Adobe Systems' Implementation of Black Point Compensation

Adobe

Adobe Systems' Implementation of Black Point Compensation

Adobe Systems' Implementation of

Black Point Compensation

1 Product Information

Black Point Compensation is a feature found in several Adobe Systems products, including Adobe Photoshop. Adobe Systems implemented Black Point Compensation (BPC) to address color conversion problems caused by differences between the darkest level of black achievable on one device and the darkest level of black achievable on another.

2 Audience

This document explains Adobe Systems' Black Point Compensation algorithm to the following kinds of developers and other readers.

- Members of the International Color Consortium (ICC) may want to understand Black Point Compensation in order to codify it as a standard. If codified as a standard, the conversion of black in ICC profiles could be brought up to the same level of quality as the conversion of white.
- Developers of image-manipulation products outside Adobe Systems may want to learn how to implement BPC themselves, in order to improve their own products' conversion of images from one ICC profile to another.
- Mathematically-minded users of Adobe Systems products may want to understand how their images get converted by BPC.
- Adobe Systems partners who develop plug-ins using Software Developer's Kits for Adobe Systems products may want to understand the effects of enabling or disabling BPC.
- Vendors who write applications that create ICC profiles may want to add BPC as a visualization option.

3 References

This document uses the following terms: *CMYK*, *color space*, *gamut*, *ICC profiles*, *L*a*b**, *LUT*, *PCS*, *rendering intent*, *RGB*, and *XYZ*. Readers are expected to be familiar with these terms and concepts, as well as with the various tags and other elements of ICC profiles. Introductory material about terminology, color conversion, and color profiles is available in the following documents:

- Tutorials on ICC color management and specifications for ICC profiles: http://www.color.org
- White paper "Color Consistency and Adobe Creative Suite":
  http://www.adobe.com/products/creativesuite/pdfs/cscolormgmt.pdf

4 Introduction

This paper explains the algorithm used for Black Point Compensation (BPC) in several Adobe Systems products and includes these topics:

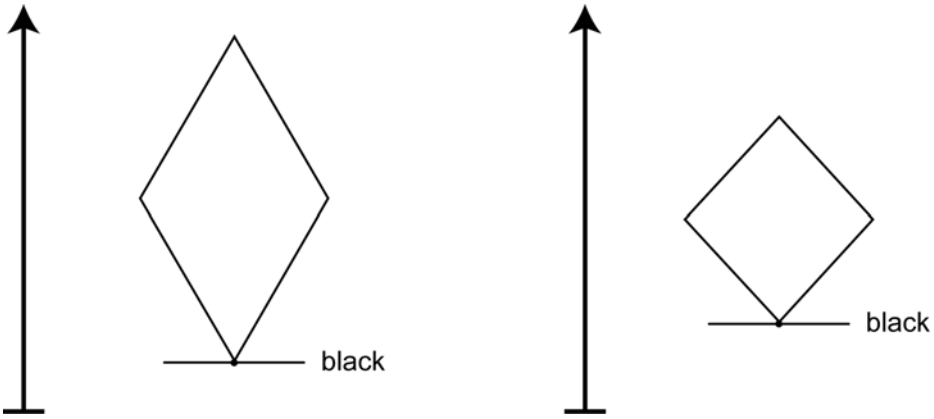
- "Why Black Point Compensation is necessary" explains what common color conversation problem is addressed by the Adobe Systems Black Point Compensation algorithm.

- "When Black Point Compensation is performed" describes the circumstances in which an Adobe Systems product does or does not perform BPC.
- "How Black Point Compensation is performed" describes in detail what the BPC algorithm does.

5 Why Black Point Compensation is necessary

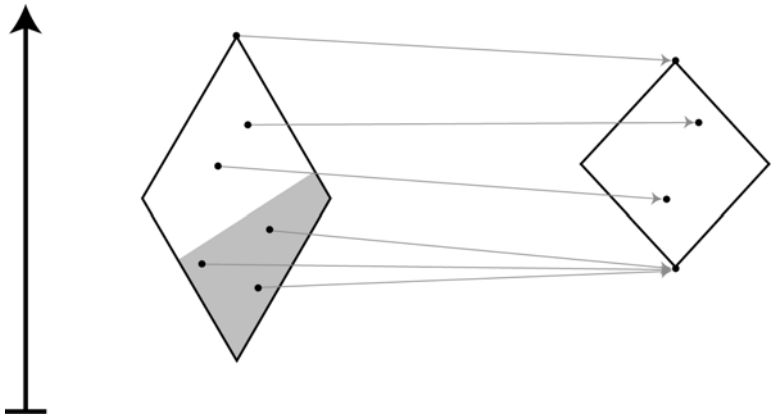
When an Adobe Systems product is used to convert an image from the color space of one device to the color space of another device, the Adobe Systems product performs color conversion.

Figure 1.



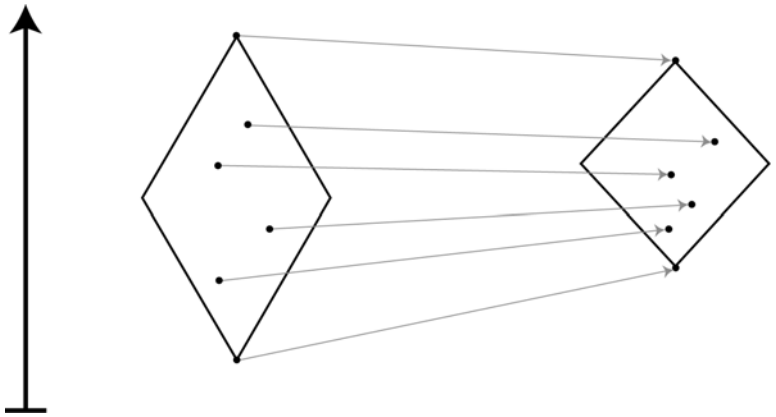
The color conversion algorithm consults the ICC profiles of the two devices (the source device and destination device) and the user's rendering intent (or intent) in order to perform the conversion. Although ICC profiles specify how to convert the lightest level of white from the source device to the destination device, the profiles do not specify how black should be converted. The user observes the effect of this missing functionality in ICC profiles when a detailed black or dark space in an image is transformed into an undifferentiated black or dark space in the converted image. The detail in dark regions (called the *shadow section*) of the image can be lost in standard color conversion.

Figure 2.



Adobe Systems implemented BPC to address this conversion problem by adjusting for differences between the darkest level of black achievable on one device and the darkest level of black achievable on another. Because BPC is an optional feature that the user can enable or disable when converting an image, the user can always decide whether the conversion of a particular image looks better with or without BPC.

Figure 3.



6 When Black Point Compensation is performed

Adobe Systems products choose whether or not to perform Black Point Compensation based on:

- The user's preference
- Rendering intent
- Source and destination profiles

6. 1. User preference

By default, BPC is enabled in all applicable situations unless the user explicitly deselects it through the product's user interface.

6. 2. Rendering intent

Typically, BPC is performed for conversions using the Relative Colorimetric intent. BPC is not available for conversions using Absolute Colorimetric intent.

Color conversion using Perceptual intent already maps source white to destination white and source black to destination black. Because this mapping preserves the relationships of the shades, it is unlikely that a whole shadow section will be mapped to the same black value. Therefore, BPC should not be necessary. BPC is available, however, for this rendering intent, to be used with malformed profiles. For a given picture, the user can decide whether using BPC improves the color conversion and can select it or deselect it accordingly.

BPC is available for color conversion using the Saturation intent. As with Perceptual intent, the user may or may not find that selecting BPC improves the conversion of a given image.

6. 3. Source and destination profiles

The purpose of BPC is to convert points in the gamut of one profile into points in the gamut of another. Therefore, when the source and destination profiles are identical, BPC is unnecessary and has no effect.

The Source profile may be Version 2 or Version 4, and the Destination profile may be Version 2 or Version 4. The versions of the source and destination profiles do not need to match.

BPC is available if the Source or Destination profile is Input, Display, Output, or Color Space.

The Destination profile either must have either Matrix/TRC tags, or it must have both AtoB* and BtoA* tags.

At present, neither the Source profile of the content nor the Destination profile of the content may be any of the following: Named Color, DeviceLink, Abstract.

Although ICC profiles can use generalized n-Color spaces, Adobe Systems currently does not apply its BPC algorithm to such color spaces.

7 How Black Point Compensation is performed

The rest of this document explains the Black Point Compensation algorithm. The algorithm consists of three phases:

1. Estimating the darkest point of the Source profile (**SourceBlackPoint**)
2. Estimating the darkest point of the Destination profile (**DestinationBlackPoint**)
3. Computing the mapping from the **SourceBlackPoint** to the **DestinationBlackPoint**

The algorithm depends only on the Source and Destination profiles, not on any points in a particular image. Therefore, the BPC algorithm can compute the mapping between given Source and Destination profiles once, and then efficiently apply that mapping to many images.

In the following discussion, the following transforms, variables, and constants will be used:

Functions

T is the function to transform a point in one profile space to another, using a given intent. We write

y = **T**(**x**, *Source Space, Destination Space, Rendering Intent*)

where **x** is the point in the Source Space and **y** is the point in Destination Space.

Variables

DestinationBlackPoint is the darkest point in the Destination profile space.

DProfile is the Destination profile (or its associated space).

SourceBlackPoint is the darkest point in the Source profile space.

SProfile is the Source profile (or its associated space).

UserIntent is the rendering intent of the conversion from Source profile space to Destination profile space.

Constants

Standard intents or color spaces are written thus:

*PerceptualIntent, RelativeColorimetric, L*a*b*, XYZ,* etc.

7. 1. Estimating the Source Black Point

The BPC algorithm first estimates a **SourceBlackPoint** as described below:

Step 1: Compute the Source Black Point:

if (**SProfile** is a CMYK Output Profile) {
intermediateCMYK = **T**((0,0,0), *L*a*b**, **SProfile**, *PerceptualIntent*)
Let **SourceBlackPoint** = **T**(**intermediateCMYK**, **SProfile**, *L*a*b**, **UserIntent**) }
else /* that is, if **SProfile** is not a CMYK Output Profile*/ {
Let **LocalBlack** be the value of black in the **SProfile** space
(for example, (0 0 0) for RGB, (100 100 100 100) for CMYK).
Let **SourceBlackPoint** = **T**(**LocalBlack**, **SProfile**, *L*a*b**, **UserIntent**). }

Now we have an estimated **SourceBlackPoint** in *L*a*b** space, consisting of the triple ($\mathbf{L^*_{SBP}}$, $\mathbf{a^*_{SBP}}$, $\mathbf{b^*_{SBP}}$).

Because CMYK profiles sometimes produce a result that is non-neutral, we set $\mathbf{a^*_{SBP}}$ and $\mathbf{b^*_{SBP}}$ to zero.

if (**SProfile** is a CMYK Profile) { /* The algorithm will use only the $\mathbf{L^*_{SBP}}$ value. */
Set $\mathbf{a^*_{SBP}} = 0$.
Set $\mathbf{b^*_{SBP}} = 0$. }

Step 2: Clip the value of $\mathbf{L^*_{SourceBlackPoint}}$ (abbreviated as $\mathbf{L^*_{SBP}}$) from its possible range of [0..100] to the range [0..50]:

if ($\mathbf{L^*_{SBP}}$) > 50 {
Set $\mathbf{L^*_{SBP}} = 50$ } .

The value of **SourceBlackPoint** is now guaranteed to have its first component $\mathbf{L^*_{SBP}}$ in the range [0..50]. The value of **SourceBlackPoint** will be used in Section 7. 3.

7. 2. Estimating the Destination Black Point

The Destination Black Point depends both on the Source and Destination profiles as well as the user's intent.

If the **DProfile** is not LUT-based Gray, RGB, or CMYK, then estimate the Destination Black Point by analogy with the method described in Section 7. 1: substitute **DProfile** for **SProfile** and **DestinationBlackPoint** for **SourceBlackPoint**. The resulting value for **DestinationBlackPoint** will be used in Section 7. 3.

Otherwise, if the **DProfile** is one of LUT-based Gray, RGB, or CMYK, perform the following process.

Step 1: Define a new variable **InitialLab** with components ($\mathbf{L^*_{Init}}$, $\mathbf{a^*_{Init}}$, $\mathbf{b^*_{Init}}$) and calculate its value: initial *L*a*b** value **InitialLab**.

a If (**UserIntent** is *RelativeColorimetric*) {

Calculate **InitialLab** from the Destination profile using the method described in Section 7. 1, substituting **DProfile** for **SProfile** and **InitialLab** for **SourceBlackPoint**. }

b Else if (**UserIntent** is *Perceptual* or *Saturation*) {

Set **InitialLab** to (0, 0, 0). }

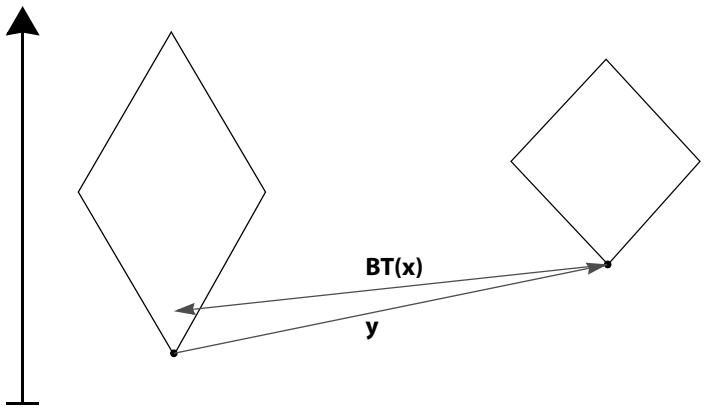
Step 2: Computing a *Destination Black Transform* **BT**:

Define a Transform **BT** for all **x** in *L*a*b** space as shown in Figure 4.

Let **y** = **T** (**x** , *L*a*b**, **DProfile**, **UserIntent)**

Define **BT**(**x**) = **T** (**y**, **DProfile**, *L*a*b**, *RelativeColorimetric*).

Figure 4.



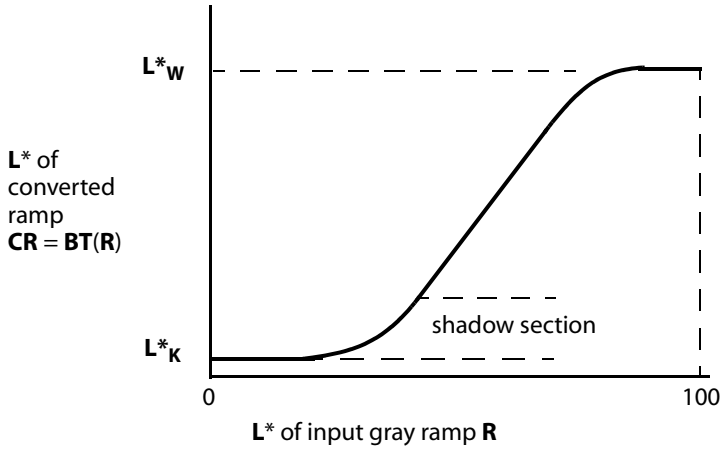
Step 3: Attempt to set **DestinationBlackPoint** using the easiest method:

If **UserIntent** is *RelativeColorimetric*
  Declare Boolean **NearlyStraightMidrange** as true
  Set **MinL** = **L** component of **BT**((0, $\mathbf{a}^*_{Init}$, $\mathbf{b}^*_{Init}$))
  Set **MaxL** = **L** component of **BT**((100, $\mathbf{a}^*_{Init}$, $\mathbf{b}^*_{Init}$))
  For ℓ ranging from 0 to 100 in steps of 1
    L = **L** component of **BT**((ℓ, $\mathbf{a}^*_{Init}$, $\mathbf{b}^*_{Init}$))
    If **L** > (**MinL** + 0.2 × (**MaxL** - **MinL**)) then
      if (|**L** - ℓ| > 4) then
        Set **NearlyStraightMidRange** to false
If **NearlyStraightMidRange** is true
  Set **DestinationBlackPoint** to **InitialLab** and skip step 4.

Conceptually, this pseudocode tests how close the source ℓ and converted **L** are to one another in the mid-range of the values. If the converted ramp of **L** values is close enough to a straight line **y=x**, then **InitialLab** is good enough to be the **DestinationBlackPoint**, and the algorithm proceeds to Section 7. 3, "Computing the mapping from SourceBlackPoint to DestinationBlackPoint".

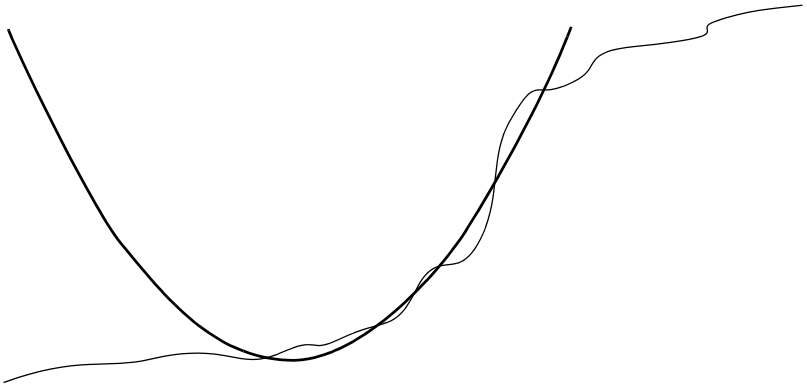
Step 4: For the remaining cases, i.e., if **NearlyStraightMidRange** is set to false in Step 3, the BPC algorithm estimates the Destination Black Point from the **L***-curve of **CR=BT(R)**, which we will refer to as the round-trip curve. The round-trip curve normally looks like a nearly constant section at the black point, with a corner and a nearly straight line to the white point.

Figure 5.



The algorithm estimates the length of the constant section to estimate the Destination Black Point. The problem is that there is often noise in the constant section, and the corner is often rounded. The algorithm ignores any "toe" around the black point and estimates exactly where the extrapolated round trip curve would intersect the nearly constant black limited area. The algorithm then fits a least squares error quadratic curve though the shadow section of the curve, and it uses the point where this curve intersects the **L*=0** round trip value as the Destination Black Point.

Figure 6.



Step 4 is executed as follows:

i Let L^*_K = L* of converted ramp at input L* = 0, and L^*_W = L* of converted ramp at input L* = 100.

ii Let **y** = $(L - L^*_K) / (L^*_W - L^*_K)$ for L = L^*_K .. L^*_W (y varies from 0 .. 1). In the above graph, L^*_K is **y** = 0, and L^*_W is y = 1.

iii Let the shadow section be points on curve such that $0.1 \le \mathbf{y} < 0.5$, for Relative Colorimetric intent, or $0.03 \le \mathbf{y} < 0.25$ for Perceptual or Saturation intents.

iv Fit a least squares error quadratic curve $\mathbf{y} = \mathbf{tx}^2 + \mathbf{ux} + \mathbf{c}$ through the shadow section. For points (**x**, **y**) in the shadow section, values of **x** will be the **L*** of the input gray ramp, and **y** will be the **L*** of the converted ramp, scaled to 0 .. 1 (see Step 2.e.ii. above).

v Compute the **x**-coordinate of the vertex of the quadratic curve: **x = -u/2t**. The vertex of the fitted quadratic curve is an approximation for when the shadow region intersects the constant section ($\mathbf{L^*_K}$). Use this value, **x**, as the **L*** of the Destination Black Point.

7. 3. Computing the mapping from SourceBlackPoint to DestinationBlackPoint

The final step of the BPC algorithm is to construct a transform that can later be used to efficiently convert color values from the **SProfile** data space to the **DProfile** data space, while compensating for black points, according to **UserIntent**. Only the **L*** values of the estimated **SourceBlackPoint** and **DestinationBlackPoint** are required; **a*** and **b*** are ignored.

The entire transformation from source to destination is descibed below, as a series of logical steps. The actual computation differs somewhat, in the interests of efficiency. The logical steps are:

Step 1: Convert the color value from the **SProfile** data space to the **SProfile** PCS according to **UserIntent**. This is the standard ICC conversion defined by **SProfile**.

Step 2: Convert the color value from the **DProfile** to a flat *XYZ* space with white point = (1,1,1). For example, when the Source PCS is *XYZ* with a white point of (0.9642, 1.0, 0.8249),

Flat **X** = (1 / 0.9642) * PCS **X**
Flat **Y** = PCS **Y**
Flat **Z** = (1 / 0.8249) * PCS **Z**

Step 3: Convert the color value, $\mathbf{XYZ}_{IN}$, to the color value, $\mathbf{XYZ}_{OUT}$ by performing Black Point scaling.

To map **L** to **Y**, the BPC algorithm uses the function DecodeL (**L**), based on the standard conversion from *L*a*b** to *XYZ*. DecodeL(**L**) is defined in the table below:

L	DecodeL (**L**)
$\mathbf{L} < 0$	- DecodeL (-**L**)
$0 \le \mathbf{L} \le 8.0$	$\mathbf{L} \times (((8 + 16) / 116)^3) / 8.0$
$\mathbf{L} > 8.0$	$((\mathbf{L} + 16) / 116)^3$

To perform Black Point scaling, the algorithm scales from $\mathbf{XYZ}_{IN}$ to $\mathbf{XYZ}_{OUT}$ and adds an offset on each channel of *XYZ*:

$\mathbf{Y}_{DST}$ = DecodeL (**L** of **DestinationBlackPoint**)
$\mathbf{Y}_{SRC}$ = DecodeL (**L** of **SourceBlackPoint**)
Scale = $(1 - \mathbf{Y}_{DST}) / (1 - \mathbf{Y}_{SRC})$
Offset = (1 – Scale)
$\mathbf{X}_{OUT} = (\mathbf{X}_{IN} *$ Scale) + Offset
$\mathbf{Y}_{OUT} = (\mathbf{Y}_{IN} *$ Scale) + Offset
$\mathbf{Z}_{OUT} = (\mathbf{Z}_{IN} *$ Scale) + Offset

Step 4: Convert the $\mathbf{XYZ}_{OUT}$ value in the flat *XYZ* space to the **DProfile** PCS, which is either *XYZ* or *L*a*b**.

Step 5: Convert the color value from the **DProfile** PCS to the **DProfile** data space according to **UserIntent**. This is the standard ICC conversion defined by **DProfile**.

The Adobe Systems Black Point Compensation algorithm uses these five logical steps to create a transformation that can be used to convert a color value in the Source profile data space into a color value in the Destination profile data space according to the user's rendering intent (**UserIntent**). After the Adobe Systems BPC algorithm defines that transformation, the transformation can be used to efficiently convert color values, taking into account Black Point Compensation.