ICCTrans: ICC-conversion of colors

Ignacio Ruiz de Conejo, Marti Maria
Version 1.0, 23-Nov-04

ICCTrans version for Matlab

This tool will apply an ICC-based color conversion to a Matlab variable. As mentioned above, running this command without parameters will display information on the intended usage for this command:

```
>> icctrans
(LMX) littlecms ColorSpace conversion tool - v0.1 BETA

usage: icctrans (mVar, flags)

mVar : Matlab array.

flags: a string containing one or more of the following options.
  -v           – Verbose
  -i<profile>  - Input profile (defaults to sRGB)
  -o<profile>  - Output profile (defaults to sRGB)
  -l<profile>  - Transform by device-link profile
  -m<profiles> - Apply multiprofile chain
  -t<0,1,2,3>  - Intent (0=Perceptual, 1=Colorimetric, 2=Saturation, 3=Absolute)
  -b           - Black point compensation
  -c<0,1,2,3>  - Precalculates transform (0=Off, 1=Normal, 2=Hi-res, 3=LoRes)
[defaults to 0]
  -p<profile>  - Soft proof profile
  -r<0,1,2,3>  - Soft proof intent

You can use following built-ins as profiles:
  '*sRGB'   -> IEC6 1996-2.1 sRGB
  '*Lab'    -> D50 based Lab
  '*LabD65' -> D65 based Lab
  '*XYZ'    -> XYZ (D50)
  '*Gray22' -> D50 gamma 2.2 grayscale.

For suggestions, comments, bug reports etc. send mail to
marti.maria@hp.com or ignacio.ruiz-de-conejo@hp.com

>>
```

Let us explore what all these options are, and what may deliver to you. Let us open an image and display it on the screen:

```
>> a = imread('test.tif');
>> image(a)
```

Simple color transformations

Now, assume this image was an sRGB image, and we want to convert it into ECI RGB. We just need to convert with ICCTrans, providing the proper command line to it:

```
>> b = icctrans(a, '-t1 –o ciergb.icc');
```

Instead of the relative colorimetric (-t1), we can use the absolute colorimetric rendering intent (-t3) for the transformation, and then display the obtained image:

```
>> b = icctrans(a, '-t3 –o ciergb.icc');
>> image(b)
```

Notice that our command line is assuming the image is in the default colorspace, which is sRGB. Otherwise, we would need to specify with the '-i' option. If we want to display the original and the modified image[1], we just tell Matlab to do so:

```
>> image([a,b])
```

Figure 1: the original image in sRGB (left) and its ECIRGB abscol transformation (right).

The same color transformation can be applied not only to images, but to matrixes, that represent color values. The following examples show the CIERGB color corresponding to a single sRGB color, and to an array (m) of Lab50 values :

```
>> icctrans([0.1 0.2 0.3], '-t0 –o ciergb.icc')
ans =
      0.1012      0.2201      0.3062
>> m = [ .1 .2 .3; .2 .3 .4; .3 .4 .5; .4 .5. .6]
>> icctrans(m, '-t1 –i *Lab –o ciergb.icc')
```

An image can also be transformed to printer space, and saved for later use:

```
>> d = icctrans(a, '-t0 –o HP2500CoatedBest.icc')
>> imwrite(d, 'testCMYK.tif');
```

Error handling

If the profile to be used is not found, Matlab will return with a documented error:

```
??? File 'ciergb.icc' not found

Error in ==> C:\MATLAB6p1\extern\examples\lcms\icctrans.dll
```

and so will be treated all the other possible errors. All of them recover elegantly from an error.

Multi-profile transformations

ICCTrans is able to apply the color transform resulting from the concatenation of several profiles. Parameter '-m' is used for this purpose, and expects that the color spaces between profiles are correctly set:

```
>> icctrans([0.1 0.2 0.3], '-m *sRGB *lab ciergb.icc')
```

Internal color spaces

As you have seen, certain color spaces can be handled internally, without the need of an external profile. This refers, however, to v2 profiles. For version 4 ICC profiles, an external file containing the profile must be specified. Implicitely, we are saying as well that *lcms* supports V4 profiles.

[1] Note: the images shown are meant to give some visual feedback, and are not to be taken as serious renditions of what one can see on the screen.

Some experimentation on mixing v2 and v4 profiles is missing, but this fact is irrelevant to the integration of the color engine. In fact, we hope it will help unveiling hidden anomalies. [Note: this is successfully addressed in version 1.14 of *lcms*.]

Soft proofing

If you want to softproof an image (preview on the screen how the printed output would look like), the '-p' parameter can be used to specify the emulation profile and '-r' to specify the rendering intent of this transformation:

```
>> c = icctrans(a, '-t1 -i sRGB.icc –p swop.icc -o ciergb.icc –r3')
>> image([a,c])
```

Figure 2: the original image in sRGB (left) and a SWOP softproof.

Device to device transforms

Device-links are a standard type of ICC profiles, that provide color transformations from device to device. They have gained popularity since the appearance of CMYKPlus, but they have always existed in the form of colormaps. To apply a CMYKPlus transformation::

```
>> c = icctrans(d, '-l HWCoatedPlus.icc')
```

Note: with *lcms* we could easily create a tool, say ICCLink, to generate device-links at our will.

The other parameters that can be issued in the command line allow the incorporation of the black point compensation (-b) into the color transform, some visual feedback (-v) of what's going on behind the curtains, and a minimal adjustment (-c) of throughput versus accuracy.

A note on Color Coordinates

The coordinates that *lcms* expects for each color space have a valid range. Currently, this range is fixed, and it's the following:

RGB:	0.0 to 1.0
CMYK:	0.0 to 100.0 (that is, ink %)
Lab:	0.0 to 100.0 and -128.0 to +128.0
XYZ:	0.0 to 1.0
Others:	0.0 to 1.0

This could be modified if needed (say 0..255 for RGB, to match Photoshop range) by adding a new optional flag.