ICC DevCon 2008

Floating point processing with ICC profiles

Max Derhak
Onyx Graphics

Overview

- **The new ICC floating point capability will be discussed,**
- **Discussion of the ICC specification change to add floating point will include:**
  - capabilities in the new MPE tag type,
  - details of the specification change and examples,
  - and significant technical implementation details with example profiles.
- **along with its implied Programmable CMM capabilities.**
- **An example of encoding spectral data in an ICC profile will be shared,**
- **Guidance will be given on using a probe profile in Photoshop to evaluate whether various transforms are executed.**
- **along with additional usage recommendations for MPE based tags.**

Background

- The initial motivation for the Floating Point Encoding Range proposal stems from an attempt to use ICC color profiles for managing colors in Motion Picture and Digital Photography Workflows

Problems with using ICC profiles

- The precision of profile elements such as LUTs, curves, and matrices is insufficient for the motion picture industry processing because:
  - Encoding is limited to 16 bits and therefore transform inversion cannot be performed precisely due to quantization errors
  - Current profile transforms only support bounded device-side color encodings (IE [0, 1]), but unbounded (floating point) encodings are used in the motion picture industry
  - PCS encoding is limited to encoding range of [0, 2) for XYZ values or [0, 100] for L* and [-128, 127] for a*, and b*
    - NOTE: There has been some confusion regarding ICC support of above-white device values. Such values can also be supported by using above-white media white points and using the absolute rendering intent. Clarifying language is provided in the Colorimetric Intent Image State (CIIS) amendment.

The Floating Point Encoding Range amendment proposal

- **Floating Point Encoding Range amendment proposal was approved by the ICC in November 2006**
- **Defines optional D2Bx/B2Dx tags based upon a new tag type**
  - The Multiple Processing Elements Tag Type (with ‘mpet’ type signature)
- **Key Purposes:**
  - Overcome limited precision in ICC profiles by allowing for the direct encoding of floating point data in an ICC profile
  - Remove bounding restrictions for both device side and PCS side encoding ranges
  - Provide for backwards compatibility to the existing ICC profile specification

Floating Point Device Encoding Range

Amendment provides:

Optional DToBx / BToDx tag transforms

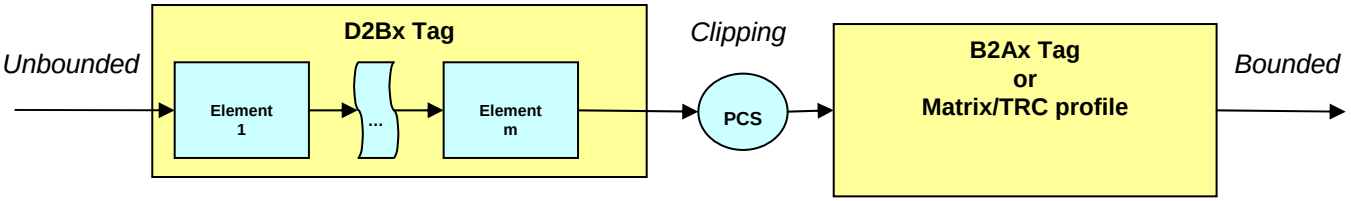
- Not bounded to fixed device range
- Negative values allowed
- PCS essentially the same, except now encoded using floating point
- Direct encoding of absolute rendering intent with D2B3/B2D3 tags
- All required tags must still be present in addition to new optional tags

These new tags use the Multi Processing Elements Tag Type (‘mpet’)

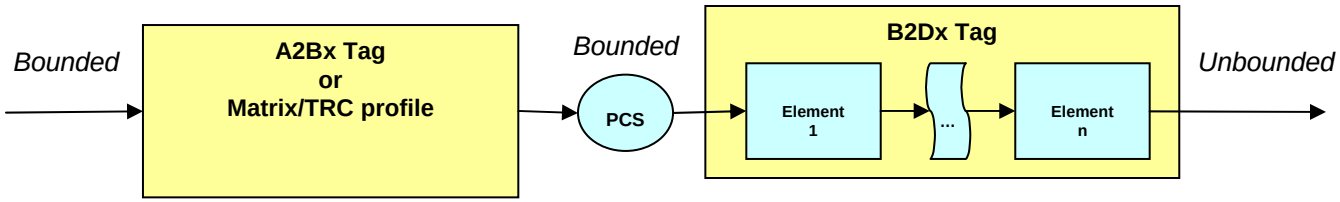
- Each tag can contain an arbitrary sequence of processing elements to define a transform from device to PCS, PCS to device, or device to device

MPE Tag Connectivity

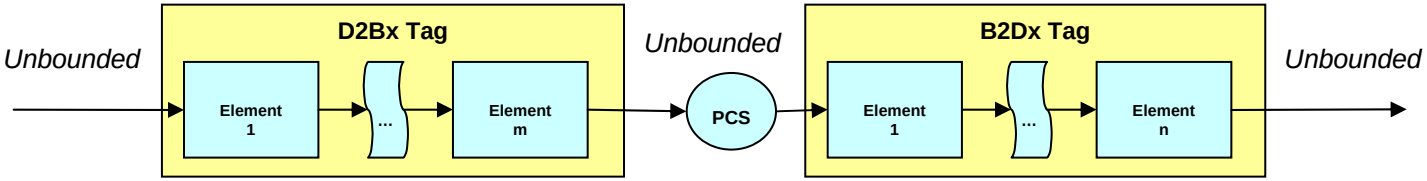
Case 1



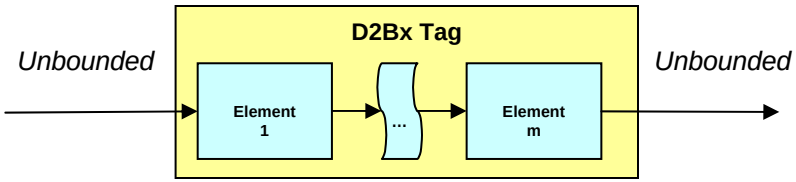
Case 2



Case 3



Case 4



MPE Tag Type Overview

- **MPET Tag Type support is OPTIONAL**
  - MPET based tags are NOT required to supported by CMM implementations in general!!!!!!
- **DToBx / BToDx tags contain a sequence of processing elements to define the transform from device to PCS or PCS to device**
- **Processing elements use 32-bit IEEE 754 floating-point encoding**
- **Initial repertoire of processing elements:**
  - N-Dimensional lookup tables
  - NxM matrices
  - One dimensional segmented curve sets.
- **Other processing element types could be added in future**

MPET Tag Type Overview (continued)

- **Sequence of processing elements is arbitrary**
  - Provides for the implementation of a limited Static Programmable CMM (discussed later)
- **The CMM performs NO manipulation of data between elements**
- **Output channels from preceding elements are direct inputs to succeeding elements**
- **Up to 65535 channels can be passed between elements**
- **Device input and output limited to 16 channels to be compatible with what can be encoded in profile header**
- **Legacy Compatibility**
  - Connectivity with legacy profiles
  - Compatible (but unbounded) PCS encoded in floating-point
  - Fall back behavior defined

MPET Elements

- **CLUT Element**
  - Stores N-dimensional lookup tables
  - Up to 16 input channels
  - Output up to 65535 channels
  - Input range is from 0.0 to 1.0 (clipping as needed)
    - Note: A separate processing element may need to be placed before a CLUT element to scale to the CLUT element input range
  - Output range is the entire floating-point encoding range.

MPET Elements (continued)

- **Matrix Element**
  - NxM matrix with a constant offset vector
  - The input and output dimensions need not be the same
  - Up to 65535 channels can be used for both input and output
  - The input and output range is the entire floating-point encoding range

MPET Elements (continued)

- **Curve Set Element**
  - Encodes one dimensional curves for each input/output channel
  - Up to 65535 separate curves can be defined
  - The input and output range is the entire floating-point encoding range
  - The curves are segmented:
    - One to 65535 segments are possible for each curve
    - Domain of each segment is encoded by break points
    - Each segment can be defined as one of the following:
      1. Formula segments
         - Define a function type and provides parameters to the function
         - Beginning and ending segments must be formula segments
      2. Sampled segments
         - Equally spaced sample points define a 1 dimensional look up table
         - First interpolation point is NOT stored in sampled segment.
         - Interpolation is performed from last point of previous curve segment.

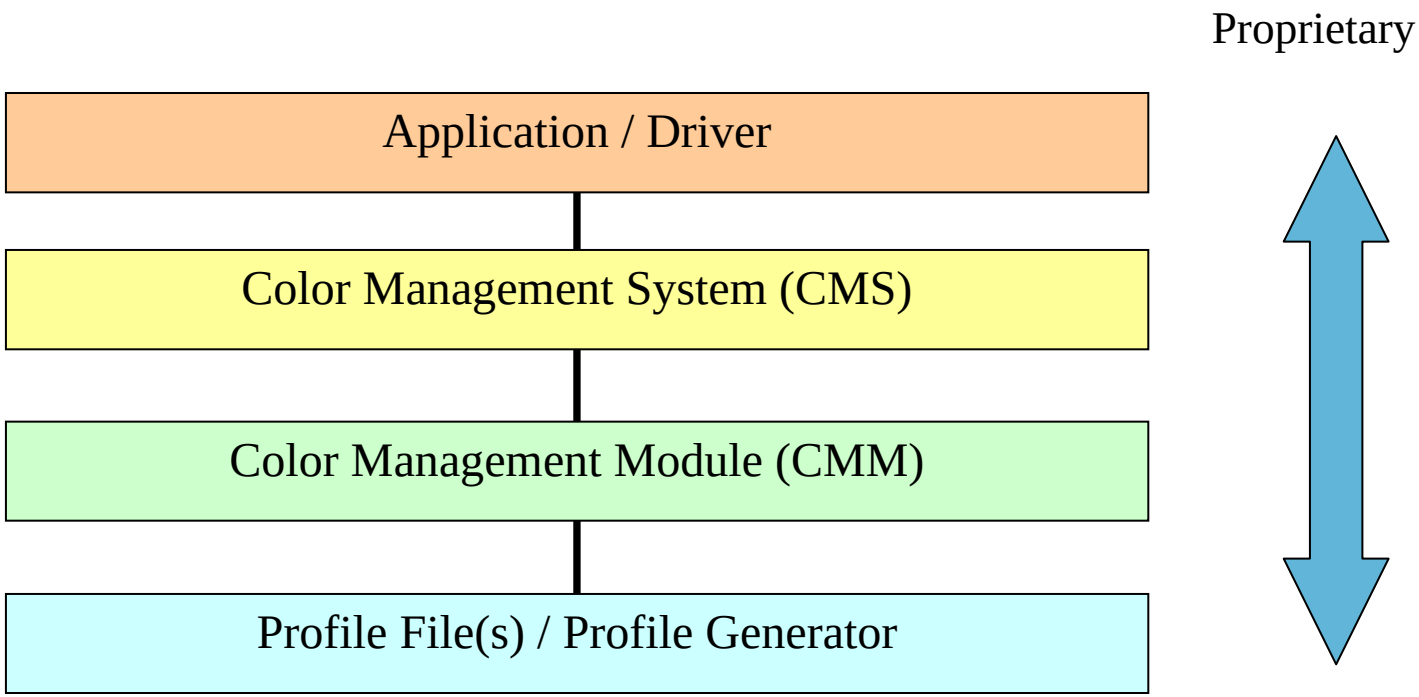
MPE Example: A D2Bx tag

- **A D2Bx tag with the following elements provides Device to PCS conversion:**
  1. A curve set to normalize input from device
  2. A matrix or N-Dimensional lookup table that converts Device data to 31 channel spectral information
  3. A 3x31 matrix to convert spectral information to XYZ colorimetric information
     - ➢ If the PCS signature in the header is ‘XYZ ‘ then we are done
  4. A curve set to perform the cubic root portion of an XYX to Lab conversion
  5. A 3x3 matrix to complete the XYZ to Lab conversion

- **Note: The CMM acts like a programmable transform interpreter since it has no understanding of what each processing element is trying to accomplish.**

Static vs Dynamic vs Programmable CMMs

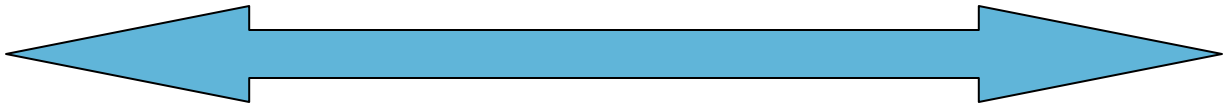
Color Management Layers



Color Management Implementation (CMI) Continuum

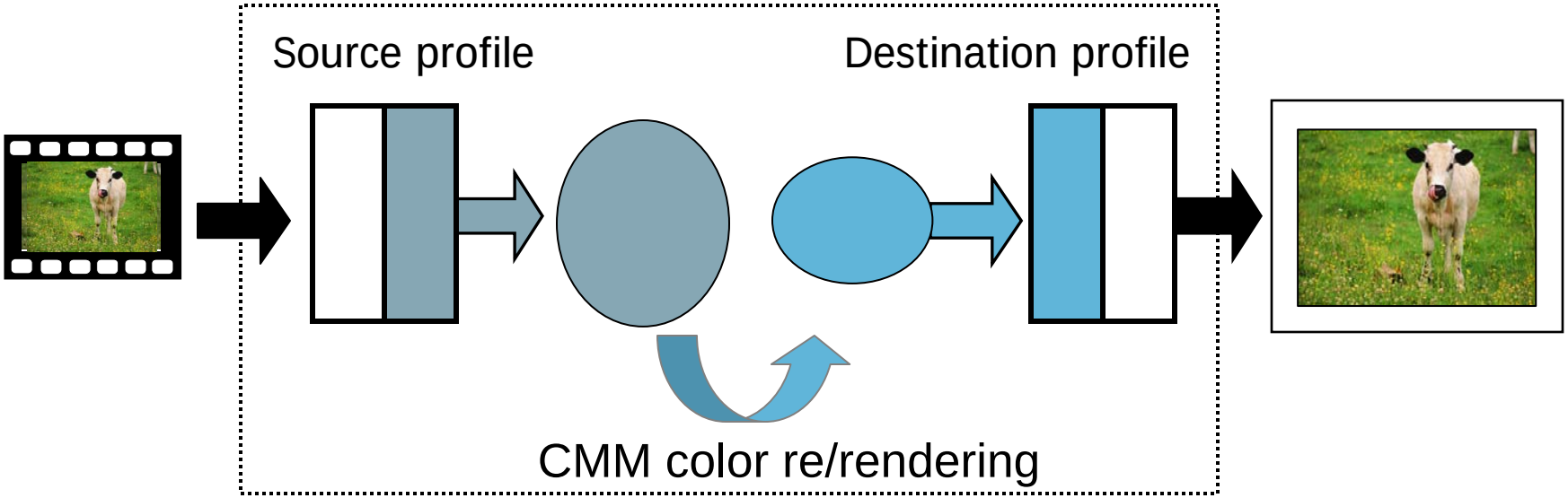
Static
- Smart Profiles
- Dumb CMM (basically an interpolation engine)

Dynamic
- Dynamic transforms
- Smart CMM



“Smart” Operations
- Device Characterization
- Function Inversion
- Rendering and Re-rendering
- Black Point Compensation
- Gamut Mapping
- Color Appearance Modeling
- Scaling
- Black Generation
- Channel preservation
- Proprietary operations

“Dynamic” CMM possibilities



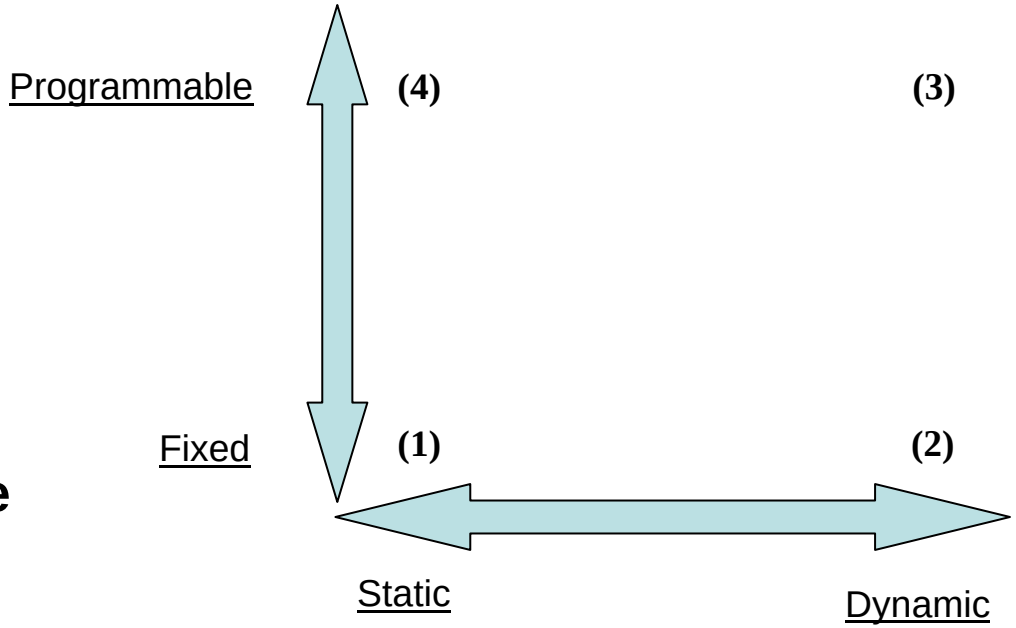
CMM algorithms color re-render source image colorimetry to be appropriate for actual output medium at run time

- considers source and output medium gamuts and viewing conditions
- supports color appearance model based color re-rendering
- can take advantage of full output medium gamut
- facilitates user adjustment of color re-rendering at time of output

“Programmable” CMM

- Each MPET tag defines a fixed sequence of processing elements to apply
- Purpose of each element is unknown to the CMM
- No dynamic decision making involved in applying elements – the operation is static
- Implies an additional “Programmable” dimension in the Color Management Implementation Continuum

Revised CMI Continuum



CMI Categories

1. **Static Fixed**
2. **Dynamic Fixed**
3. **Dynamic Programmable**
4. **Static Programmable**

Addition of Programmable / Fixed dimension provides four general categories of Color Management Implementation

MPE: SampleICC implementation

- **SampleICC is an open-source library of colour management functions**
- **SampleICC has added MPE support as follows:**
  - Implemented as additions to SampleICC's IccProfLib library
  - New CIccCmmMPE() class derived on CIccCmm provides access to apply new D2Bx/B2Dx tags when present.
  - MPE tags implemented through CIccTagMPE and CIccMpe based classes
  - ApplyNamedCmm, wxProfileDump, & lutA2BtoMPE applications
  - lutA2BtoMPE app was created to allow conversion of lutA2B based profile to MPE based profile for testing purposes.
  - Third party application support through Windows SampleICC CMM
  - The extendible CIccElemCreator factory design pattern class allows for user extension of private element types without modification to IccProfLib

SampleICC CMM Demonstration

The Future

- The ICC specification has established an architecture for interoperable and unambiguous communication of color between devices.
- Going forward, there are several challenges for the ICC:
  - The context for color management is widening, and industries and devices beyond the traditional color management users are emerging. The original ICC architecture was not optimal for all of these industries and devices, but now the main challenge is to promote adoption of the v4 spec and the recent amendments.
  - The color processing models defined in the ICC architecture can be significantly extended to support a wider range of color transforms. This needs to be done in a way which continues to be open, interoperable and vendor-neutral

Summary

- Digital Motion Picture workflows have a better chance at being supported by ICC colour management through new technology tags and the Floating Point Encoding amendment
- MPE tags and the unbounded PCS in DToB and BToD tags enable a wide range of transforms previously unsupported by the ICC architecture
- Demonstration of an example implementation and some of the possibilities of MPE tags