The World's Sixth Sense®

# Using Color Correction in Blackfly S

Revised 11/24/2016

## Subject

Technical Application Note (TAN2016010): Using Color Correction in Blackfly S

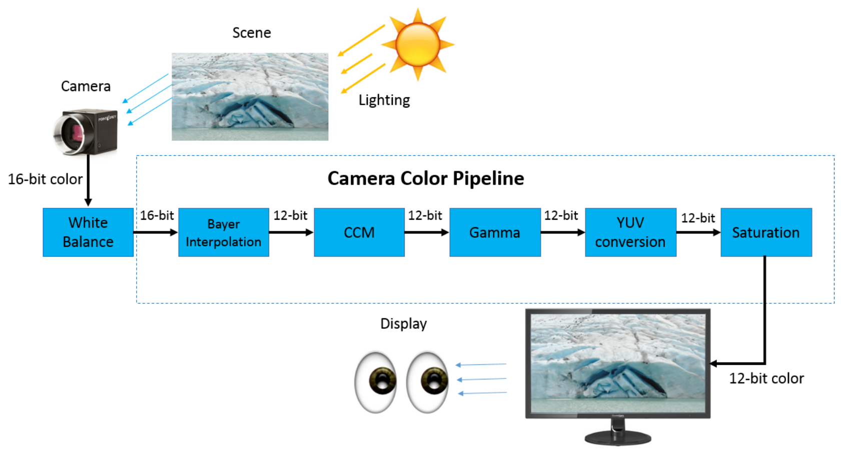
### Applicable Products

- Blackfly S and Spinnaker SDK

### Application Note Description

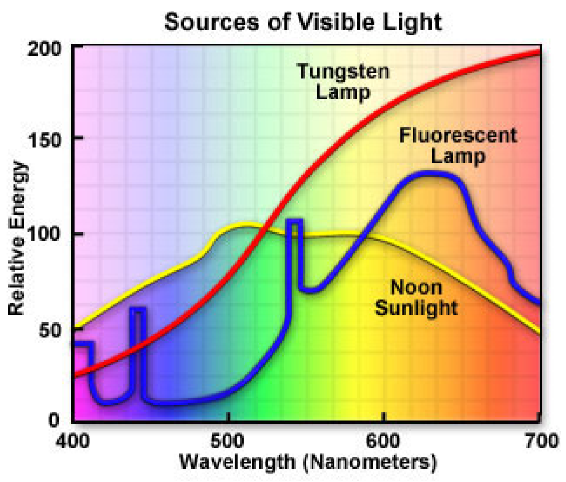
This document provides a description of the color correction matrix (CCM) used by the Blackfly S camera and provides instructions for the user to calculate a custom CCM.

## What is Color Correction?



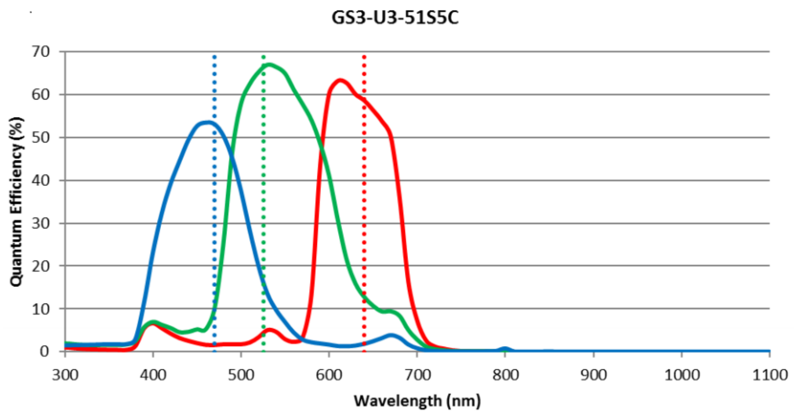
**Color Processing Pipeline**

Every sensor has a particular response to lighting and each lighting condition (e.g., sunlight, fluorescent light) has its own emission spectrum which affects how the image appears when captured.



**Emission spectrum of various light sources**

The sensor's efficiency at converting photons into electrons is measured as QE and varies depending on the wavelength.



**QE of GS3-U3-51S5C (IMX250 sensor)**

Color correction takes into account how each color channel interacts with the others, and scales each color channel dependently. A CCM is used to measure and compensate for these interactions.

## Why is Color Correction Important?

Reproducing accurate color is important in applications such as inspecting or sorting products, where minor differences in color can affect the accuracy and reliability of results. Blackfly S uses CCM transformations that correct the output image to sRGB color space. This is the most commonly used color space because it provides the "best guess" for how a monitor reproduces color. For more information on the different types of sRGB profiles and their uses, see: http://color.org/srgbprofiles.xalter.

## How is Color Correction Implemented?

Color correction is the process of converting the colors displayed in the source image (captured image) to a corrected color target image (final view).

The following example show the calculations for a single pixel:

$$\begin{bmatrix} R_{target} \\ G_{target} \\ B_{target} \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} R_{source} \\ G_{source} \\ B_{source} \end{bmatrix}$$

The expanded equation shows the results for each target:

$$R_{target} = a \times R_{source} + b \times G_{source} + c \times B_{source}$$

$$G_{target} = d \times R_{source} + e \times G_{source} + f \times B_{source}$$

$$B_{target} = g \times R_{source} + h \times G_{source} + i \times B_{source}$$

For an image size of n x m pixels, this calculation is performed n x m times.

## What is the Difference between CCM and White Balance?

CCM takes into account how the color channels interact with each other and scales the separate channels accordingly. A CCM is non-diagonal and the target RGB values are a function of the RGB values as described above.

White Balance (WB) adjusts for the emission spectrum by scaling each R, G, or B channel independently. A white balance matrix is a diagonal matrix and the target RGB values are scaled by a constant from their source RGB values.

$$\mathrm{WB} = \begin{bmatrix} x & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & z \end{bmatrix} \qquad \mathrm{CCM} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

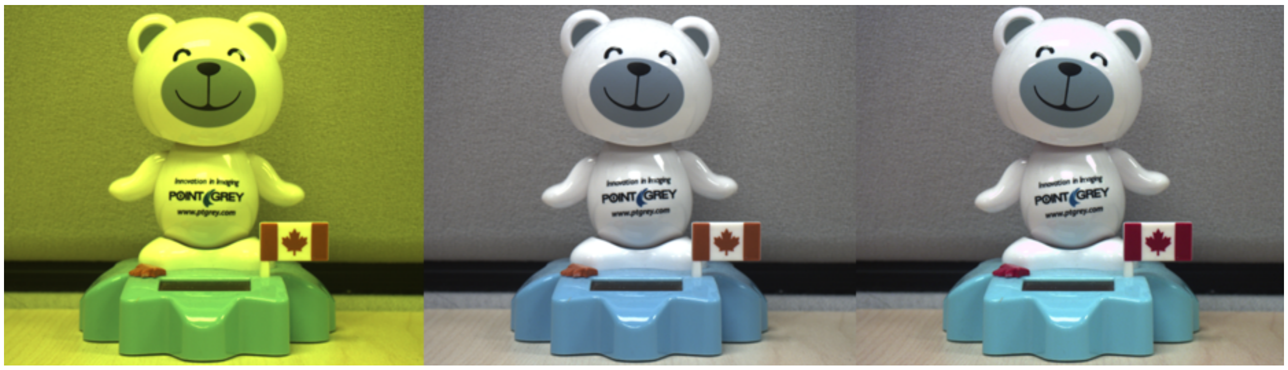
The white balance adjusts colors according to the following equations:

$$R_{target} = x \times R_{source}$$

$$G_{target} = y \times G_{source}$$

$$B_{target} = z \times B_{source}$$

The following images were taken in warm fluorescent light conditions. Compare the results with no color correction methods, white balance only, and white balance and CCM enabled.



**From right to left: WB off, CCM off / WB on, CCM off / WB on, CCM on**

In this example of fluorescent lighting, enabling CCM provides the following transformation to the source image:

$$R_{target} = 0.61R_{source} + 0.13G_{source} - 0.02B_{source}$$

$$G_{target} = -0.43R_{source} + 1.02G_{source} + 0.42B_{source}$$

$$B_{target} = 0.04R_{source} - 0.15G_{source} + 1.2B_{source}$$

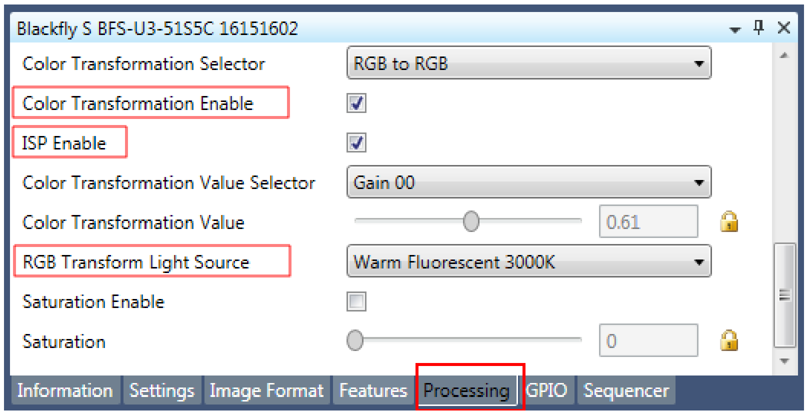
## How Do I Correct the Color of Images with Blackfly S?

Blackfly S uses predefined CCMs to create the correct sRGB output for different lighting conditions. The predefined CCMs are integrated into the RGB Transform Light source because the CCM cannot be independent of the White Balance (Transform Light Source). The CCM may be set using the SpinView GUI or the API.

The Color Transformation Selector shows the standard option of RGB toRGB. If a different pixel format is used, such as YUV or YCbCr pixel format, the Color Transformation Selector offers the option of RGB to YUV. The CCM changes to account for how YUV encodes color information

To set the CCM in SpinView:

1. Connect the camera to the PC.
2. Open SpinView.
3. Click on the Processing tab and check ISP Enable.
4. Check Color Transformation Enable.
5. In RGB Transform Light Source, select the appropriate RGB Transform Light Source (warm fluorescent, cool fluorescent, sunny, cloudy, tungsten, shade, general).


Blackfly S BFS-U3-51S5C 16151602

| | |
|---|---|
| Color Transformation Selector | RGB to RGB |
| Color Transformation Enable | ✓ |
| ISP Enable | ✓ |
| Color Transformation Value Selector | Gain 00 |
| Color Transformation Value | 0.61 |
| RGB Transform Light Source | Warm Fluorescent 3000K |
| Saturation Enable | |
| Saturation | 0 |

To enable CCM in SpinAPI, use the following commands (C++ example):

```
CBooleanPtr ptrIspEnable = nodeMap.GetNode("IspEnable");
ptrIspEnable->SetValue(1);
CBooleanPtr ptrColorTransformEnable = nodeMap.GetNode("ColorTransformationEnable");
ptrColorTransformEnable->SetValue(1);
CEnumerationPtr ptrRgbTransformLightSource = nodeMap.GetNode
("RgbTransformLightSource");
CEnumEntryPtr ptrRgbTransformationLightSourceWarm = ptrRgbTransformLightSource-
>GetEntryByName("WarmFluorescent3000K");
ptrRgbTransformLightSource->SetIntValue(ptrRgbTransformationLightSourceWarm->GetValue
());
```

## How to Derive Your Own CCM

There are several types of standard target image colors. Below are the typical color targets used:

- sRGB: standard RGB color space. This is the most common color space used for monitors, printers, and the Internet.
- Adobe RGB: provides a larger color space than sRGB.
- CIE XYZ: the color space that maps how a human eye responds to specific wavelengths of light. sRGB and Adobe RGB are subsets of this space.

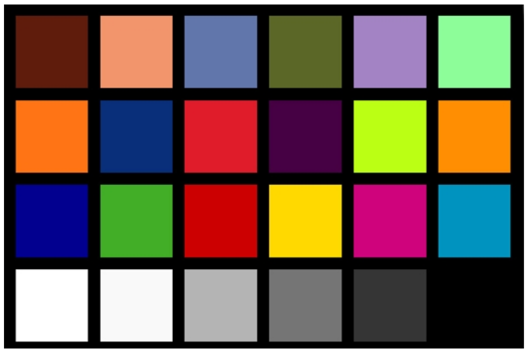
In some situations, the default settings may not be sufficient for your application. For example, if the target color is not sRGB, it is necessary to use a custom CCM. The example below shows how to convert from the image colors of one camera (source camera) to the image colors of camera (target camera).

To derive a custom CCM:

1. Use the following target and source notation to define the target and source cameras.

$$\begin{bmatrix} R_{target} \\ G_{target} \\ B_{target} \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} R_{source} \\ G_{source} \\ B_{source} \end{bmatrix}$$

2. Take an image of a standardized color calibration chart under standard lighting conditions. (This example shows a Macbeth color checker with 24 squares.)



3. Determine the average pixel color value in each square to establish 24 unique colors to compare. In these calculations, k = 24 as the 24 average pixel colors in each square:

$$\begin{bmatrix} R_{T_1} & G_{T_1} & B_{T_1} \\ \vdots & \vdots & \vdots \\ R_{T_k} & G_{T_k} & B_{T} \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} R_{S_1} & G_{S_1} & B_{S_1} \\ \vdots & \vdots & \vdots \\ R_{S_k} & G_{S_k} & B_{S_k} \end{bmatrix}$$

   Perform these calculations for both the Target camera [T] and Source camera [S].

4. Determine the CCM using matrix multiplication:

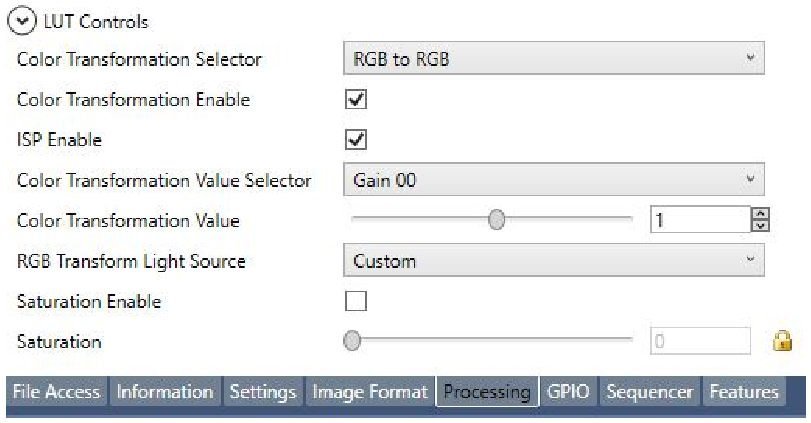
$$[\mathbf{T}]_{kx3} = [\mathbf{CCM}]_{3x3}[\mathbf{S}]_{kx3}$$

5. With the CCM defined, map the CCM values to the corresponding parameters in the Blackfly S camera. The example below shows mapping the CCM entry to a specific node in the camera.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} \mathrm{Gain}_{00} & \mathrm{Gain}_{01} & \mathrm{Gain}_{02} \\ \mathrm{Gain}_{10} & \mathrm{Gain}_{11} & \mathrm{Gain}_{12} \\ \mathrm{Gain}_{20} & \mathrm{Gain}_{21} & \mathrm{Gain}_{22} \end{bmatrix}$$

6. Use SpinView to set the gain values for your custom CCM:
   a. From the RGB Transform Light Source drop-down list, select Custom.
   b. From the Color Transformation Value Selector drop-down, select the specific Gain (e.g., Gain 00, Gain01, etc.).
   c. In the Color Transformation Value, enter the calculated CCM index value.

   Repeat steps b-c to set all Gain Values for the CCM.



LUT Controls

| | |
|---|---|
| Color Transformation Selector | RGB to RGB |
| Color Transformation Enable | ✓ |
| ISP Enable | ✓ |
| Color Transformation Value Selector | Gain 00 |
| Color Transformation Value | 1 |
| RGB Transform Light Source | Custom |
| Saturation Enable | ☐ |
| Saturation | 0 |

Example code to set the custom CCM in SpinAPI (C++ example)

```
CBooleanPtr ptrIspEnable = nodeMap.GetNode("IspEnable");
ptrIspEnable->SetValue(1);
CBooleanPtr ptrColorTransformationEnable = nodeMap.GetNode
("ColorTransformationEnable");
ptrColorTransformationEnable->SetValue(1);
CEnumerationPtr ptrRgbTransformLightSource = nodeMap.GetNode
("RgbTransformLightSource");
CEnumEntryPtr ptrRgbTransformationLightSourceCustom = ptrRgbTransformLightSource-
>GetEntryByName("Custom");
ptrRgbTransformLightSource->SetIntValue(ptrRgbTransformationLightSourceCustom-
>GetValue());
CEnumerationPtr ptrColorTransformationValueSelector = nodeMap.GetNode
("ColorTransformationValueSelector");
CEnumEntryPtr ptrGain00 = ptrColorTransformationValueSelector->GetEntryByName
("Gain00");
CFloatPtr Gain00Value = nodeMap.GetNode("ColorTransformationValue");
Gain00Value->SetValue([Enter CCM Value]);
```

## Downloads and Support

FLIR endeavors to provide the highest level of technical support possible to our customers. Most support resources can be accessed through the Support section of our website.

The first step in accessing our technical support resources is to obtain a Customer Login Account. This requires a valid name and email address. To apply for a Customer Login Account go to our Downloads page.

Customers with a Customer Login Account can access the latest **software** and **firmware** for their cameras from our website. We encourage our customers to keep their software and firmware up-to-date by downloading and installing the latest versions.

### Finding Information

**Spinnaker SDK**—The Spinnaker SDK provides API examples and the SpinView camera evaluation application. Available from our Downloads page.

**API Documentation**—The installation of the Spinnaker SDK comes with API references for C++, C#, and C code. A Programmer's Guide is included in each of the references. Available from:

- Start Menu→All Programs→Point Grey Spinnaker SDK→Documentation
- The SpinView application Help menu

**Getting Started with SpinView**—A quick guide to using the SpinView camera evaluation application provided in the Spinnaker SDK. Available from:

- Start Menu→All Programs→Point Grey Spinnaker SDK→Documentation
- The SpinView application Help menu
- Camera Reference zip package

**Camera Reference**—A zip package containing PDF and HTML copies of the camera's references, including: Installation Guide, Technical Reference, and Getting Started. Available from our Downloads page.

**Knowledge Base**—A database of articles and application notes with answers to common questions as well as articles and tutorials about hardware and software systems. Available from our Knowledge Base.

**Learning Center**—Our Learning Center contains links to many resources including videos, case studies, popular topics, other application notes, and information on sensor technology.

### Contacting Technical Support

Before contacting Technical Support, have you:

1. Read the product documentation?
2. Searched the Knowledge Base?
3. Downloaded and installed the latest version of software and/or firmware?

If you have done all the above and still can't find an answer to your question, contact our Technical Support team.

## Additional Resources

**GenICam**—A programming interface for cameras and devices. More information available on the EMVA.org website.

**USB3 Vision**—A vision standard for the USB 3.0 interface that uses GenICam. More information available on the AIA Vision Online website.