

LittleCMS: A free color management engine in 100K.

Background

One of the main components of a color management solution is the Color Matching Module, or CMM, which is the software engine in charge of controlling the color transformations that take place inside the system. Some of these color solutions use a CMM supported by the operating system, such as Apple's ColorSync and Microsoft's ICM, while others take advantage of a proprietary CMM, as is the case of Adobe's Color Engine or Gretag Macbeth's LOGO, to name a few. The aim of this paper is to introduce an open source, cross platform CMM engine that is freely available to everybody at no cost. The original target for this CMM was the applications required to run on several platforms, where different CMMs or no CMM at all were available, without the need of rewriting code and adapt it to the platform's CMM.

Introduction

A Color Matching Module (CMM) is a piece of software that uses measured data to convert and match a color in a given color space on a given device to or from another color space or device, perhaps a device-independent color space. When colors consistent with one device's gamut are displayed on a device with a different gamut, a CMM attempts to minimize the perceived differences in the displayed colors between the two devices. The CMM does this by mapping the out-of-gamut colors into the range of colors that can be produced by the destination device. It uses lookup tables and algorithms for color matching, previewing, color reproduction capabilities of one device on another, and checking for colors that cannot be reproduced.

A more open concept, platform independent view of color management, is the use of an ICC-compatible CMM. The International Color Consortium¹ (ICC) is an industry consortium which has defined an open standard for the CMM, and color profiles (ICC profiles) for the devices and working space. Each ICC profile provides a number of colour transformations that define the colour expected from the encoded data of the digital image, in an open format. In other words, the profile defines the colour to be expected with any set of image values – which are often device values, but may be in some standard colour image encoding. Another ICC concept is to make color profiles a part of file formats like TIFF, JPEG, PNG, EPS, PDF, and SVG.

The color space internally used by ICC profiles is the internationally accepted CIE system² for defining color matches, so using this it is possible to ensure that colors from input will match those on output, for the viewing conditions for which the color is defined. The fact that the format is public means that any ICC compliant system should be able to use these profiles to interpret the colour intended for that digital image.

The basic way in which ICC profiles are typically used to achieve colour reproduction is by combining a source profile with a destination profile to enable input device data to be transformed to that required to give the required colour on output. The combining of the profiles is performed by a CMM, which can be provided at various places in the workflow. In some reproduction procedures there may be more than two profiles used, or even special cases where only one is used that has been constructed by combining a source and destination profile.

CMM availability

Unfortunately, very few CMM are available. As per today (April 11, 2006) only 15 CMM have been registered worldwide with the ICC³. This is a very small quantity, when considering the number of existing color managed applications available. The reason for such limited number may be the inherent complexity of this software component. Some platforms like Mac and Windows already have a CMM integrated at system level; this solves the issue to some extent. But for application software intended to run on several operating systems, that also means a different set of code should be maintained for each supported platform. In some cases, results may vary across different CMM and this negatively affects color consistency. Cross-platform CMMs are certainly possible, but those are proprietary tools, and the cost of developing such software component is often prohibitive for small companies.

Our solution

In order to overcome such problem, and in the hope this would popularize ICC color management, we have developed a tiny CMM and made it freely available under an extremely liberal open source license. Our CMM is named LittleCMS in regard of its small overhead. With a typical footprint of about 100K, it includes all necessary code to create and apply color transforms, either on raster data or by others means like using ASIC hardware or PostScript as host language. LittleCMS is free under the MIT license agreement⁴. That has made LittleCMS very popular, as it has been readily adopted by open source community as well as some commercial companies.

The main goal of this engine is applying color management. Hence, by design, it does not include profile management functionality (installation, association with devices), nor profiling (generation of profiles from device measurements). The main purpose of its functions is to gain access to profiles, create color transformations and apply them to objects, either numbers or images.

The declared goals of this architecture, sorted by priority, are:

- Easy to use and learn.
- Fast.
- Embeddable
- Thread safe.
- Small, with few required dependencies.

Other basic assumptions of LittleCMS are:

- Memory management to be transparent
- Error management to be very simple
- Enforce strong encapsulation

Any developer that has supported color applications for the main operating systems will feel comfortable right away, with the additional benefit of its simplicity of usage. The CMM should be fast and robust enough to be used in production code. And this is actually the case, since several vendors have already adopted this solution and there are devices that already incorporate LittleCMS in their firmware.

Features

The Key feature of LittleCMS is portability. A portable program is one that you can move with little or no extra investment of effort to a computer that differs from the one on which you originally developed the program. LittleCMS Code is written in ISO C99⁵ in order to improve portability, although writing a program in Standard C does not guarantee that it will be portable; you must be aware of the aspects of the program that can vary among implementations. LittleCMS has been proved to be portable across many different platforms including big and little endian, 32 and 64 bits, x86, Alpha and ARM processors among others. In order to improve portability and simplify the installation procedure, LittleCMS uses Autoconf⁶. Autoconf is a tool for producing shell scripts that automatically configure software source code packages to adapt to many kinds of UNIX-like systems. The configuration scripts produced by Autoconf are independent of Autoconf when they are run.

The CMM is organized as a software library; able to be used either as a shared object or statically linked. It may be called directly from C and C++, or additionally, the programmer may use wrappers for other languages like Delphi, Python or Matlab. Those wrappers are already included in LittleCMS package. The ability of hosting scripting languages is very important in order to prototype workflows.

Another important aspect is performance. Any CMM able to be used in production code should be fast enough to fit user expectative. Unfortunately, portability is often a stopper for complete optimization, so a compromise has to be taken to accomplish both goals.

There are three different build modes for maximizing compatibility:

- ASM mode for fastest operation (only works on x86 processors),
- C-only fixed point for obtaining reasonable speed and big portability,
- "Float point" mode for processors with lightning-fast floating point math.

Special care has been taken on minimizing required resources. The underlying philosophy "small is beautiful" fits very well on embedded systems hence is very adequate for firmware.

Another key point is smooth learning curve. A competent programmer should be able to integrate LittleCMS in few days.

The v2 ICC profile format specification has been widely adopted by the colour imaging community and proved very important in achieving and maintaining colour fidelity of images. However, despite its successful usage in many situations this widespread use has also identified ways in which it can be even further improved. That was the main driving force behind the v4 revision of the specification (dated December 2001) - in particular ways to improve interoperability. Certain ambiguities in the previous versions of the specification have occasionally permitted producers of profiles to misinterpret the reference colour space and also the information they need to provide in the profile. Thus profiles could be produced that were inconsistent with those produced by other vendors and when two such profiles are used together can give rise to unexpected results. Furthermore, these ambiguities permitted ICC compliant profiles to be produced that

were interpreted slightly differently when used with different CMMs. LittleCMS does support the latest 4.2 ICC specification.

Aside the basic functionality, LittleCMS offers other advanced features like a CGATS parser, CIECAM02 appearance model implementation, profile writing capabilities and non-ICC transforms like K-preserving on CMYK->CMYK workflow.

Several color management tools have been developed already with this CMM. A first set are utilities that apply color transformations to specific graphic file formats (TIFF, JPEG), with profiles and options specified in the command line. Other utilities include a profile linker to generate devicelink profiles and other checking tools. Those utilities were primarily intended as samples, but many people keeps using them on it own.

Usage

The basic way in which ICC profiles are typically used to achieve colour reproduction is by combining a source profile with a destination profile to enable input device data to be transformed to that required to give the required colour on output. Selection of the appropriate transforms, by selection of the rendering intent, enables the desired reproduction to be achieved. The combining of the profiles is performed by the CMM, which can be provided at various places in the workflow (such as the image editing software, raster image processor or printer driver, among others).

LittleCMS defines two kinds of structures that are used to manage the various abstractions required to access ICC profiles. These are profiles and transforms.

In a care of good encapsulation, these objects are not directly accessible from a client application. Rather, the user receives a 'handle' for each object it queries and wants to use. This handle is a stand-alone reference; it cannot be used like a pointer to access directly the object's data.

- cmsHPROFILE identifies a handle to a profile.
- cmsHTRANSFORM identifies a handle to a transform.

Programmer gets those handles by calling the adequate functions. For example:

```
hsRGB = cmsOpenProfileFromFile("sRGB.icc", "r");
```

That would be the way to open a disk-based ICC profile. The function has a second parameter to specify access mode. In this example we are open the profile for reading. Further references to this profile are done by using the 'hsRGB' handle. We can open as many ICC profiles as we wish, maximum amount of simultaneous profiles are restricted only by operating system resources. Opening profiles uses few memory, big tables are loaded on demand when creating color transform. We can open any type of ICC profile in the same way, for example, a CMYK profile:

```
hCMYK = cmsOpenProfileFromFile("CMYK.icc", "r");
```

Once profiles are open, we can create a color transform by joining them. This logical unit does identify how pixel translation will take place. To identify a color transform, some parameters must be specified. Of course, the involved ICC profiles. Other parameter is the rendering intent, or the way the CMM should deal with different gamut. LittleCMS does support all ICC rendering intents: Perceptual, Saturation, Relative colorimetric and Absolute colorimetric.

Other required information is the way pixel buffers are encoded. LittleCMS can handle a lot of formats. In fact, it can handle:

- 8 and 16 bits per sample
- up to 15 channels
- extra channels like alpha
- swapped-channels like BGR
- endian-swapped 16 bps formats like PNG
- chunky and planar organization
- Reversed (negative) channels
- Floating-point numbers

In describing such formats, LittleCMS does use a 32-bit value, referred as “format specifier”. The format specifiers are useful above color management. This will provide a way to handle a lot of formats, converting them in a single, well-known one. For example, if you need to deal with several pixel layouts coming from a file (TIFF for example), you can use a fixed output format, say `TYPE_BGR_8` and then, vary the input format on depending on the file parameters.

There are several (most usual) encodings predefined as constants. As an example, here are some:

```
TYPE_GRAY_8, TYPE_GRAY_16, TYPE_GRAYA_8, TYPE_RGB_8
TYPE_BGR_8  TYPE_RGB_16  TYPE_BGR_16  TYPE_RGBA_8
TYPE_RGBA_16,TYPE_ABGR_8,  TYPE_ABGR_16
TYPE_CMY_8,  TYPE_CMY_16,  TYPE_CMYK_8,
TYPE_CMYK_16, TYPE_KYMC_8  TYPE_KYMC_16
TYPE_XYZ_DBL  TYPE_Yxy_DBL, TYPE_Lab_DBL
TYPE_CMYKcm_8, TYPE_CMYKcm_16
...
```

The “_8” and “_16” postfix identifies the pixel depth. Only 8 and 16 bits per sample are supported. 11, 12 or 14 bits would imply serious performance penalties, and would add unwanted complexity. The “_DBL” postfix identifies double (floating point) values. These are intended mostly for computation and are not as fast as the encoded counterparts.

Example

The best way to expose how this is intended to work seems to be by an example. So, here is an example to show, step by step, how a client application can transform a bitmap between two ICC profiles using the LittleCMS API

```
#include "lcms.h"

int main(void)
{

    cmsHPROFILE hInProfile, hOutProfile;
    cmsHTRANSFORM hTransform;
    int i;

    hInProfile = cmsOpenProfileFromFile("HPSJTW.ICM", "r");
    hOutProfile = cmsOpenProfileFromFile("sRGBColorSpace.ICM", "r");

    hTransform = cmsCreateTransform(hInProfile,
                                   TYPE_BGR_8,
                                   hOutProfile,
                                   TYPE_BGR_8,
                                   INTENT_PERCEPTUAL, 0);

    for (i=0; i < AllScanlinesTilesOrWatseverBlocksYouUse; i++)
    {
        cmsDoTransform(hTransform, YourInputBuffer,
                      YourOutputBuffer,
                      YourBuffersSizeInPixels);
    }

    cmsDeleteTransform(hTransform);
    cmsCloseProfile(hInProfile);
    cmsCloseProfile(hOutProfile);

    return 0;
}
```

1

2

3

4

1. Open the required profiles. You will need the profile handles for create the transform. In this example, I will create a transform using a HP Scan Jet profile as input, and sRGB profile as output.
2. Create the transform. When creating transform, you are giving to LittleCMS all information it needs about how to translate your pixels. You give the profile handles, the format of your buffers, the rendering intent and a combination of flags controlling the transform behavior.
3. Next, you can translate your bitmap, calling repeatedly the processing function. This function is intended to be quite fast. You can use this function for translating a scan line, a tile, a strip, or whole image at time.
4. Last, free all stuff.

Conclusions and Availability

In this paper we have presented a freely available CMM in open source form. It is fast, reliable and mature enough to be used in production code. Many open source and commercial software systems are using it, and have become as the “de facto” CMM for Linux systems. Further information, as well as the package itself can be found at <http://www.littlecms.com>

Author Biography

Marti Maria is a color engineer in the imaging and color group at the large format printer division of Hewlett-Packard. He has as degree in telecommunication engineering from the Universitat Politècnica de Barcelona. His research interests span various color imaging technologies and techniques for computer based graphics.

¹ International Color Consortium. Specification ICC.1:1998-09, File Format for Color Profiles, Available from <http://www.color.org/>).

² Commission International de l'Eclairage. Colorimetry. Publication CIE No 15.2, 1986. Available from <http://www.cie.co.at>

³ International Color Consortium. Private and ICC Tag and CMM Registry Available from <http://www.color.org/privatetag2005-12.pdf>

⁴ The MIT license agreement. Available from <http://www.opensource.org/licenses/mit-license.php>

⁵ In 1999, ISO/IEC 9899 (generally known as C99) Available from <http://www.open-std.org/jtc1/sc22/wg14/>

⁶ Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA <http://www.gnu.org/software/autoconf/>