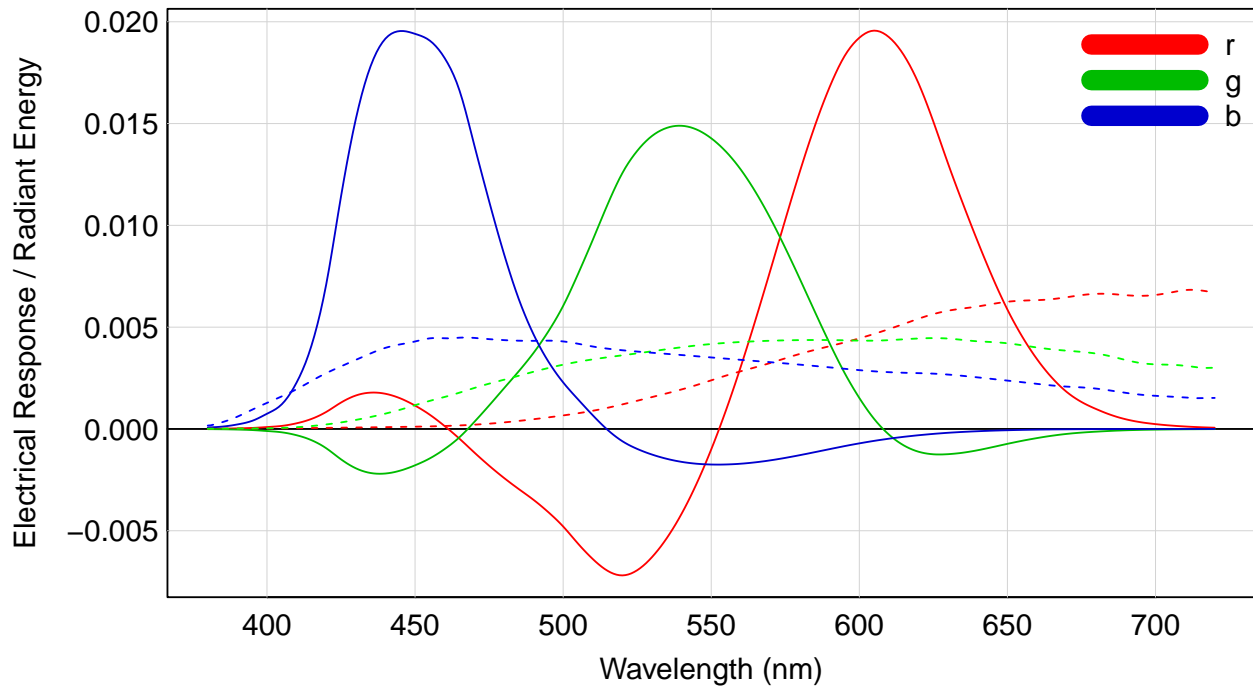
Figure 1: Reference camera BT.709.RGB (solid) vs Foveon X3 camera (dashed)

```
# read the Macbeth ColorCheck target
path = system.file( 'extdata/targets/CC_Avg30_spectrum_CGATS.txt', package='colorSpec')
MacbethCC = readSpectra( path, wave=wave )  # MacbethCC is a 'colorSpec' object
MacbethCC = MacbethCC[ order(MacbethCC$SAMPLE_ID), ]  # still class 'colorSpec'
print( extradata(MacbethCC), row.names=F )
```

SAMPLE_ID	SAMPLE_NAME	Munsell	ISCC-NBS_Name	LEFT	TOP	WIDTH	HEIGHT
1	dark skin	3YR 3.7/3.2	moderate brown	7	9	29	29
2	light skin	2.2YR 6.47/4.1	light reddish brown	40	9	29	29
3	blue sky	4.3PB 4.95/5.5	moderate blue	73	9	29	29
4	foliage	6.7GY 4.2/4.1	moderate olive green	106	9	29	29
5	blue flower	9.7PB 5.47/6.7	light violet	139	9	29	29
6	bluish green	2.5BG 7/6	light bluish green	172	9	29	29
7	orange	5YR 6/11	strong orange	7	42	29	29
8	purplish blue	7.5PB 4/10.7	strong purplish blue	40	42	29	29
9	moderate red	2.5R 5/10	moderate red	73	42	29	29
10	purple	5P 3/7	deep purple	106	42	29	29
11	yellow green	5GY 7.1/9.1	strong yellow green	139	42	29	29
12	orange yellow	10YR 7/10.5	strong orange yellow	172	42	29	29
13	Blue	7.5PB 2.9/12.7	vivid purplish blue	7	75	29	29
14	Green	0.25G 5.4/8.65	strong yellowish green	40	75	29	29
15	Red	5R 4/12	strong red	73	75	29	29
16	Yellow	5Y 8/11.1	vivid yellow	106	75	29	29
17	Magenta	2.5RP 5/12	strong reddish purple	139	75	29	29
18	Cyan	5B 5/8	strong greenish blue	172	75	29	29
19	white	N9.5/	white	7	108	29	29
20	neutral 8	N8/	light gray	40	108	29	29
21	neutral 6.5	N6.5/	light medium gray	73	108	29	29
22	neutral 5	N5/	medium gray	106	108	29	29
23	neutral 3.5	N3.5/	dark gray	139	108	29	29
24	black	N2/	black	172	108	29	29

Note that `MacbethCC` is organized as `'df.row'` and contains extra data for each spectrum, most importantly the coordinates of the patch rectangles.

Calculate the RGB responses to both cameras and display them.

```
RGB.ref = product( illum, MacbethCC, reference) # this is *linear scene* sRGB
# add the rectangle data to RGB.ref, so the patches are plotted in proper places
df.ref = extradata(MacbethCC)
df.ref$RGB.ref = RGB.ref
#  display in proper location, and use the sRGB display transfer function
par( omi=c(0,0,0,0), mai=c(0,0,0,0) )
plotPatchesRGB( df.ref, space='sRGB', which='scene', back='gray20', labels=FALSE )
# repeat with foveon camera, and add to existing plot
RGB.foveon = product( illum, MacbethCC, foveon )
df.foveon = extradata(MacbethCC)
df.foveon$RGB.foveon = RGB.foveon
plotPatchesRGB( df.foveon, space='sRGB', which='scene', shape='bottomright', add=T )
```
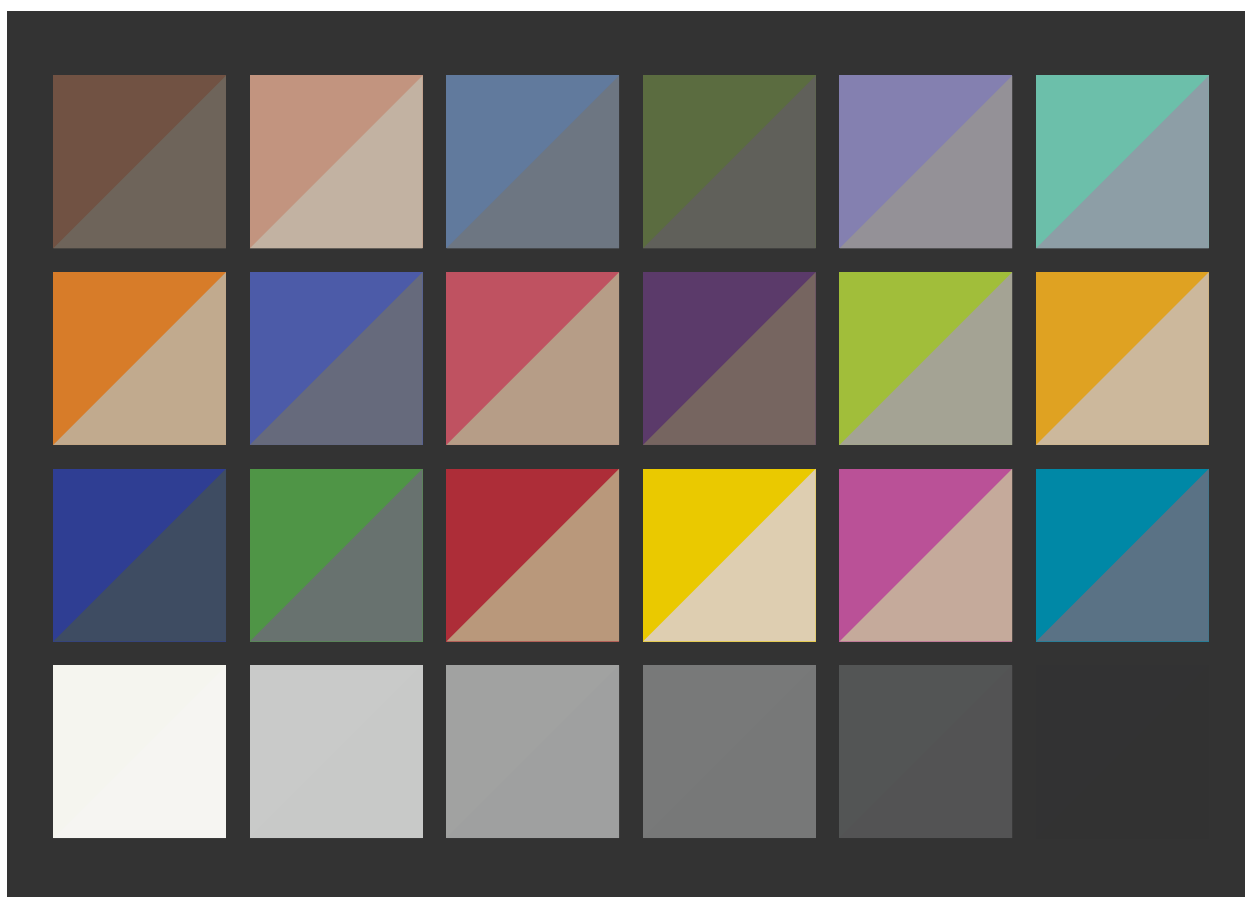

Figure 2: Rendering with Illuminant E, with Foveon RGB in bottom right half

There is only agreement for the neutral patches, as might be expected.

Now modify the Foveon camera, using both a pre-filter and a matrix, to emulate the reference.

```
foveon.mod = emulate( foveon, reference, filter=TRUE, matrix=TRUE )
par( omi=c(0,0,0,0), mai=c(0.5,0.9,0.2,0) )
plot( reference, main='' )
plot( foveon.mod, lty=2, add=TRUE, legend=FALSE )
```
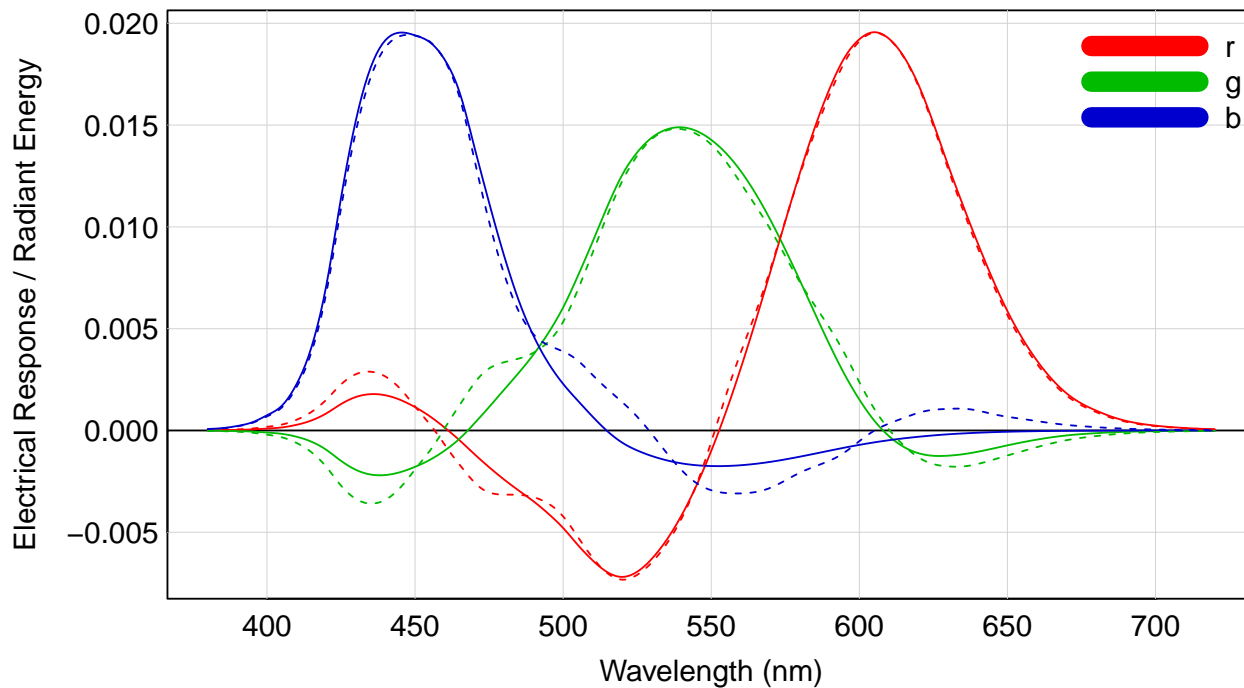

Figure 3: Reference camera (solid) vs the modified Foveon camera (dashed)

The agreement is now much better. Replot the ColorChecker to visualize the improvement.

```
par( omi=c(0,0,0,0), mai=c(0,0,0,0) )
plotPatchesRGB( df.ref, space='sRGB', which='scene', back='gray20', labels=FALSE )
# repeat with modified foveon camera, and add to existing plot
df.foveon.mod = extradata(MacbethCC)
df.foveon.mod$RGB.foveon.mod  = product( illum, MacbethCC, foveon.mod )
plotPatchesRGB( df.foveon.mod, space='sRGB', which='scene', shape='bottomright', add=T )
```

Figure 4: Rendering with Illuminant E, with modified Foveon RGB in bottom right half

The agreement in the RGBs is now much bettter, c.f. Figure 2. There is a noticeable difference in the **Red** and **Magenta** patches, and minor differences in some others. However, the neutrals are now worse; the green is low so they have a purple tint. A new feature - *white-point preservation* - might be added to a future version of `emulate()`, using the techniques in [5]. Alternatively, one could also re-calibrate (white-balance) `foveon.mod`.

The computed pre-filter and matrix are attached to `foveon.mod`, and are easy to print and plot.

```
attr(foveon.mod,"emulate")$A

              r          g          b
Red     13.054064  -7.362972   5.627526
Green  -10.008693  10.482482 -13.407764
Blue     2.848111  -3.419678  11.254107


par( omi=c(0,0,0,0), mai=c(0.5,0.9,0.2,0) )
prefilter = attr(foveon.mod,"emulate")$filter
specnames(prefilter) = "prefilter for modified Foveon"
plot( prefilter,  main='', ylim=c(0,1.1) )
```
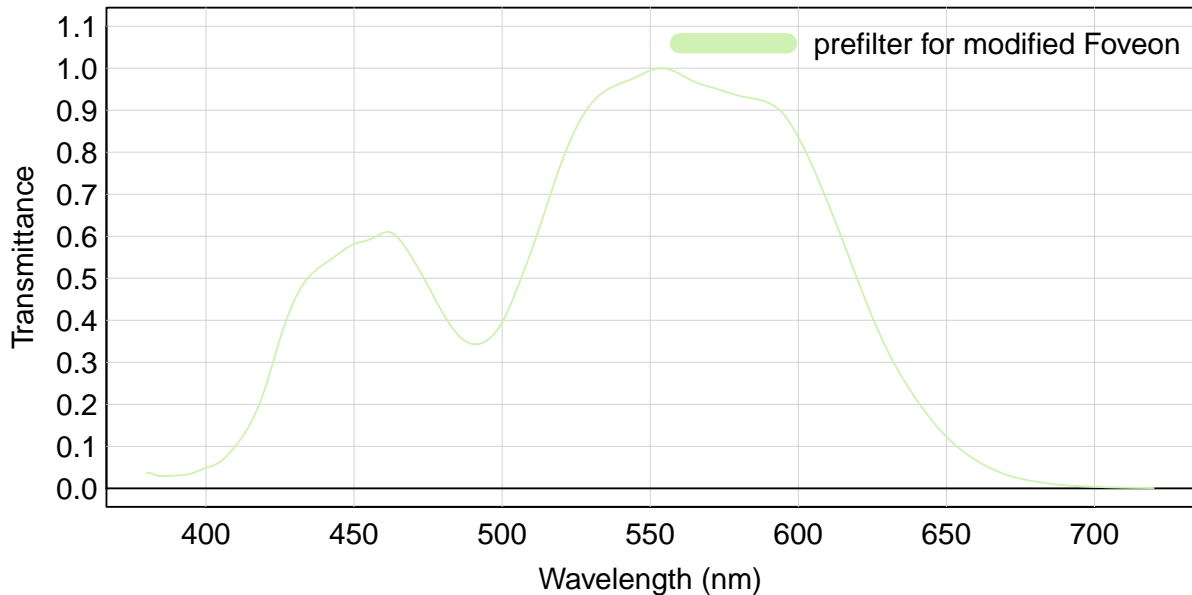
Figure 5: Prefilter for the modified Foveon camera

This curve is broadly similar to those in [1] and [2]. All are bimodal and have a valley near 500nm. But the peaks are in different locations, shaped differently, and both of their peaks have a maximum of 1. I suspect that they are different because of different optimization criteria. The function `emulate()` uses a simple least-squares criterion with the same weight at every wavelength. [1] uses a "Metamerism Index" defined in [6]. This index uses color targets which *might* be the same as those in the *ColorChecker*. [2] uses a criterion based on principal angles between subspaces. These optional criteria might be added to `emulate()` in the future. For a good comparison of the other 2 prefilters, see Figure 6 in [2].

A real engineering implementation of these modifications would have to include a noise and sensitivity analysis. We will not pursue that here, except to observe the condition number of the matrix.

```
A = attr(foveon.mod,"emulate")$A    # A is the 3x3 matrix already printed above
kappa( A, exact=TRUE, norm='2' )    # kappa() returns the condition number of A

[1] 14.27221
```

This is quite large so that is not a good sign.

2 Red Epic Dragon and Plumbicon

The plumbicon, introduced in 1965, is a graylevel television camera tube. The Red Epic Dragon, announced in 2013, is a modern high-speed cinema RGB camera with 19.4 Megapixel CMOS sensor. We will find a good linear combination of the RGB responsivities of the Dragon to emulate the graylevel responsivity of the plumbicon.

Create a fixed wavelength vector, and resample both cameras to the same wavelengths. Then calibrate and plot both cameras.
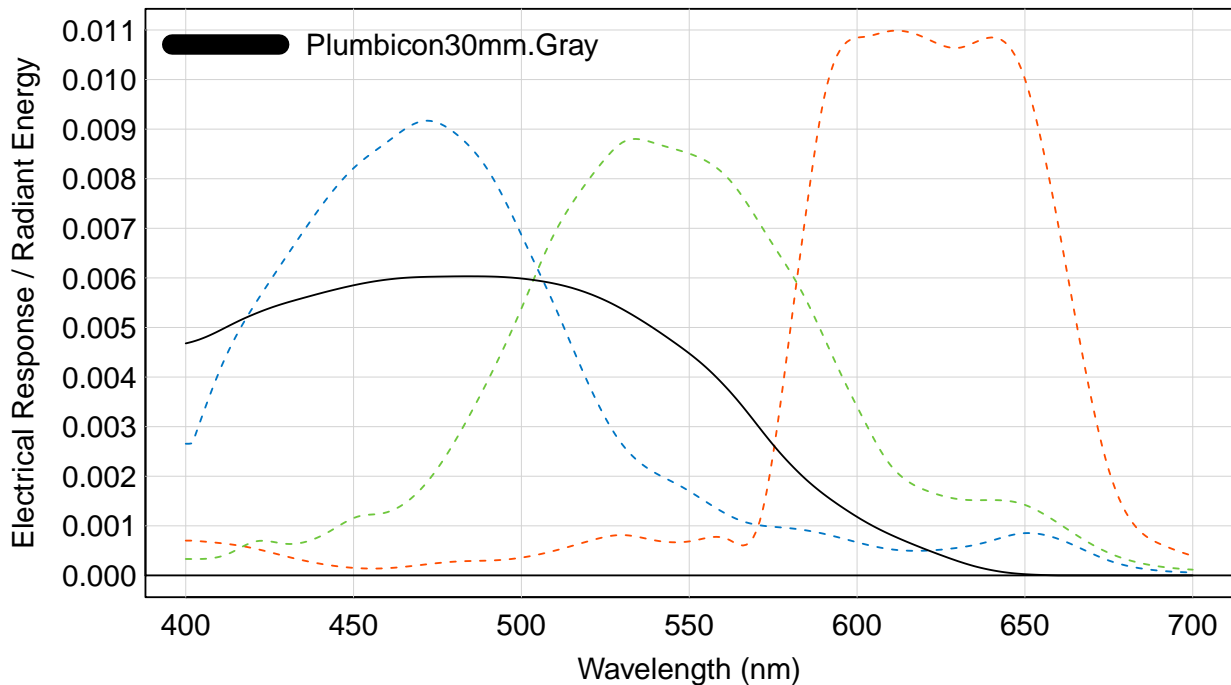
Figure 6: plumbicon (solid) vs Dragon (dashed)

```
wave = 400:700
# read the 2 cameras
path = system.file( 'extdata/cameras/Plumbicon30mm.txt', package='colorSpec')
plumbicon =  readSpectra( path, wave=wave )
path = system.file( 'extdata/cameras/Red-Epic-Dragon.txt', package='colorSpec')
dragon = readSpectra( path, wave=wave )
# calibrate to normalize the response to Illuminant E
illum = illuminantE(wave=wave)
plumbicon = calibrate( plumbicon, stimulus=illum )
dragon = calibrate( dragon, stimulus=illum )
# plot both for comparison
par( omi=c(0,0,0,0), mai=c(0.5,0.9,0.1,0) )
plot( dragon, main='', lty=2, legend=FALSE )
plot( plumbicon, col='black', lty=1, add=TRUE, legend='topleft'  )
```

The integral of all 4 curves is 1. Now matrix the Dragon camera to emulate the plumbicon. A filter is not used here, since the plumbicon has only one output channel, the problem is underdetermined and we could get an *exact* match with a filter.

```
dragon.mod = emulate( dragon, plumbicon, filter=FALSE, matrix=TRUE )
specnames( dragon.mod ) = "Dragon, matrixed"
combo = bind( plumbicon, dragon.mod )
par( omi=c(0,0,0,0), mai=c(0.5,0.9,0.2,0) )
plot( combo, main='', lty=c(1,2), col='black' )
```
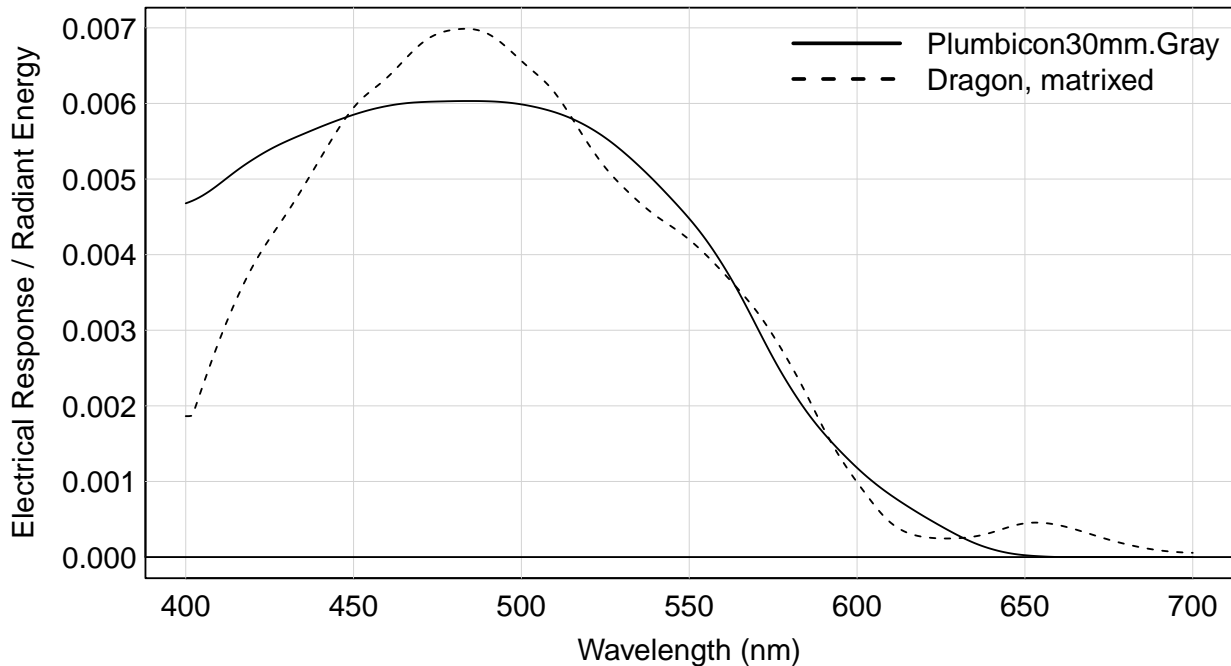
Figure 7: Plumbicon camera (solid) vs the modified Dragon camera (dashed)

The match on the interval [400,500] is not good. The RGB weights are attached to `dragon.mod` and easy to display. Note that the red weight is small.

```
t( attr(dragon.mod,"emulate")$A )

                            R          G          B
Plumbicon30mm.Gray  -0.06454983  0.3641843  0.6736629
```

Calculate the *ColorChecker* graylevel responses from both cameras and display them.

```
MacbethCC = resample(MacbethCC,wave=wave)
graylevel = product( illum, MacbethCC, plumbicon)
RGB.plumbicon = matrix( graylevel, length(graylevel), 3 )
df.plumbicon = extradata(MacbethCC)
df.plumbicon$RGB = RGB.plumbicon
par( omi=c(0,0,0,0), mai=c(0,0,0,0) )
plotPatchesRGB( df.plumbicon, space='sRGB', which='scene', back='black' )
# repeat with dragon.mod camera, and add to existing plot, as triangles
graylevel = product( illum, MacbethCC, dragon.mod)
df.dragon = extradata(MacbethCC)
df.dragon$RGB = matrix( graylevel, length(graylevel), 3 )
plotPatchesRGB( df.dragon, space='sRGB', which='scene', add=T, shape='bottomright' )
```
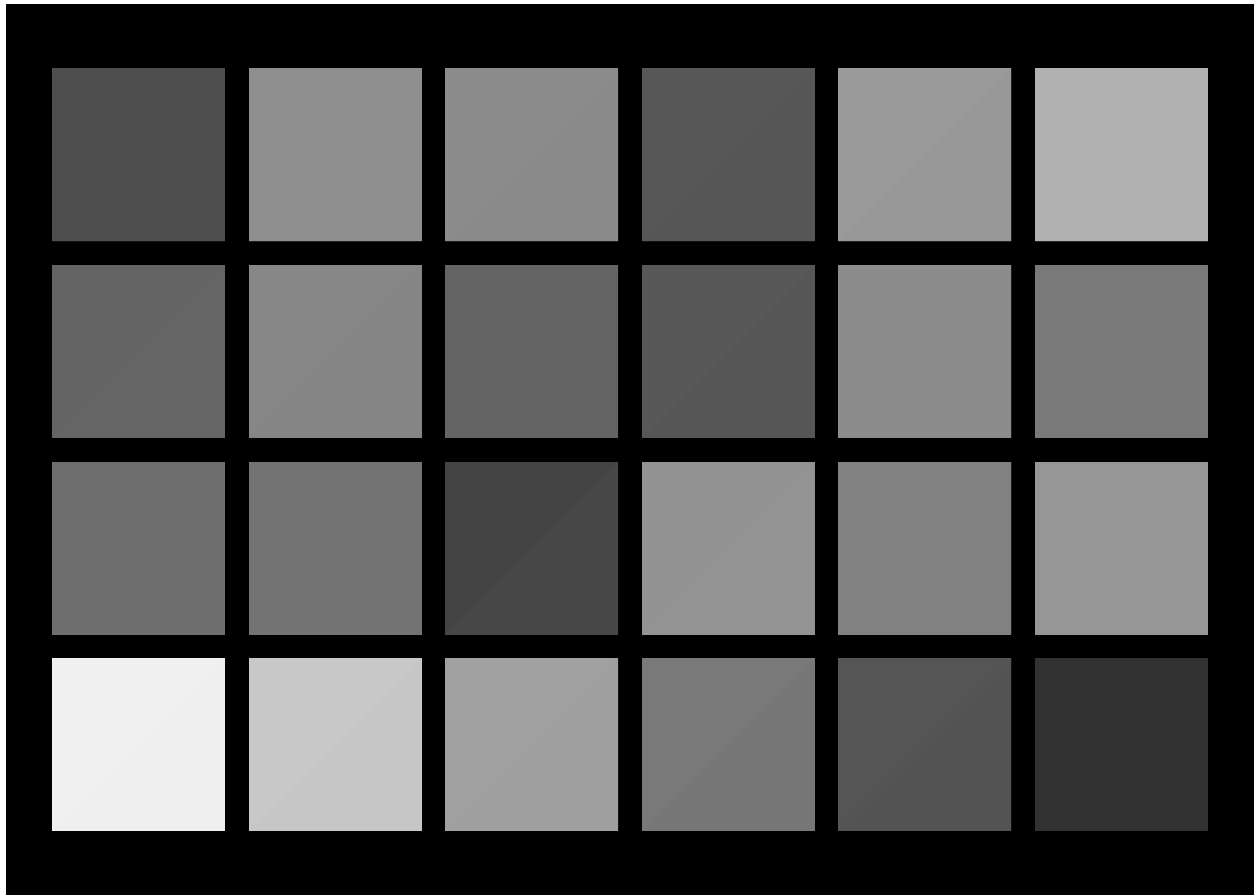
Figure 8: Rendering with Illuminant E, with matrixed Dragon in bottom right half

Despite the mismatch on the interval [400,500], the visual agreement is pretty good.

References

[1] R. F. Lyon and P. M. Hubel, "Eyeing the camera: Into the next century," in *in Proc. IS&T/SID 10th Color Imaging Conference*, vol. 10, (Scottsdale, AZ, USA), pp. 349–355, 2002.

[2] S. Bezryadin, "Quality criterion for digital still camera," in *Proceedings SPIE*, vol. 6502, 2007.

[3] Wayne E. Bretl, "Viewing 1950s Color, Over 50 Years Later," 2008. http://www.bretl.com/Viewing

[4] D. Pascale, "The ColorChecker, page 2." `http://www.babelcolor.com/colorchecker-2.htm`.

[5] G. D. Finlayson and M. S. Drew, "Constrained least-squares regression in color spaces," *Journal of Electronic Imaging*, vol. 6, pp. 484–493, October 1997.

[6] ISO/17321, "Graphic technology and photography – Colour characterisation of digital still cameras (DSCs) – Part 1: Stimuli, metrology and test procedures," standard, International Organization for Standardization, Geneva, CH, 2012.

Appendix

This document was prepared January 27, 2024, with the following configuration:

- R version 4.3.2 (2023-10-31 ucrt), `x86_64-w64-mingw32`

- Running under: `Windows 10 x64 (build 19045)`

- Matrix products: default

- Base packages: base, datasets, grDevices, graphics, methods, stats, utils

- Other packages: colorSpec 1.5-0, knitr 1.42, spacesRGB 1.5-0

- Loaded via a namespace (and not attached): MASS 7.3-60, R6 2.5.1, bslib 0.4.2, cachem 1.0.8, cli 3.6.1, compiler 4.3.2, digest 0.6.31, evaluate 0.21, fastmap 1.1.1, highr 0.10, htmltools 0.5.5, jquerylib 0.1.4, jsonlite 1.8.4, microbenchmark 1.4.10, rlang 1.1.1, rmarkdown 2.21, sass 0.4.6, tools 4.3.2, xfun 0.39, yaml 2.3.7