Working with RefIccMAX

ICC DevCon 2020 - The Future of Color Management

Max Derhak(PhD)
Principal Scientist
Onyx Graphics, Inc.

Workshop Topics

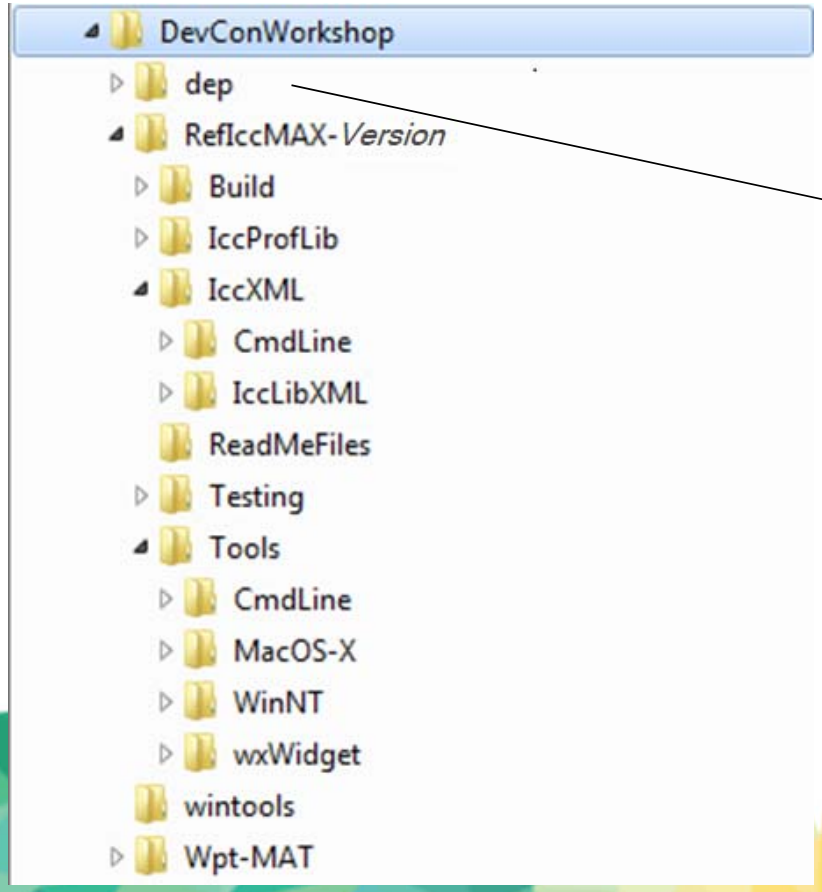
- Becoming familiar with RefIccMAX
  - Install and build libraries and tools
- Using RefIccMAX tools to generate, apply, view and manipulate iccMAX profiles
  - Fun with MetaCow
  - Applying named color profiles
- Developing with RefIccMAX's IccProfLib library
  - How to write code to apply profiles
- Working with different observers and illuminants
  - Understanding CATs, CAMs and MATs
  - Using excell and Matlab to create data for PCCs
  - Applying different observers with Rec2020 monitor profiles
- Manipulating spectral reflectance
  - How it is done
  - More fun with MetaCow
- Exploring workflows with iccMAX profiles
  - Brief look at other profiles in Testing folder (time permitting)
- Wrap-up

What is RefIccMAX?

- RefIccMAX is an open source C++ reference implementation related to the iccMAX specification
  - Includes both libraries and tools
  - Allows for the interaction, creation, manipulation, and application of iccMAX based color management profiles
- Source project accessible through GitHub
  - https://github.com/InternationalColorConsortium/RefIccMAX
- Executables and sample profiles available on ICC web site
  - http://www.iccmax.org

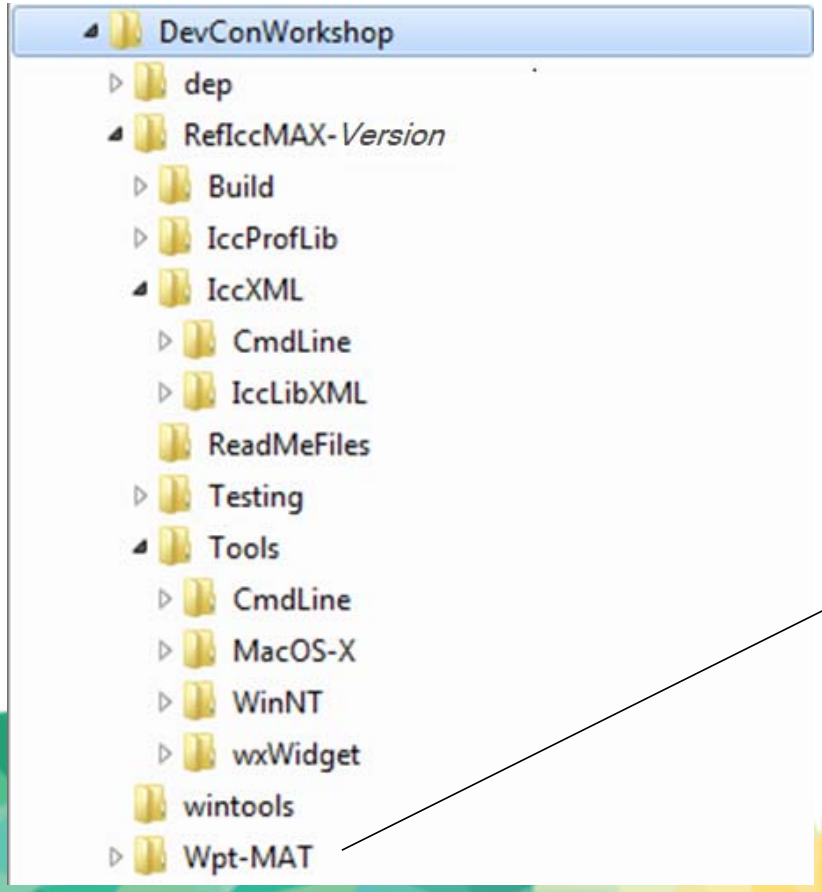
Getting familiar with Workshop Files

Contents of workshop folder



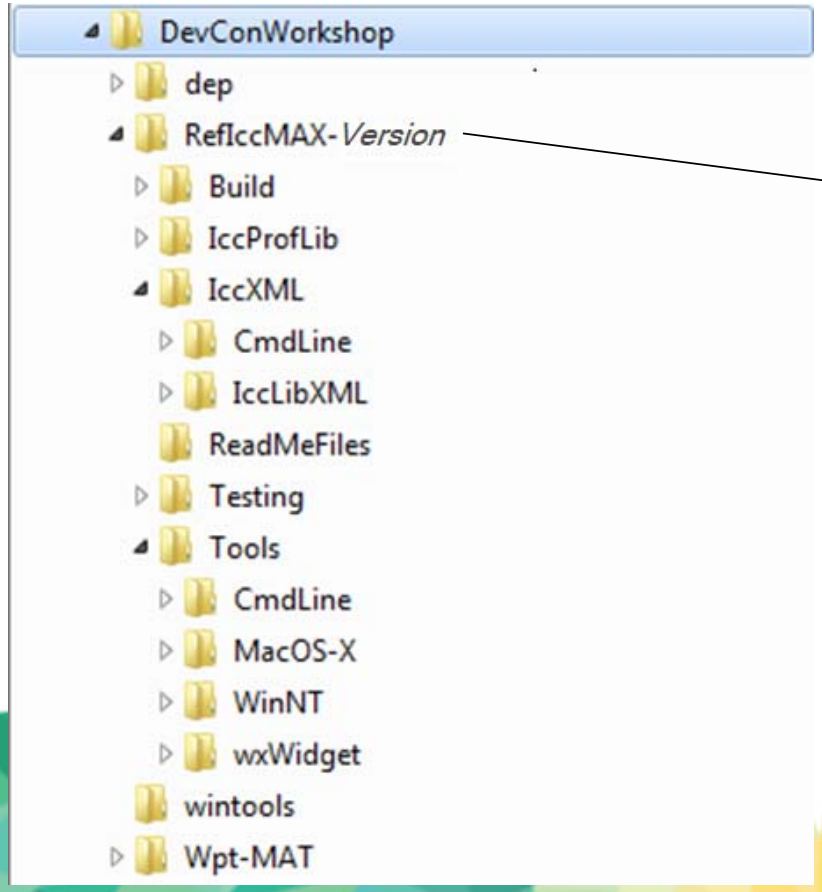
dep
3rd party (dependent) libraries

Contents of workshop folder



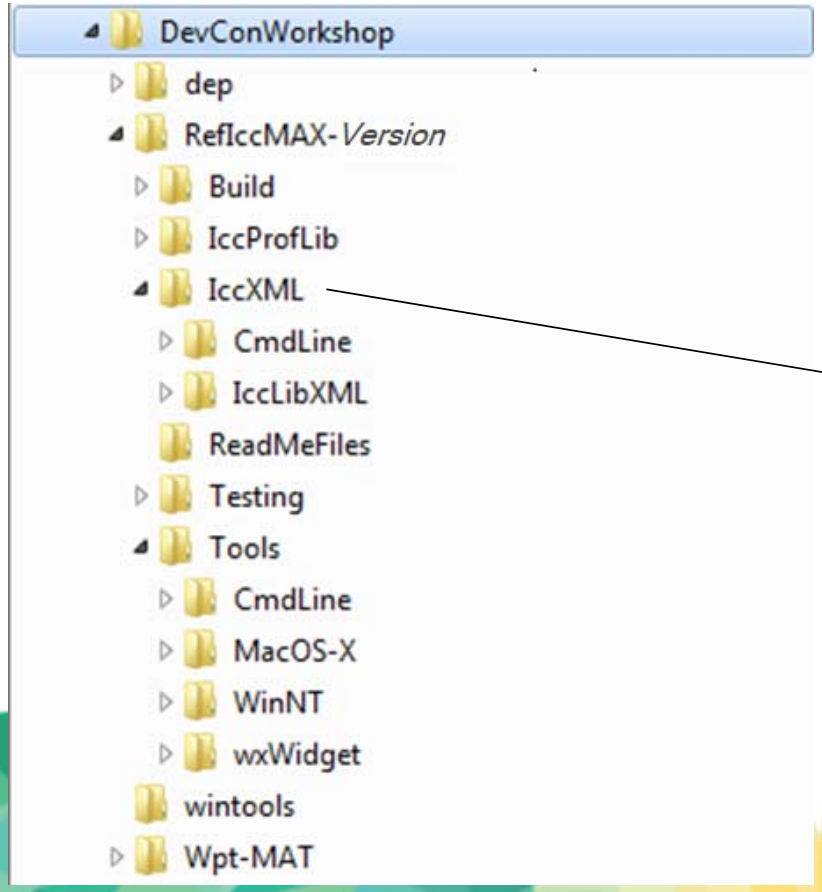
***Waypoint* toolkit for defining Material Adjustment Transforms**

Contents of workshop folder



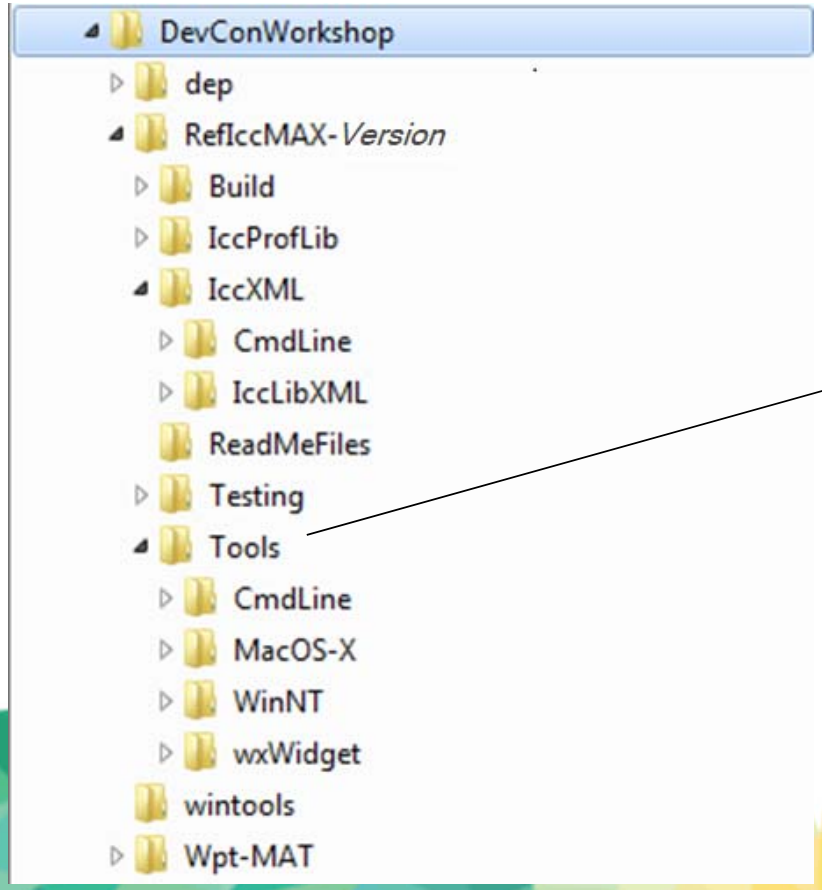
RefIccMAX
Project Folder

Contents of workshop folder



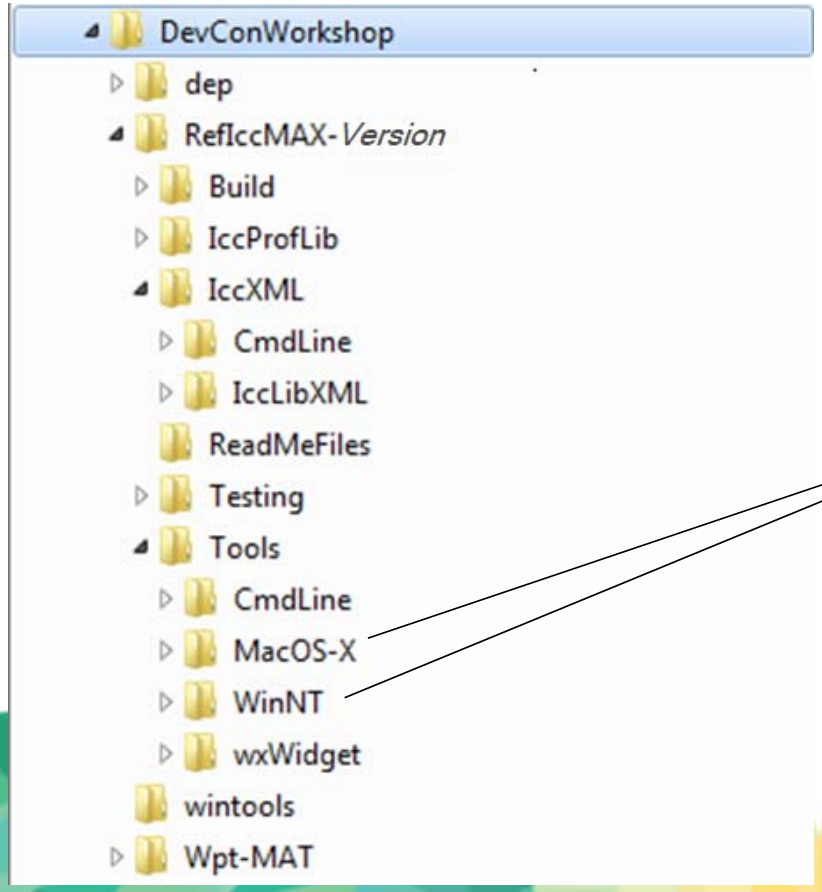
IccLibXML
Extension library source and project for reading and writing iccMAX profiles as XML files

Contents of workshop folder



Source and projects for command line tools that apply and work with iccMAX profiles

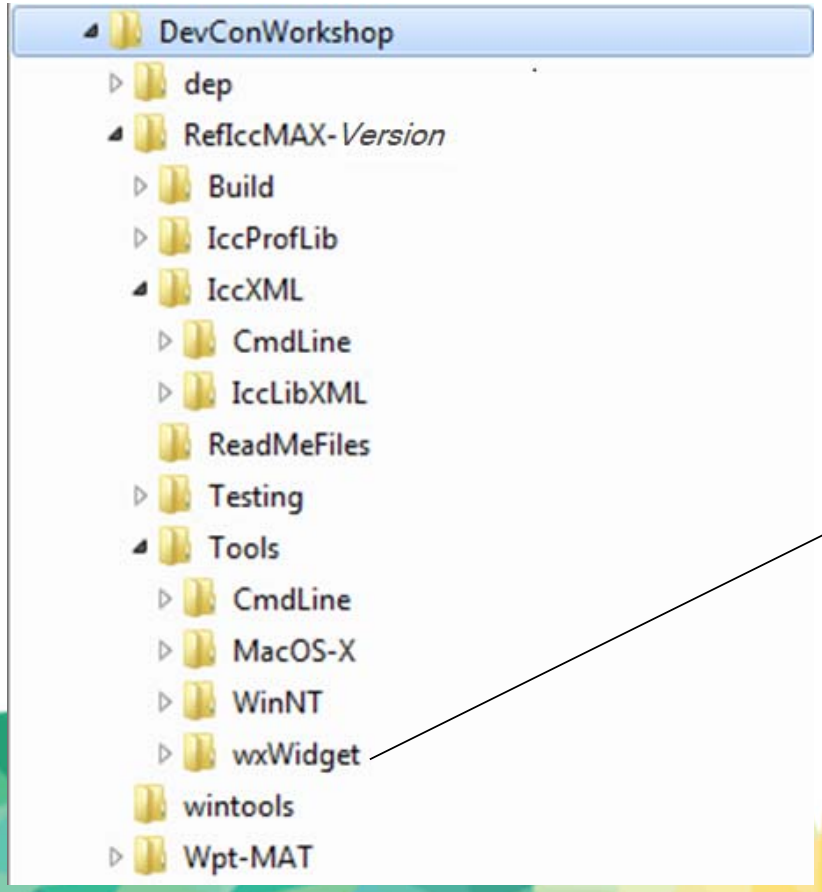
Contents of workshop folder



Source and projects for Operating System level iccMAX based CMMs

RefIccMAX Overview and Install

Contents of workshop folder



wxProfileDump
Source and project for UI based iccMAX investigation tool

Install and Building Libraries / Tools

Demonstration & Practice

Generating / Investigating iccMAX Profiles

Using RefIccMAX Tools

iccFromXML

iccToXML

iccDumpProfile / wxProfileDump

Representing profiles with RefIccMAX

- RefIccMAX supports two ways of representing iccMAX profiles
  - Binary
    - Implemented by IccProfLib library
    - Defined by iccMAX specification
    - Compact, embeddable format
  - XML
    - Implemented by IccLibXML
    - Currently defined by implementation
    - Human readable / editable
- iccFromXML and iccToXML utilities allow for conversions between representations

Investigating profiles using RefIccMAX command line utilities

- Usage: iccToXML icc_profile_file saved_xml_file
  - *icc_profile_file* defines path to iccMAX profile file to parse
  - *saved_xml_file* defines path to xml profile file to create based on parsing *icc_profile_file*
    - *saved_xml_file* can then be viewed/edited using text editor

- Usage: iccProfileDump icc_profile_file tag_id
  - *icc_profile_file* defines path to iccMAX profile file to display
  - *tag_id* defines signature of tag to display contents
    - Use “ALL” to diplay contents of all tags

- Usage: wxProfileDump
  - GUI based profile dump utility

Generating profiles using iccFromXML command line utility

- Usage: iccFromXML xml_file saved_icc_profile_file
  - *xml_file* defines path to XML file to parse
  - *saved_icc_profile_file* defines path to icc profile file to create based on parsing *xml_file*

Generating and investigating iccMAX Profiles

Demonstration & Practice

Working with Spectral Reflectance Images

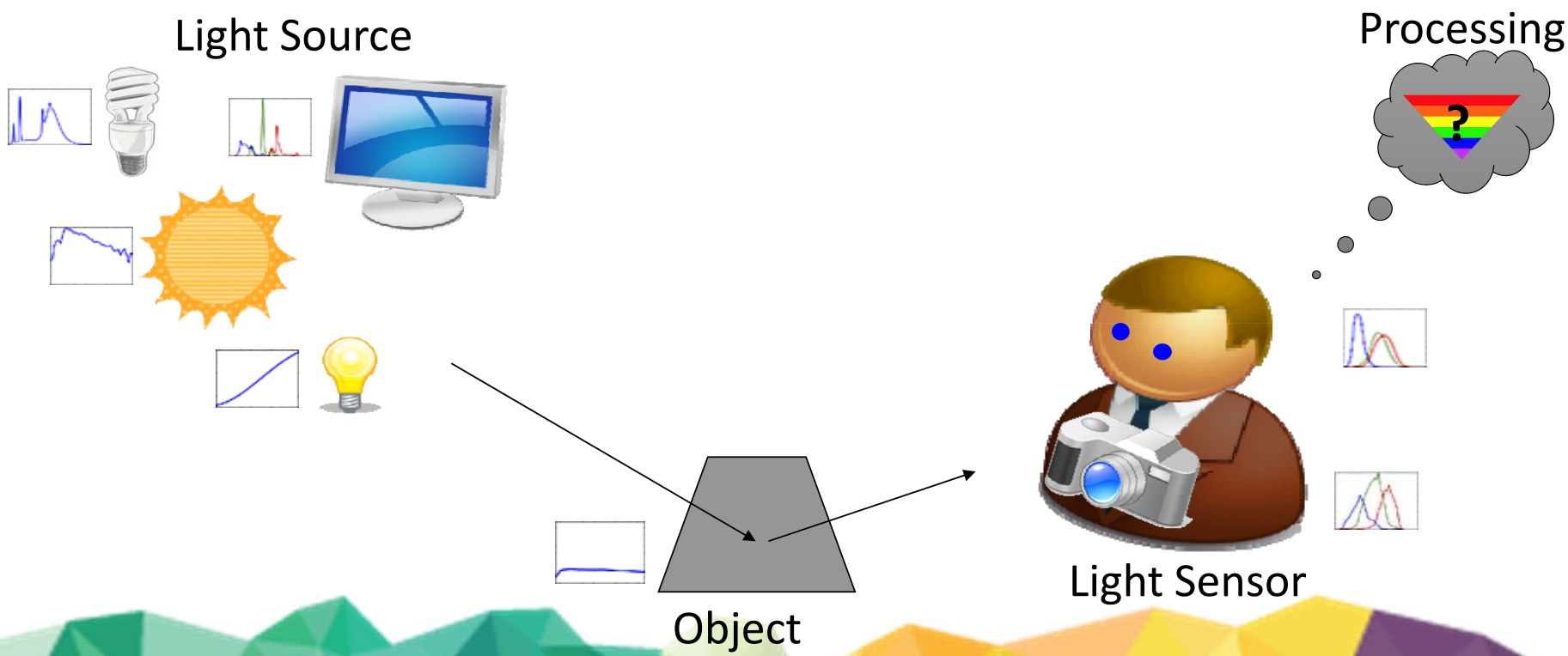
Using RefIccMAX Tools

iccSpecSepToTiff

iccTiffDump

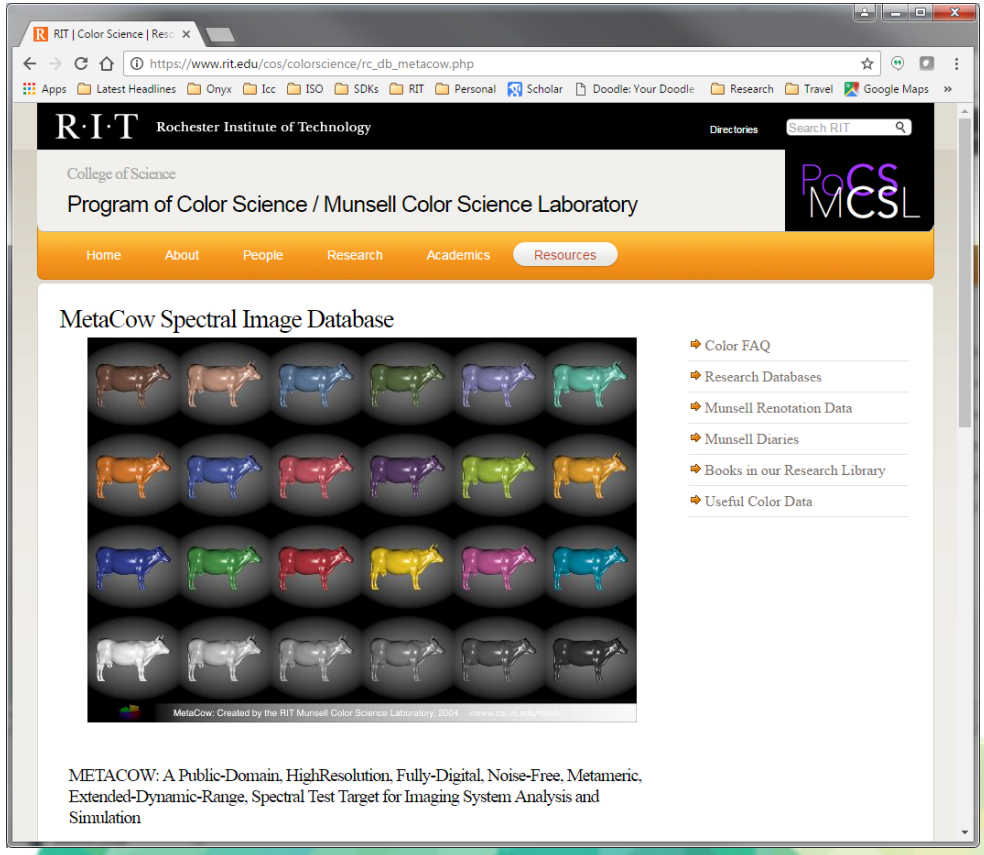
iccApplyProfiles

What is a Spectral Reflectance Image?



- A spectral reflectance image defines how light is reflected at each wavelength for every pixel

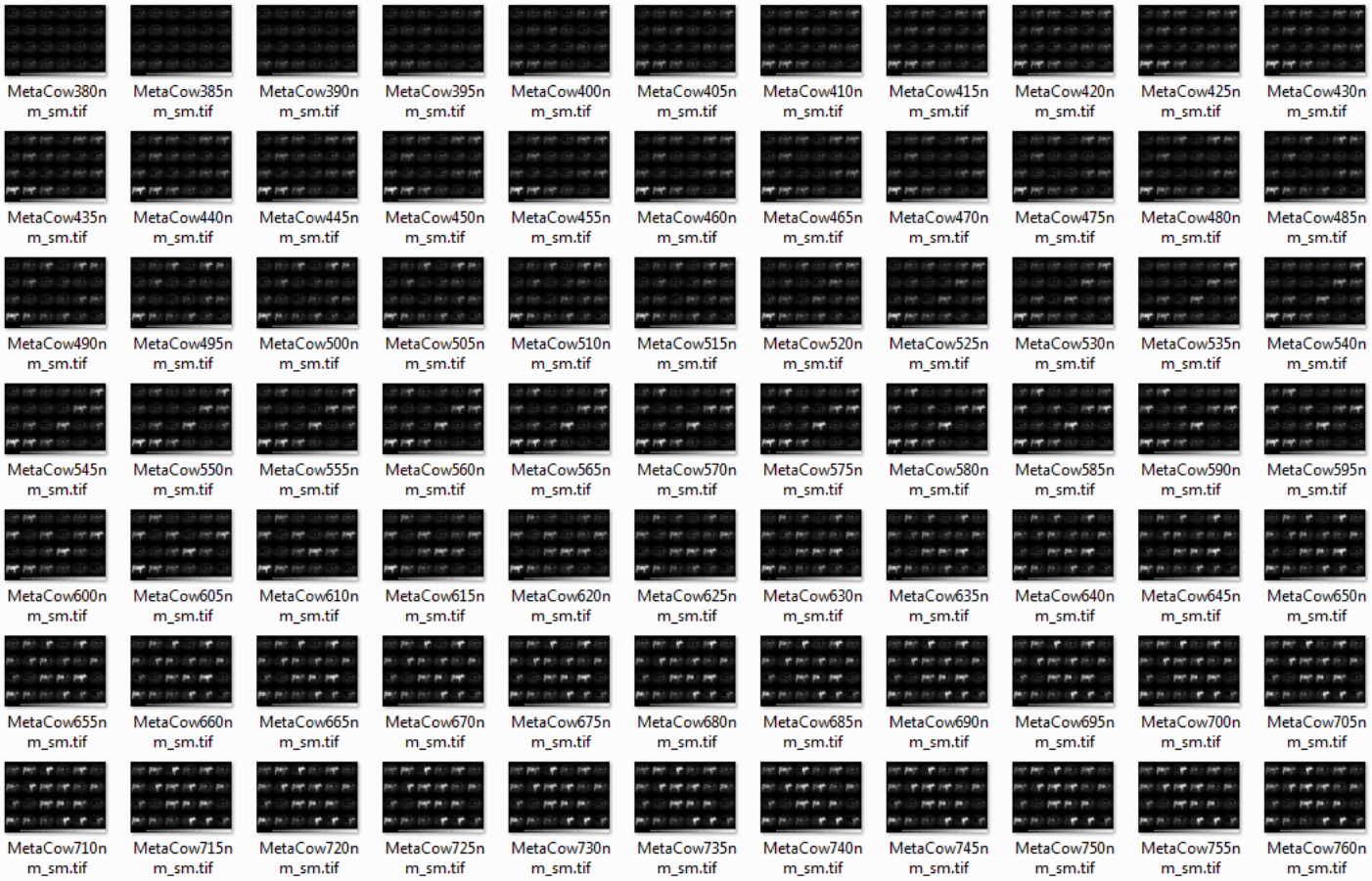
Fun with MetaCow



- The MetaCow spectral image database provides individual images for each wavelength
- Designed so that the heads match the tails under D65 illumination for the standard 2-degree observer
- Small and Large image databases provided

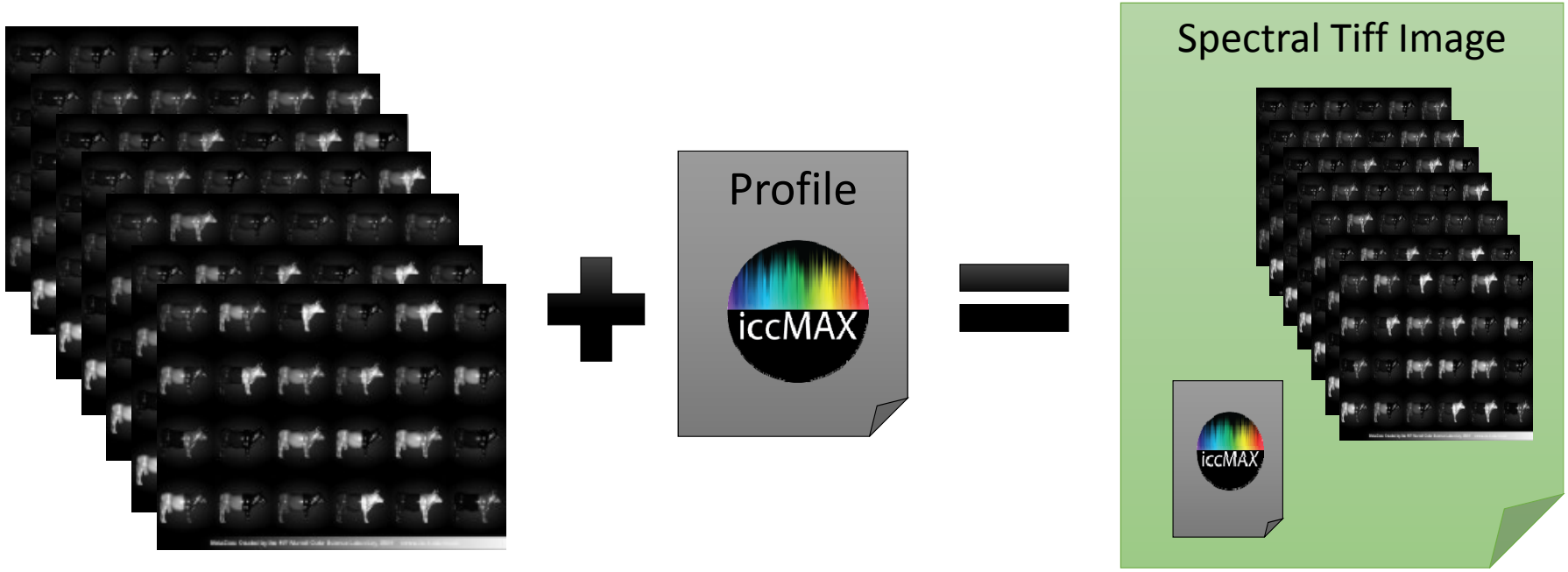
https://www.rit.edu/cos/colorscience/rc_db_metacow.php

The Small MetaCow Database



Images found in:
DevConWorkshop\
RefIccMAX-*Version*\
Testing\
MetaCow\
SmTiffs-400-10-700\

Combining things with iccSepToTiff



- Multiple single wavelength specific images can be combined into a single image embedded with an iccMAX profile
- This results in a spectral reflectance image file that can be color managed using iccMAX

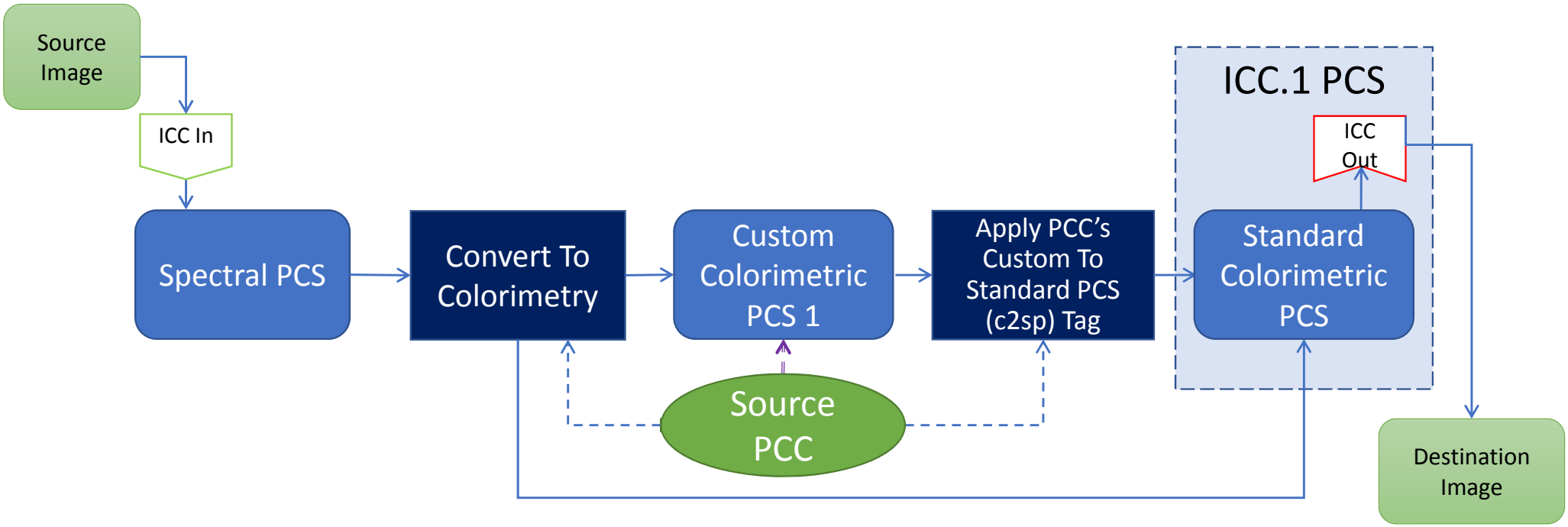
iccSpecSepToTiff

- Usage: iccSpecSep2Tiff output_file compress_flag sep_flag infile_fmt_file start_nm end_nm inc_nm {embedded_icc_profile_file}
  - *output_file* defines path to TIFF file to create
  - *compress_flag* determines whether *output_file* is compressed
    - 0 (no compression) or 1 (compression)
  - *sep_flag* determines whether samples are stored as either
    - 0 : interleaved pixels (ABCDABCDABCD…)
    - 1 : planar interleaved (AAA…BBB…CCC…DDD…)
  - *infile_fmt_file* defines a "sprintf" file designator for selecting files to combine into a single tiff with single integer value specifier
    - Example: ref%03d_nm.tif matches (ref400_nm.tif, ref410_nm.tif etc.)
  - *start_nm* defines starting wavelength value for file matching
  - *end_nm* defines ending wavelength value for file matching
  - *inc_nm* defines wavlength step value for file matching
  - *embedded_icc_profile_file* defines an optional iccMAX profile that can be additionally embedded in *output_file*

iccTiffDump

- Usage: iccTiffDump iccTiffDump tiff_file {exported_icc_file}
  - *tiff_file* defines path to TIFF file to show information about
  - *exported_icc_profile_file* an optional argument that fines an file name to export the embedded iccMAX profile as

Inside Color Management to Display Spectral Reflectance Images

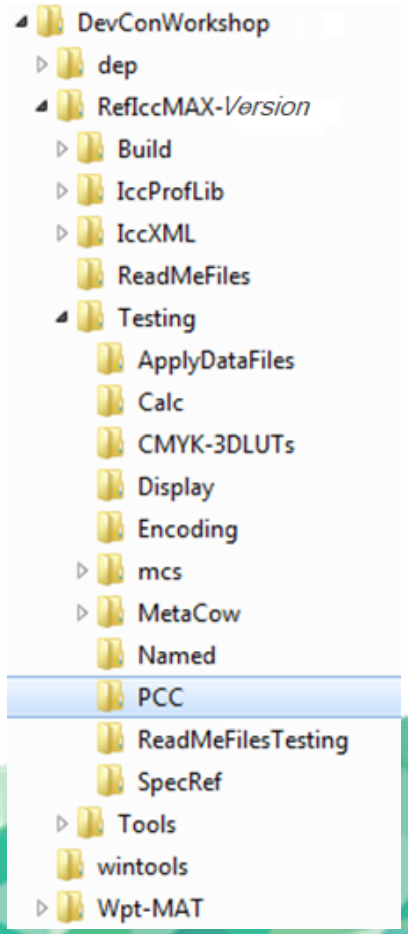


- Profile Connection Conditions (PCC) provide information to CMM for necessary color conversions
- Source PCC can be part of Input ICC or provided separately

Profile Connection Conditions (PCC)

- Profile Connection Conditions (PCC) comprise of:
  - **Header color space and spectral PCS metadata**
  - **spectralViewingConditionsTag** (Signature: 'svcn')
    - A spectralViewingConditionsType tag that spectrally specifies observer and illuminant
    - Illuminant and observer signatures used to determine PCS pass through connection
    - Colorimetric information about surround
  - **customToStandardPcsTag** (Signature: 'c2sp')
    - A multiProcessElement tag to used convert from custom PCS colorimetry to standard D50 / 2-degree observer colorimetry
  - **standardToCustomPcsTag** (Signature: 's2cp')
    - Defines a multiProcessElement tag used to convert from standard D50 / 2-dgree observer colorimetry to custom PCS colorimetry
- PCC information is required whenever spectral PCS is used or colorimetric PCS is NOT standard D50 using 2-degree observer
  - Provides for interoperability to connect different viewing conditions
  - Provides flexibility for when and how conversions are made
  - Provides colorimetric conversion transforms for changes in observer and / or illuminant

Using PCC profiles



- The PCC subfolder contains abstract iccMAX profiles that provide PCC information for various illuminants and observers
  - These spectral reflectance profiles provide identity transforms over spectral range of 400nm to 700nm in 10nm steps
- These profiles can be used to:
  - Define embedded profile when creating spectral reflectance images
  - Provide override PCC when applying profiles

Using iccApplyProfiles to apply iccMAX profiles to images

- Usage: iccApplyProfiles *src_tiff_file dst_tiff_file dst_sample_encoding interpolation dst_compression dst_planar dst_embed_icc* {{**-ENV:***sig value*} *profile_file_path* rendering_intent {**-PCC** *profile_connection_conditions_file*}}
  - *src_tiff_file* is path to source tiff image
  - *dst_tiff_file* is path to create resulting tiff image
  - *dst_sample_encoding* (0=same as src, 1=8-bit, 2=16-bit, 3=float)
  - *interpolation* (CLUT interpolation: 0=linear, 1=tetrahedral)
  - *dst_compression* (0=none, 1=LZW compression)
  - *dst_planar* (0=pixel interleave, 1=separate planes)
  - *dst_embed_icc* (0=no, 1=embed last profile applied
  - *sig* is four character signature of CMM environment values to define
    - Use of CMM environment values is optional multiple environment values can be defined for each profile
  - *value* is numeric value to associate with a CMM environment value
  - *profile_path* is path to ICC profile to apply
    - Use -embedded to use embedded profile in image
    - When multiplOne or more profiles can be applied
  - *rendering_intent*
    - (0=perceptual, 1=relative, 2=saturation, 3=absolute, and others
  - *profile_connection_conditions_file* provides path to iccMAX profile to get override PCC

Applying spectral iccMAX Profiles

- Use iccApplyProfiles with various profile sequences

Applying iccMAX Profiles

Working with Spectral Reflectance Images

Demonstration & Practice

Building applications using ICCProfLib

Getting Inside iccApplyProfiles and iccApplyNamedCmm

Building Applications using IccProfLib

- Applications can utilize IccProfLib to parse, manipulate, and apply both ICC and iccMAX profiles
- This involves:
  - Adding IccProfLib to include folders for project
  - Linking to an appropriate IccProfLib library
    - Release/Debug
    - 32-bit / 64-bit
    - Static library / Dynamic library
  - Using functions and class objects from IccProfLib in your code
- RefIccMAX tools provide convenient examples for coding

Basic Profile Application

1. Include IccCmm.h
2. Instantiate a CIccCMM/CIccNamedColorCmm object
3. Add Profiles to CMM object
   - Specifying rendering intent, PCC, and other transform configurations
4. Instruct CMM to begin transformations
   - Via CMM's Begin() member function call
5. *Optionally get CIccApply object(s) from CMM*
   - *Profile application is NOT thread safe so separate Apply objects should be used for each thread*
6. Apply colors/pixels using either CMM or Apply object
   - Color/pixel encoding modifications may be needed
7. Cleanup
   - Delete Apply and CMM objects as needed

Snippets of code from iccApplyProfiles.cpp

```
#include <CIccCmm.h>

icStatusCMM stat;      //status variable for CMM operations

//Allocate a CIccCmm to use to apply profiles.
CIccCmm theCmm(icSigUnknownData, icSigUnknownData, true);

//Read profile from path and add it to theCmm
stat = theCmm.AddXform(argv[nCount], nIntent<0 ? icUnknownIntent :
                       (icRenderingIntent)nIntent,
                       nInterp, pPccProfile,
                       (icXformLutType)nType, bUseMPE, &Hint);

//All profiles have been added to CMM
if((stat=theCmm.Begin())) {

//Use CMM to convert SrcPixel to DestPixel
theCmm.Apply(DestPixel, SrcPixel);
```

Non-Image based CMM application

- The iccApplyNamedCmm tool takes a text data file containing colors to be applied by a sequence of profiles
- The data file has header that specifies color space signature and encoding type
- iccApplyNamedCmm allows for application and testing of Named Color profiles
- Debugging iccApplyNamedCmm allows for surgical access to understanding workings of IccProfLib
  - Useful when debugging transforms in profiles as you create them using XML

iccApplyNamedCmm.cpp uses CIccNamedColorCmm

```
if(SrcspaceSig==icSigNamedData) {//Are names coming is as an input?
    switch(namedCmm.GetInterface()) {
        case icApplyNamed2Pixel:
            namedCmm.Apply(DestPixel, SrcNameBuf, tint); break;
        case icApplyNamed2Named:
            namedCmm.Apply(DestNameBuf, SrcNameBuf, tint); break;
        default: //error
    }
}
else { //pixel sample data coming in as input
    switch(namedCmm.GetInterface()) {
        case icApplyPixel2Pixel:
            namedCmm.Apply(DestPixel, SrcPixel); break;
        case icApplyPixel2Named:
            namedCmm.Apply(DestNameBuf, SrcPixel); break;
        default: //error
    }
}
```

Using iccApplyNamedCmm to apply iccMAX profiles to images

- Usage: iccApplyNamedCmm {**-debugcalc**} *data_file_path* final_data_encoding{:FmtPrecision{:FmtDigits}} interpolation {{**-ENV:***sig value*} *profile_file_path rendering_intent* {**-PCC** *connection_conditions_path*}}
  - Inclusion of –**debugcalc** forces verbose calculator element debugging
  - *data_file_path* is path to data file to get color data to apply
  - *final_data_encoding* (0=value, 1=percent, 2=UnitFloat(clipping), 3=Float, 4=8-bit, 5=16-bit, 6=16-bitV2)
    - *FmtPrecision* & *FmtDigits* define precision of output floating point values
  - *interpolation* (CLUT interpolation: 0=linear, 1=tetrahedral)
  - *sig* is four character signature of CMM environment values to define
    - Use of CMM environment values is optional multiple environment values can be defined for each profile
  - *value* is numeric value to associate with a CMM environment value
  - *profile_path* is path to ICC profile to apply
    - Use -embedded to use embedded profile in image
    - When multiplOne or more profiles can be applied
  - *rendering_intent*
    - (0=perceptual, 1=relative, 2=saturation, 3=absolute, and others
  - *profile_connection_conditions_file* provides path to iccMAX profile to get override PCC

Building applications using ICCProfLib

Demonstration & Practice

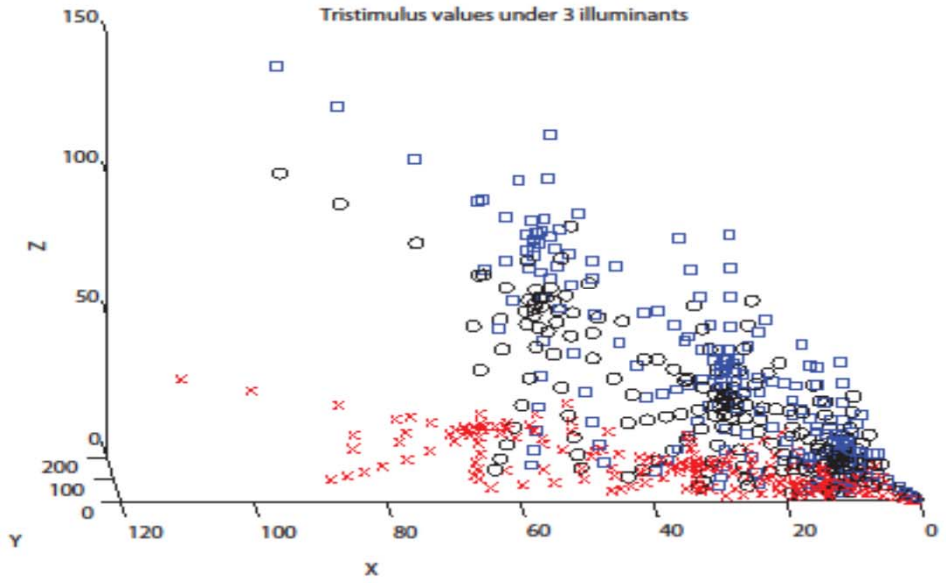
Setting Up Profile Connection Condition Tag Values

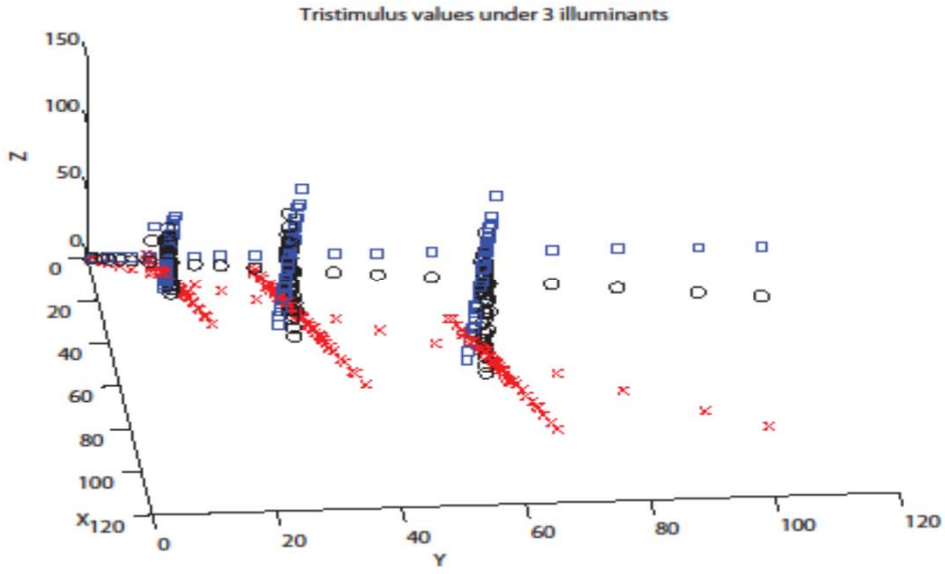
Using Wpt-MAT Tools

Profile Connection Conditions Tags

- The Tags associated with Profile Connection Conditions (PCC)
  - Define the observer's color matching functions
    - Must provide XYZ like values that can be converted to/from Lab using CIELAB conversion math
  - Define the illuminant's spectral power distribution
  - Provide conversions between custom colorimetry and legacy Standard PCS colorimetry
    - The (legacy) Standard PCS uses a CIE D50 illuminant with the CIE Standard 1931 2-degree observer
- How do you populate these tags if you want to use a custom or alternate observer and/or illuminant?

Why are colorimetric conversions needed?





- Colorimetric values change for different illuminants/observers for the same spectral reflectances
  - Figure of Tristimulus values for same Munsell glossy reflectances for 2-degree observer under Illuminant A (red), D65 (black), D100 (blue)
  - Similar changes occur for changes in observer
- To make conversions - some form of “colour equivalency” for changes in observer and/or illuminant is needed

Converting PCC Colorimetry

Methods of converting PCC colorimetry

1. No Conversion
   - Just plug in Identity matrices
2. White Balancing (vonKries Normalization)
   - Used by CIELAB equations
3. Chromatic Adaptation
   - Based on Corresponding Color
   - Used by Color Appearance Models (CIECAM02)
4. Material Color Adjustment
   - Based on Material Color Equivalency

White Balancing (vonKries Normalization)

$$\mathbf{c}_t = \mathbf{M}\mathbf{c}_s$$

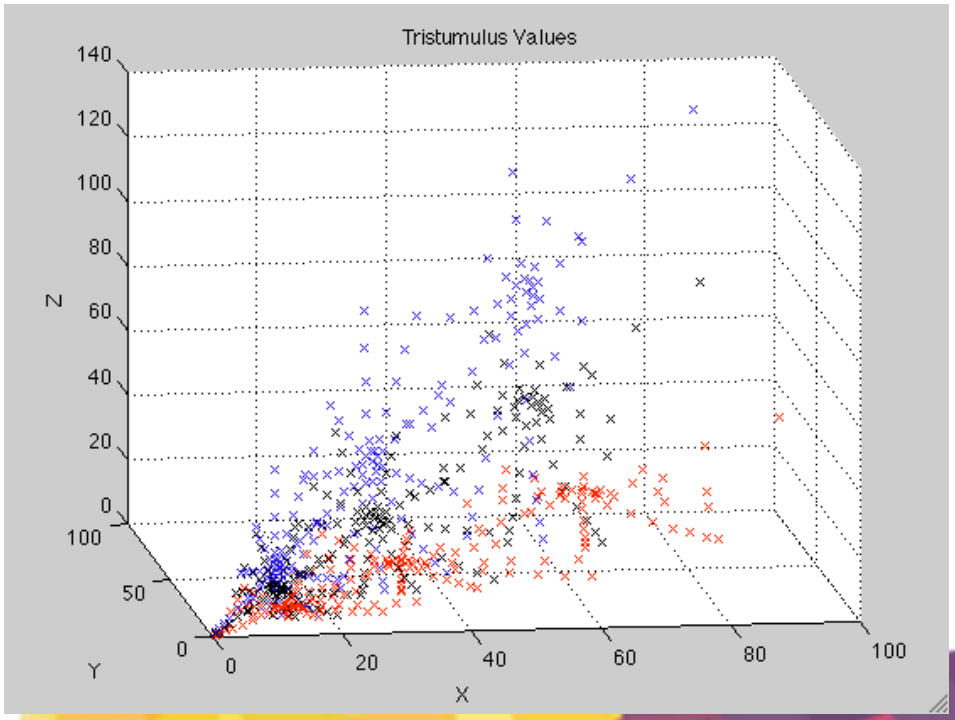
$$\mathbf{M} = \begin{bmatrix} w_{1,t}/w_{1,s} & 0 & 0 \\ 0 & w_{2,t}/w_{2,s} & 0 \\ 0 & 0 & w_{3,t}/w_{3,s} \end{bmatrix}$$

$$\mathbf{w}_s = \mathbf{Cl}_s, \mathbf{w}_t = \mathbf{Cl}_t$$

$$\mathbf{M} = (\mathbf{A}_t)^{-1}\mathbf{A}_s$$

$$\mathbf{A}_x = \begin{bmatrix} 1/w_{1,x} & 0 & 0 \\ 0 & 1/w_{2,x} & 0 \\ 0 & 0 & 1/w_{3,x} \end{bmatrix}$$

$$\mathbf{w}_x = \mathbf{Cl}_x$$



- Determined by multiplying by ratio of destination colorimetry divided by source colorimetry for illuminants in both connection conditions
- Only provides grayscale color equivalency
- Doesn't predict color appearance through adaptation or material color
- Doesn't predict differences seen by different observers

Converting PCC Colorimetry

Chromatic Adaptation Transforms

$$\mathbf{c}_t = \mathbf{T}\mathbf{c}_s$$

$$\mathbf{T} = \mathbf{S}^{-1}\mathbf{M}\mathbf{S}$$

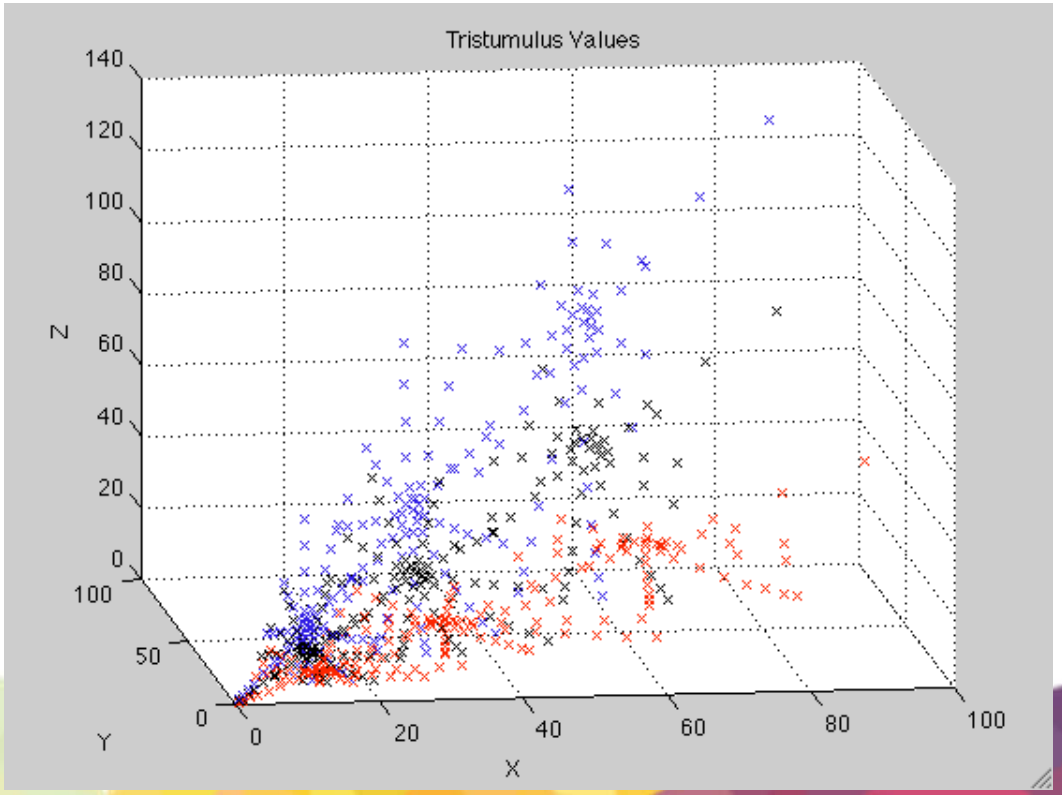
$$\mathbf{M} = \begin{bmatrix} w_{1,t}/w_{1,s} & 0 & 0 \\ 0 & w_{2,t}/w_{2,s} & 0 \\ 0 & 0 & w_{3,t}/w_{3,s} \end{bmatrix}$$

$$\mathbf{w}_s = \mathbf{S}\mathbf{C}\mathbf{I}_s,\ \mathbf{w}_t = \mathbf{S}\mathbf{C}\mathbf{I}_t$$

$$\mathbf{T} = (\mathbf{N}_t)^{-1}\mathbf{N}_s$$

$$\mathbf{N}_x = \begin{bmatrix} 1/w_{1,x} & 0 & 0 \\ 0 & 1/w_{2,x} & 0 \\ 0 & 0 & 1/w_{3,x} \end{bmatrix}\mathbf{S}$$

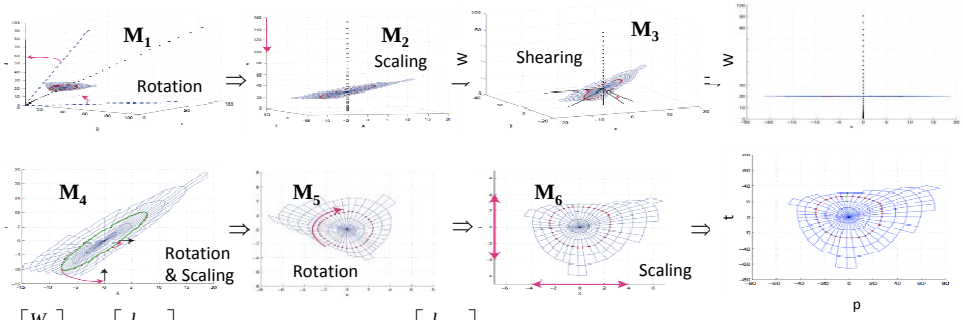
$$\mathbf{w}_x = \mathbf{S}\mathbf{C}\mathbf{I}_x$$



- Determined by:
  - First applying sharpening matrix optimized to predict corresponding color experiments
  - Then applying vonKries normalization
  - Then applying inverse sharpening matrix
- Useful for predicting appearance change by single observer for changes in illuminant
- Doesn't predict material color
- Doesn't predict differences seen by different observers

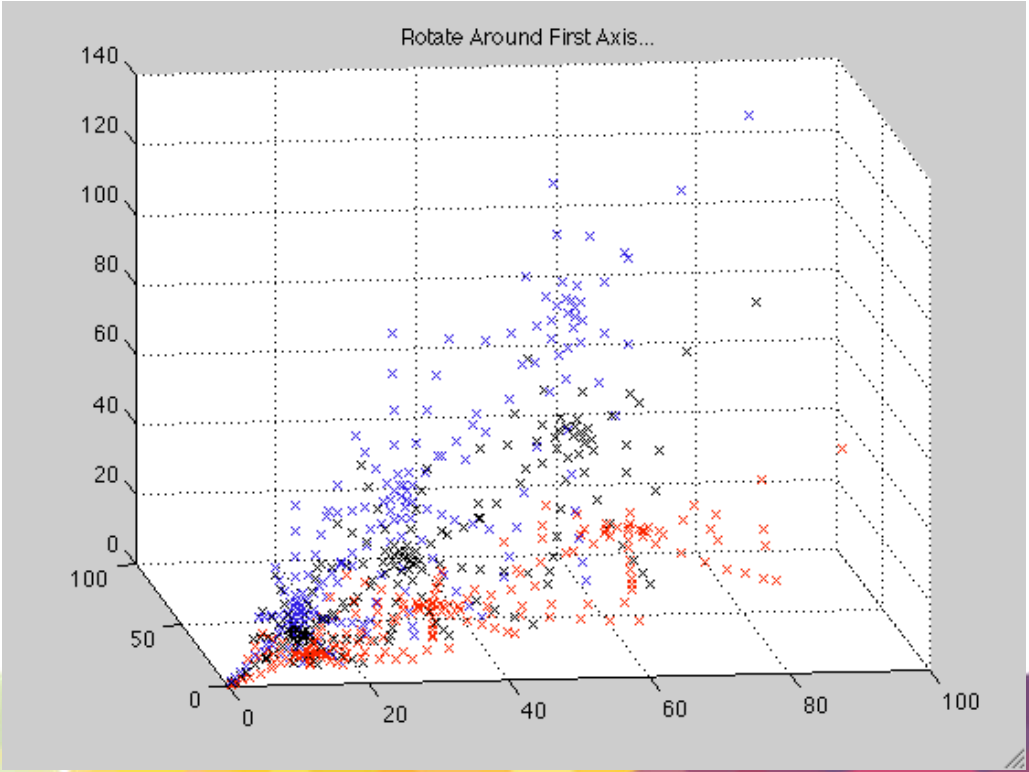
Converting PCC Colorimetry

Material Color Equivalency using Wpt (WayPoint) Normalization



$$\begin{bmatrix} W \\ p \\ t \end{bmatrix} = \mathbf{A} \begin{bmatrix} l_{src} \\ m_{src} \\ s_{src} \end{bmatrix} = \mathbf{M}_6\mathbf{M}_5\mathbf{M}_4\mathbf{M}_3\mathbf{M}_2\mathbf{M}_1 \begin{bmatrix} l_{src} \\ m_{src} \\ s_{src} \end{bmatrix}$$

- Determined by applying sequence of linear transforms to material sensor coordinates in order to separately minimize differences in white point, lightness, chroma and hue
  - Regression can be used to minimize to reference observing conditions
- Doesn't predict color appearance
- Predicts material color equivalency
- Can be used to define Material Adjustment Transform (MAT) that predicts differences seen by different observers



Wpt coordinates are invariant of Linear transformations of observer color matching functions

Material Color Equivalency using Wpt (WayPoint) Normalization

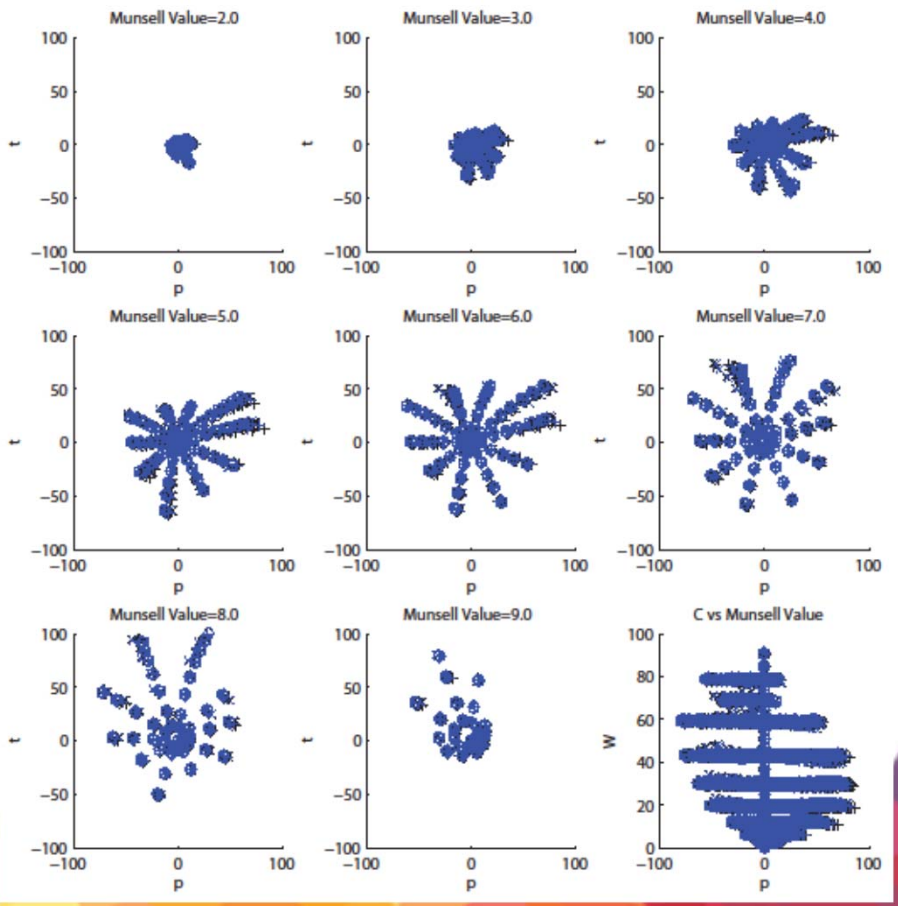
Material Adjustment Transforms

$$\mathbf{c}_t = \mathbf{M}\mathbf{c}_s$$

$$\mathbf{M} = \left(\mathbf{A}_t\right)^{-1}\mathbf{A}_s$$

$$\mathbf{A}_x = T\left(\mathbf{C}_x, \mathbf{l}_x\right)$$

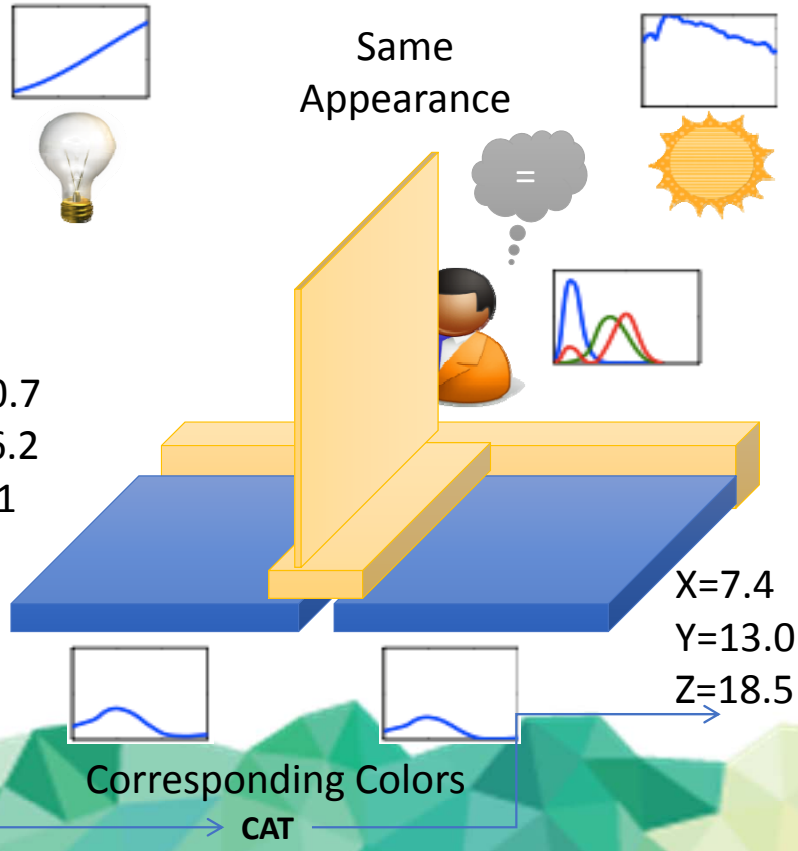
- A Material Adjustment Transform is determined by multiplying the inverse of the Wpt normalization matrix for the destination observing conditions (combination of observer and illuminant) by the Wpt normalization matrix for the source observing conditions
- Material color equivalency results in minimized differences for changes in both observer and illuminant



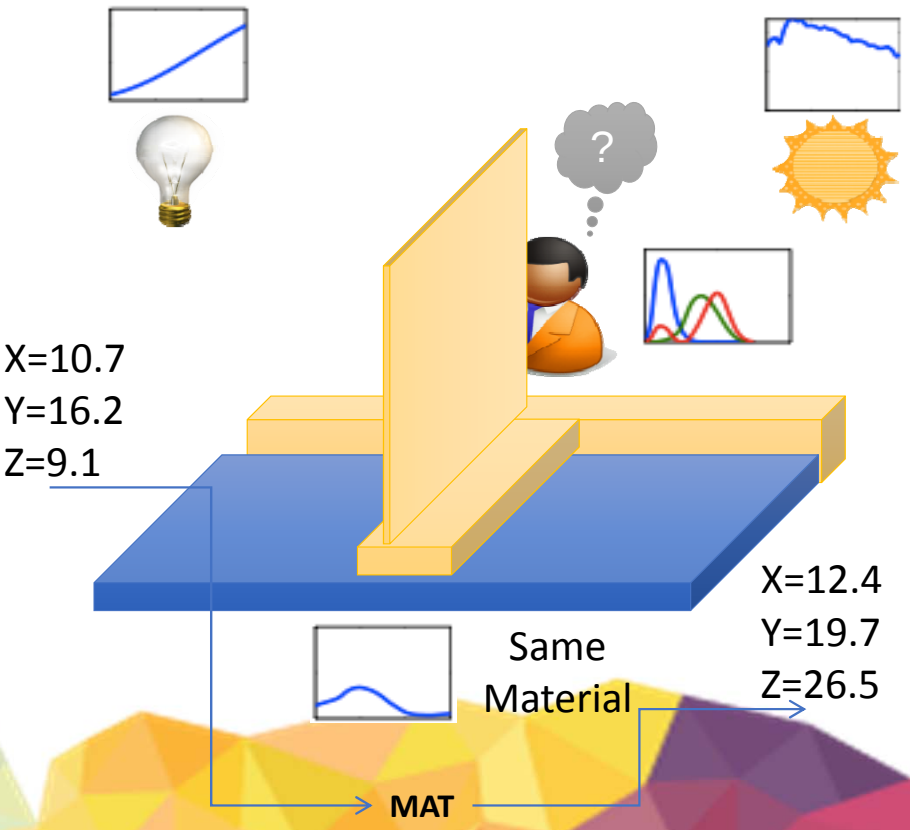
Converting PCC Colorimetry

Differences Between CATs and MATs

Chromatic Adaptation

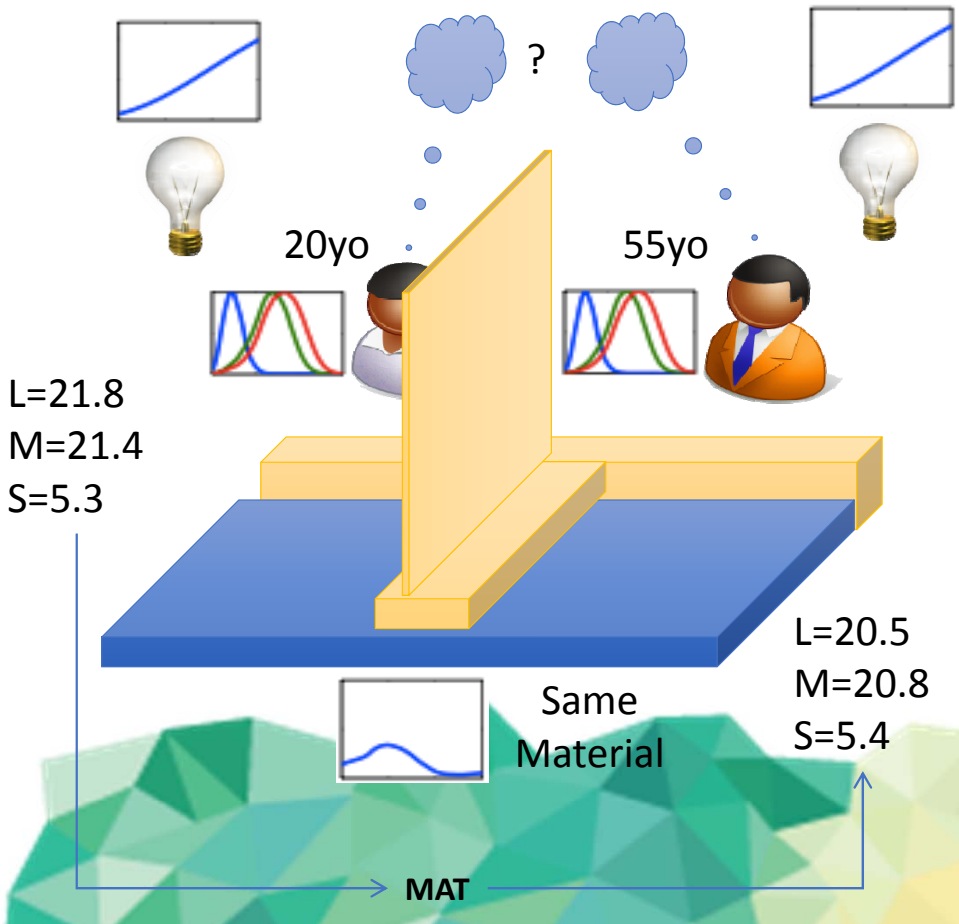


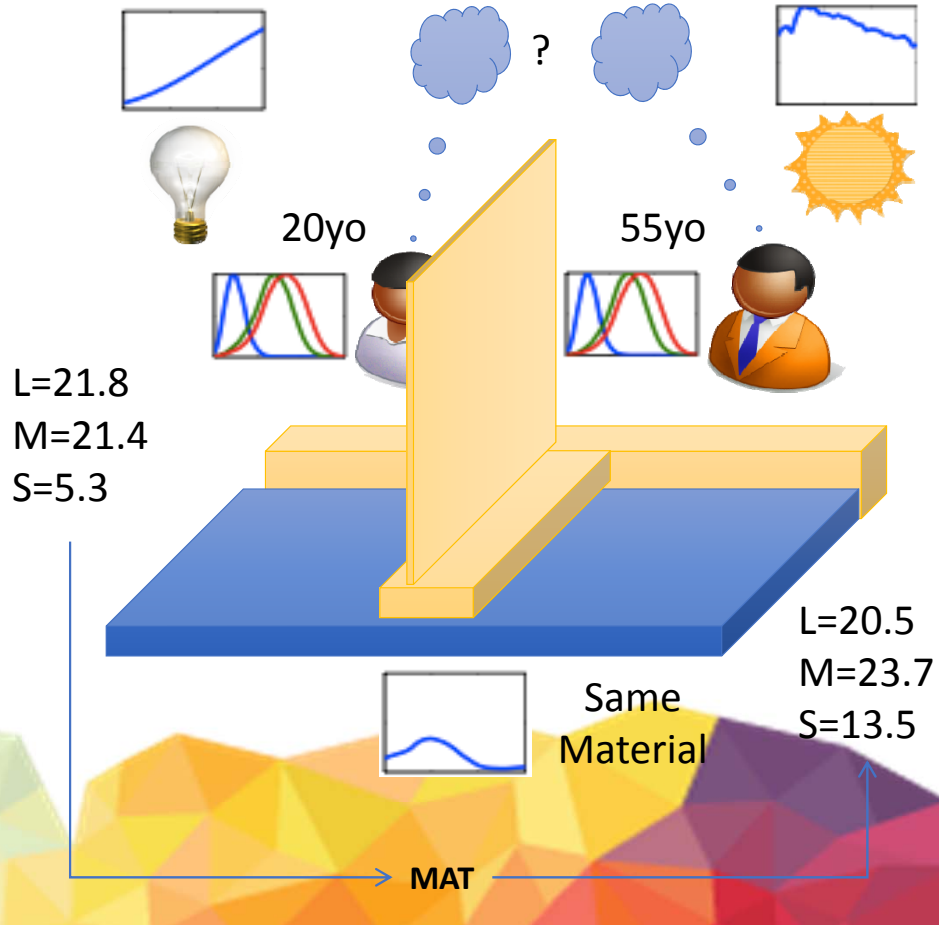
Material Adjustment



CATs and MATs are both examples of Sensor Adjustment Transforms (SATs)

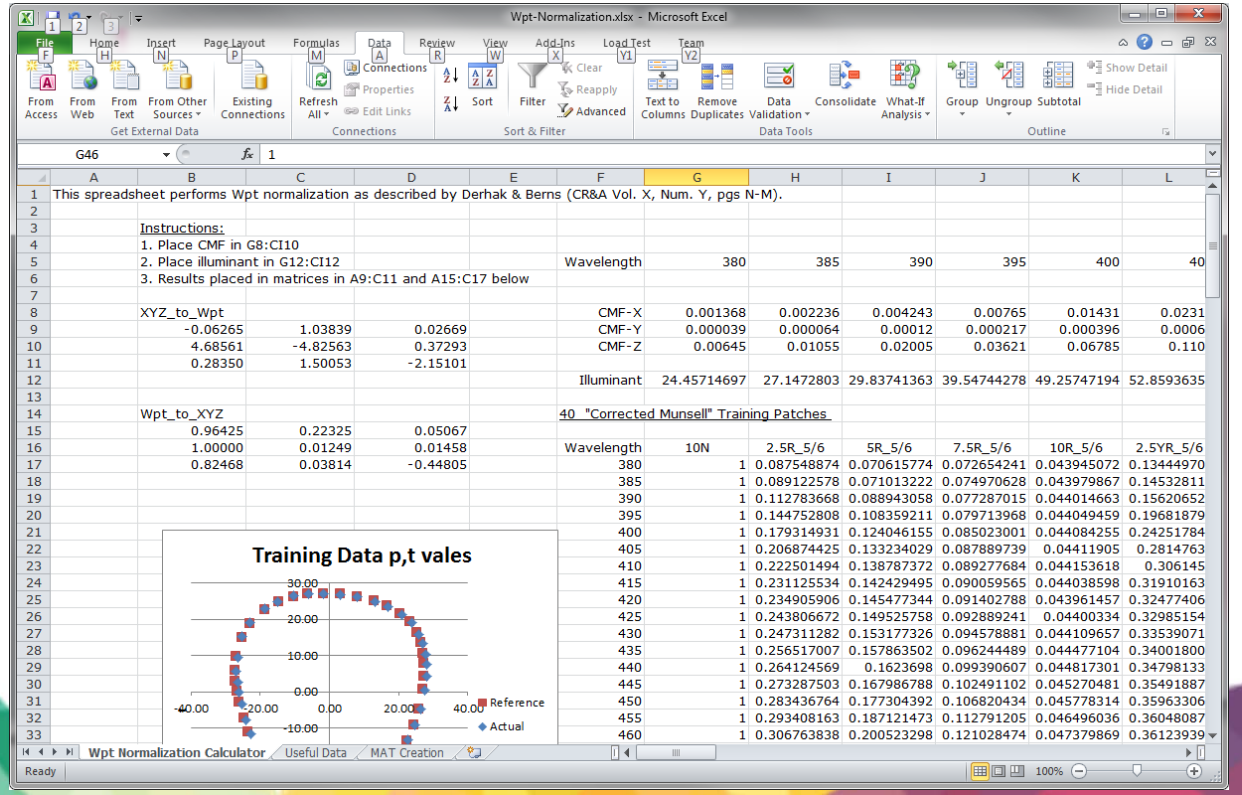
MATs predict differences in both observer and illuminant





Using Excel spreadsheet to Populating PCC tags

- Wpt-Tools folder Wpt has Normalization Excel spreadsheet
- The “Wpt Normalization Calculator” tab can be used to determine Wpt normalization matrix for a given observer and illuminant
- The “Useful Data” tab has both 2-degree and 10-observer data as well as data for various illuminants
- The “MAT Creation” tab can be used to create matrices the CustomToStandardPcs and StandardToCustomPcs tags



Populating PCC Tags

Using MATs to determine Color Matching Functions

- Wpt Normalization matrices can be used to convert Sensor Sensitivity Functions or Cone Fundamentals ($\mathbf{F}$) to Color Matching Functions ($\mathbf{M}$) for PCC observer population as follows:
  1. Determine Wpt Normalization matrix ($\mathbf{A}_s$) for Sensor Sensitivity Functions or Cone Fundamentals and Illuminant used in PCC
  2. Determine inverse Wpt Normalization matrix ($(\mathbf{A}_d)^{-1}$) for standard 2-degree observer and Illuminant used in PCC
  3. Color Matching Functions ($\mathbf{M}$) are found by transforming Cone Fundamentals ($\mathbf{F}$) as follows:
     - $\mathbf{M} = (\mathbf{A}_d)^{-1}\mathbf{A}_s\mathbf{F}$
- This results in Color Matching Functions that:
  - Result in same XYZ values for the PCC illuminant as the standard 2-degree observer
  - Predict the same color matches as the Sensor Sensitivity Functions or Cone Fundamentals do
  - Have similar characteristics in XYZ values as standard 2-degree observer

Populating PCC Tags using Material Adjustment Transforms – Part 1

- Populating spectralViewingConditionsTag

1. Place Spectral Power Distribution of illuminant in *illuminant* field
2. Use MAT to determine Color Matching Functions for custom observer/sensors and place in *observer* field
3. Determine Illuminant XYZ for observer and place in *illuminantXYZ* field
4. Make other metadata fields match observer and illuminant fields

Populating PCC Tags using Material Adjustment Transforms – Part 2

- Populating customToStandardPcsTag

1. Setup multiProcessElementsType with single MatrixElement
2. Determine matrix values for MatrixElement
   - If a MAT was used to determine ColorMatchingFunctions then MatrixElement values are defined by MAT going from PCC illuminant with 2-degree observer to D50 with 2-degree observer
   - Else if MAT was not used (i.e. custom ColorMatchingFunctions are used) then MatrixElement values are defined by MAT going from PCC illuminant with PCC observer to D50 with 2-dgree observer

Populating PCC Tags using Material Adjustment Transforms – Part 3

- Populating standardToCustomPcsTag

1. Setup multiProcessElementsType with single MatrixElement
2. Determine matrix values for MatrixElement
   - If a MAT was used to determine ColorMatchingFunctions then MatrixElement values are defined by MAT going from D50 with 2-degree observer to PCC illuminant with 2-degree observer
   - Else if MAT was not used (i.e. custom ColorMatchingFunctions are used) then MatrixElement values are defined by MAT going from D50 with 2-dgree observer to PCC illuminant with PCC observer
   - Matrix in standardToCustomPcsTag should be an inverse of the matrix in customToStandardPcsTag

Useful MATLAB functions in Wpt-TOOLS\MATLAB

- getCIEStruct to get structure with CIE illuminants and color matching functions
  - cie=getCieStruct(380:5:780);
- getWptStruct to get structure with Wpt Normalization Matrices for a given observer and illuminant
  - wpts250=getWptStruct(cie.lambda, cie.cmf2deg, cie.illD50);
- interpCmf to resample Color Matching Functions, Cone Fundamentals or Sensor Sensitivity Functions
  - cf2=interpCmf(lambda, cf, cie.lambda, 'linear')
- cie2006cfs to get Cone Fundamentals for CIE 2006 observer
  - cf=get2006cfs(4, 50, cie.lambda);
- cf2cmf to convert Cone Fundamentals to Color Matching Functions
  - [cmf ctop stoc wt]=cf2cmf(cie.lambda, cf, cie.illD65);
- dumpMatrix dumps contents of matrix to file or to console
  - dumpMatrix(1, cmf);

Setting Up Profile Connection Condition Tag Values

Demonstration & Practice

Estimating and Manipulating spectral Reflectance with iccMAX Profiles

Using Profiles in Testing\SpecRef and

More Fun with MetaCow

• Color Management and Spectral Reflectance

- Two signification questions arise with ability to perform color management with a Spectral Reflectance PCS
  - How do you estimate spectral reflectance from colorimetry?
  - How do you manipulate spectral reflectance to achieve desired changes in appearance for a given observer and illuminant?
    - Color Rendering
    - Gamut Mapping
- Answering first question provides method of answering second question

n wi th Sp ec tr al R ef le ct an

Spectral Estimation from Wpt coordinates

- Derhak's PhD dissertation outlines a method of estimating spectral reflectance directly from Wpt coordinates

1. Convert W,p,t to polar based W,Cpt,hpt
2. Determine characteristic reflectance $\mathbf{O}_{\text{select}}$ based on hpt
   - Various methods proposed in dissertation
3. Calculate $\mathbf{M}_{\text{select}}$ from the W and Cpt associated with $\mathbf{O}_{\text{select}}$
4. Use $\mathbf{M}_{\text{select}}$ to find g and s scalars
   - Can be pre-computed for each hue
5. Use Chau's spectral decomposition equation to get spectral estimation

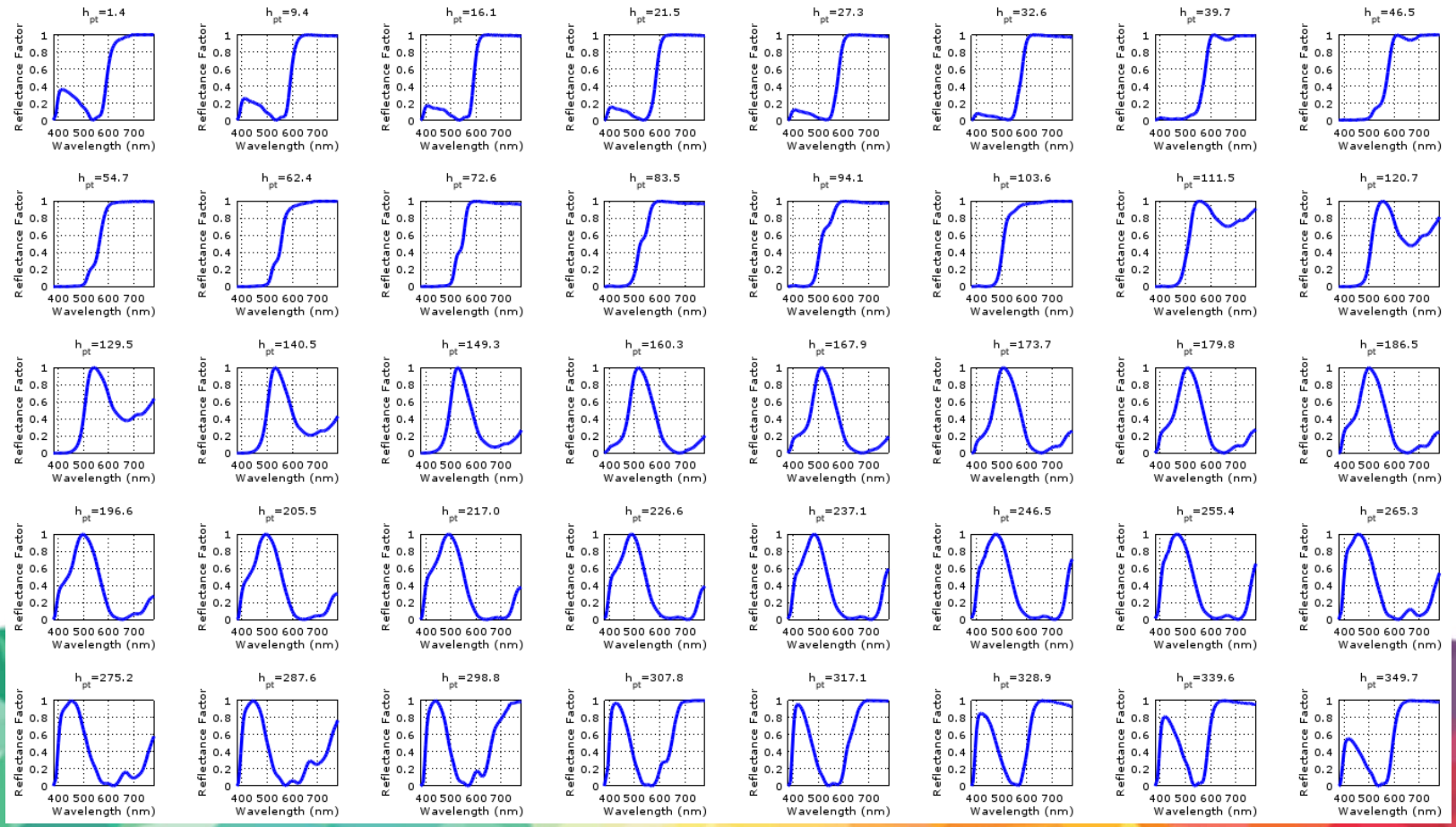
$$\mathbf{M}_{\text{select}} = \begin{bmatrix} W_{\mathbf{I}} & W_{\text{select}} \\ 0 & C_{\text{select}} \end{bmatrix}^{-1}$$

$$\begin{bmatrix} g \\ s \end{bmatrix} = \mathbf{M}_{\text{select}} \begin{bmatrix} W \\ C_{pt} \end{bmatrix}$$

$$\mathbf{O}_{\text{any}} = g\mathbf{I} + s\mathbf{O}_{\text{select}}$$

Direct Spectral Estimation

Munsell Characteristic Hue Reflectances ($\mathbf{O}_{\text{select}}$)



Estimating Spectral Reflectance

Creating RGB profiles with Spectral Reflectance PCS

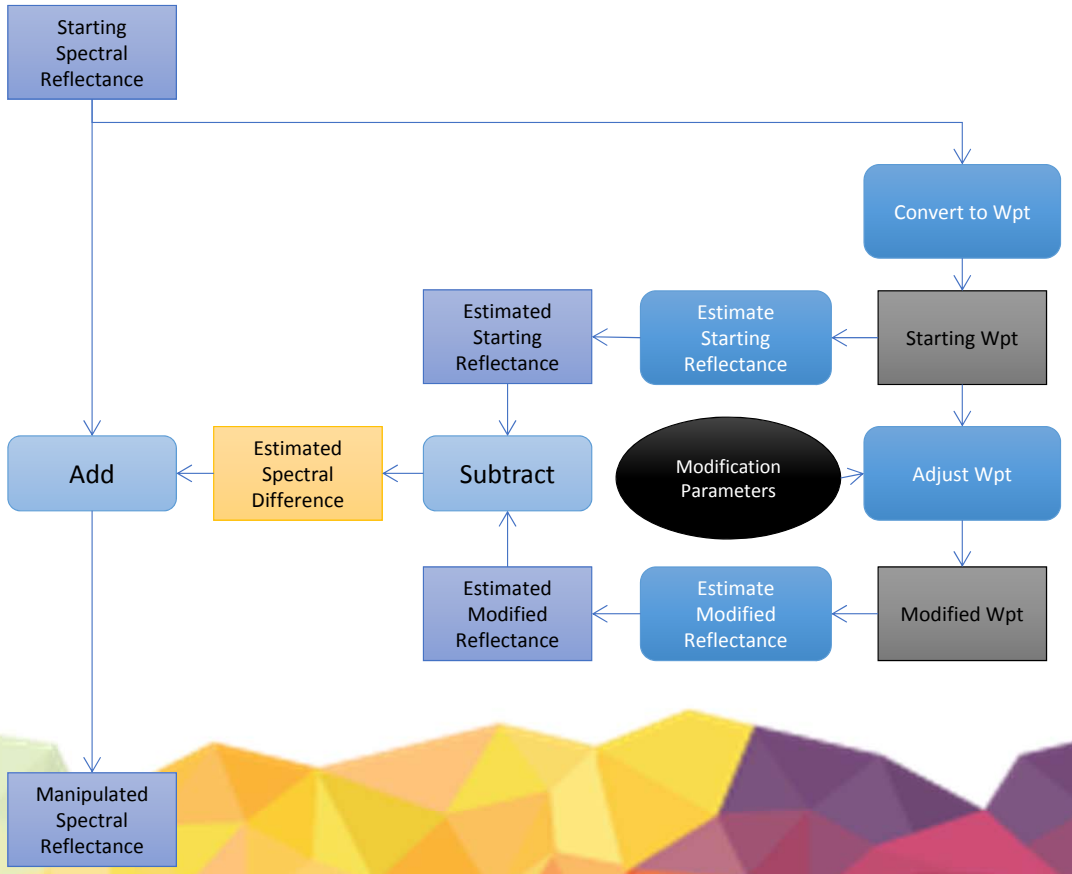
1. Degammafy RGB to get Linear RGB
2. Apply matrix (concatenated RGB to XYZ and XYZ to Wpt matrices) to convert Linear RGB to Wpt
3. Convert Wpt to polar notation (W, C, h)
4. Use normalized h with tintArrayElement to get $\mathbf{O}_{\text{select}}$ and $\mathbf{M}_{\text{select}}$ for hue
5. Determine g and s scalars from W and C
6. Estimated spectral reflectances determined as linear combination of g, s, and $\mathbf{O}_{\text{select}}$

- Examples: srgbRef.icc and argbRef.icc in Testing\SpecRef

RGB To Spectral Reflectance

Manipulating Spectral Reflectance

1. Apply matrix to *starting spectral reflectance* to get *starting Wpt*
2. Estimate *starting estimated reflectance* from *starting Wpt*
3. Adjust *starting Wpt* to get *modified Wpt*
4. Estimate *modified estimated reflectance* from *modified Wpt*
5. Subtract *modified estimated reflectance* from *starting estimated reflectance*
6. Add difference to *starting spectral reflectance* to get *manipulated spectral reflectance*

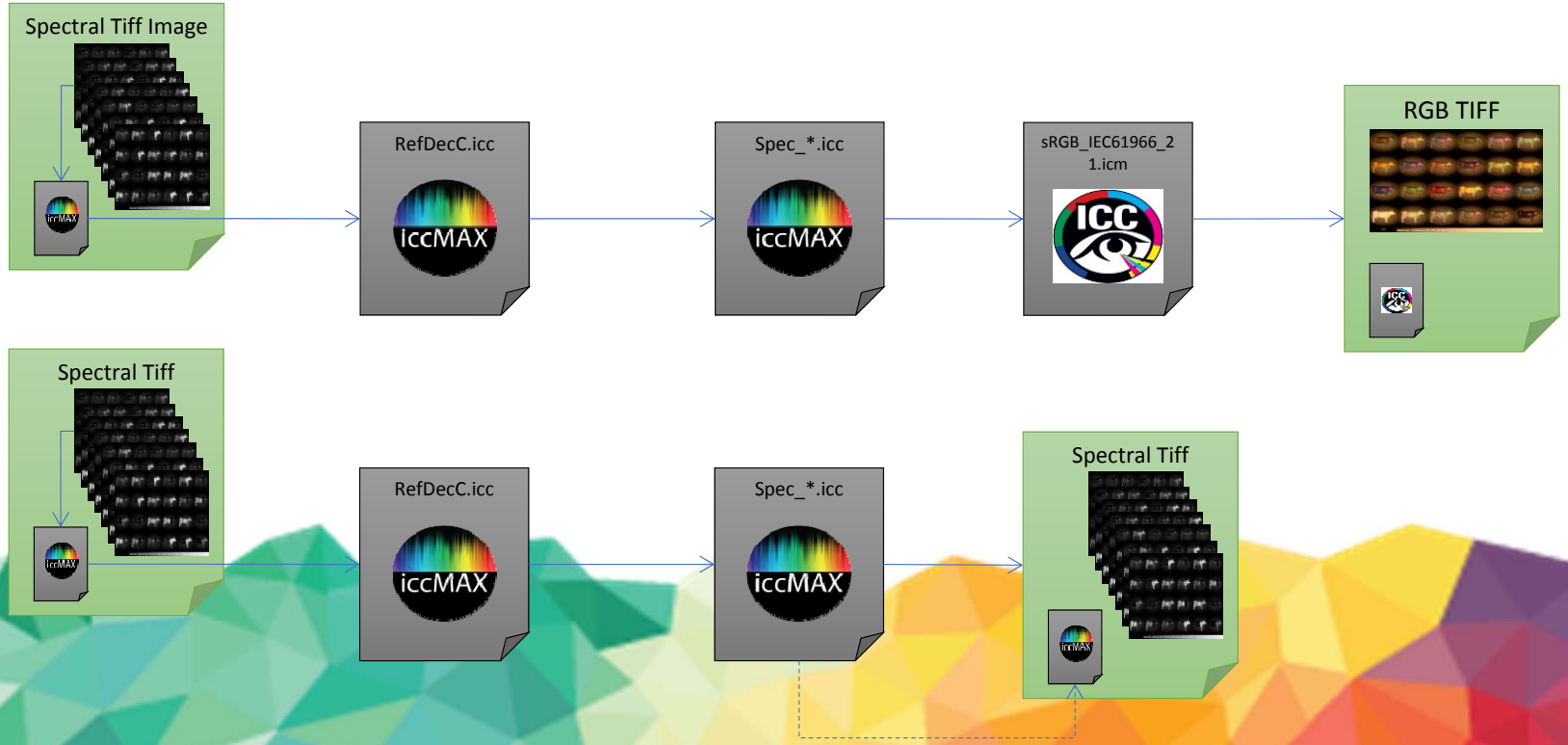


Using Abstract Reflectance iccMAX Profiles

- Several abstract iccMAX profiles in Testing\SpecRef folder demonstrate spectral reflectance Manipulation
  - RefDecC decreases chroma
  - RefDecH decreases hue
  - RefIncW increases lightness
  - RefMetaBlack adds different metameric blacks based on lightness

Applying spectral manipulation Profiles

- Use iccApplyProfiles with following profile sequences



Estimating and Manipulating spectral Reflectance with iccMAX Profiles

Demonstration & Practice

Exploring workflows with iccMAX profiles

Various Profiles in Testing Folder

Possible Workflows to Explore

- Observer variability with Rec2020 RGB profiles
- Packaging workflows using MCS profiles
- Camera modeling
- Using Reduced dimensionality CLUTs with Higher dimensionality color spaces
- Fluorescence in Named Color Profiles

Exploring workflows with iccMAX profiles

Demonstration & Practice

Wrap-UP

In Conclusion...

Workshop Review

- Becoming familiar with RefIccMAX
  - Install and build libraries and tools
- Using RefIccMAX tools to generate, apply, view and manipulate iccMAX profiles
  - Fun with MetaCow
  - Applying named color profiles
- Developing with RefIccMAX's IccProfLib library
  - How to write code to apply profiles
- Working with different observers and illuminants
  - Understanding CATs, CAMs and MATs
  - Using excell and Matlab to create data for PCCs
  - Applying different observers with Rec2020 monitor profiles
- Manipulating spectral reflectance
  - How it is done
  - More fun with MetaCow
- Exploring workflows with iccMAX profiles
  - Brief look at other profiles in Testing folder (time permitting)
- Wrap-up

Reference Materials

- ICC web page
  - http://www.color.org
- iccMAX web page:
  - http://www.iccmax.org
- ICC specification documents:
  - http://www.color.org/icc_specs2.xalter
- iccMAX reference implementation:
  - https://github.com/InternationalColorConsortium/RefIccMAX
- Max Derhak's PhD dissertation
  - Spectrally Based Material Color Equivalency: Modeling and Manipulation
  - http://scholarworks.rit.edu/theses/8789/

Thank You

Questions?