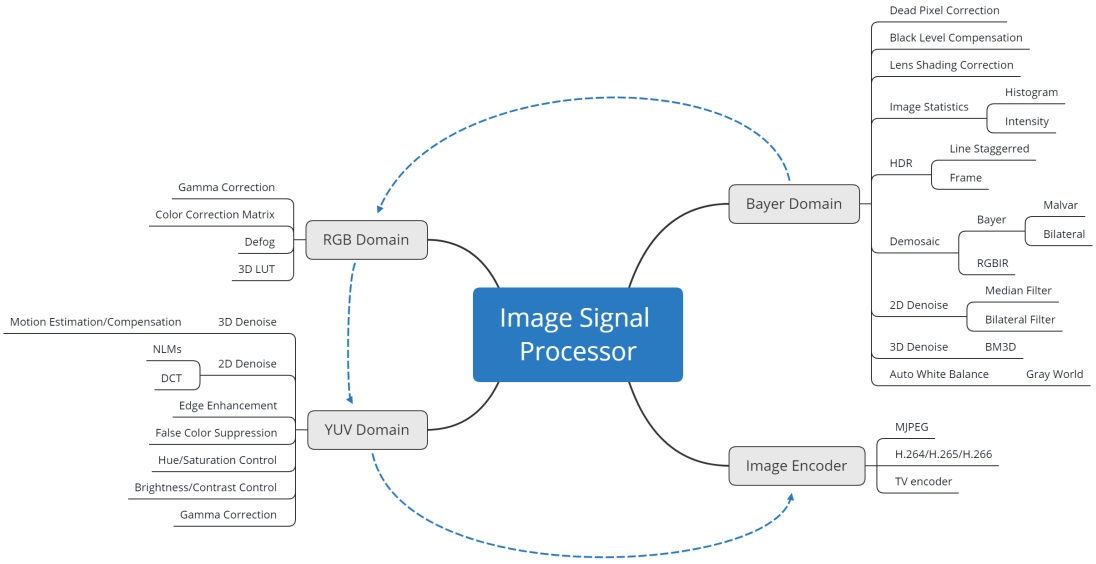
Image Signal Processor

Introduction

Image Signal Processor (ISP) is an application processor to do digital image processing, specifically for conversion from RAW image (acquired from Imaging Sensors) to RGB/YUV image (to further processing or display).

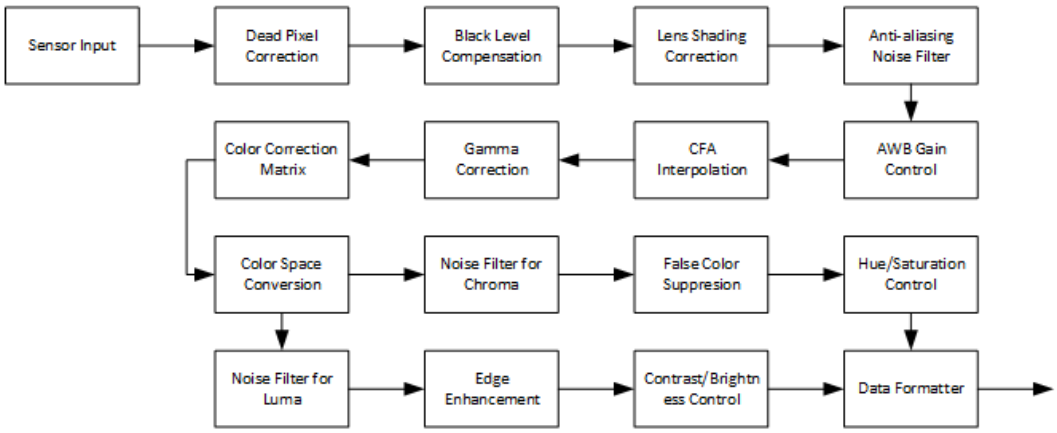


Objectives

This project aims to provide an overview of ISP and stimulate the whole ISP pipeline and some tuning functions. The proposed ISP pipeline consists of following modules, dead pixel correction (DPC), black level compensation (BLC), lens shading correction (LSC), anti-aliasing noise filter (ANF), auto white balance gain control (AWB), color filter array interpolation (CFA), gamma correction (GC), color correction matrix (CCM), color space conversion (CSC), noise filter for luma and chroma (NF), edge enhancement (EE), false color suppression (FCS), hue/saturation/control (HSC) and brightness/contast control (BCC).

Some advanced functions like wide/high dynamic range (W/HDR) and temporal/spatial noise filter (T/SNF) will be implemented furthermore.

ISP Pipeline



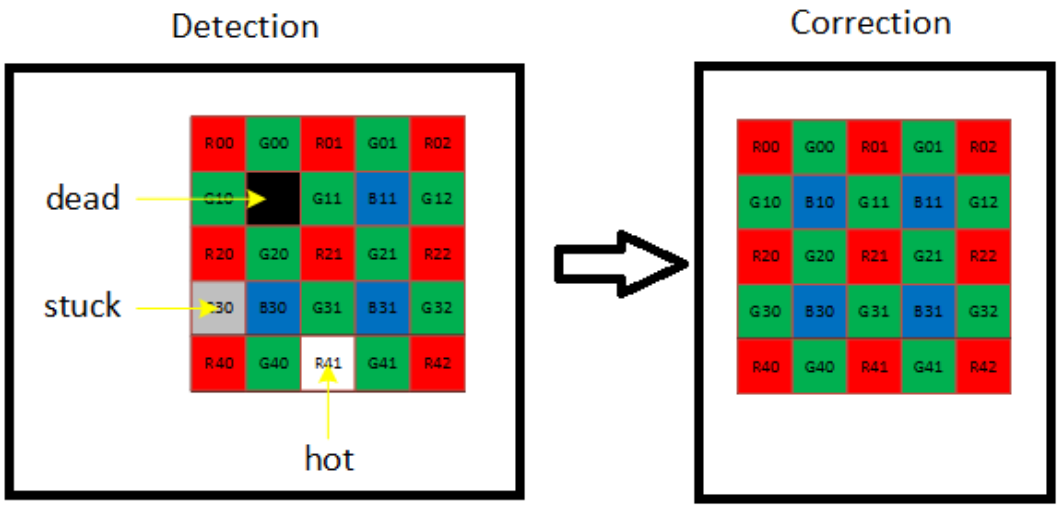
Bayer Domain Processing

Dead Pixel Correction

From the perspective of sensor manufacturing, an image sensor may have a certain number of defective pixels due to the uncertainty of manufacturing process, such as dusts, manufacturing faults, incompletion of exposure etc. In other words, the chance of having defective pixels only depends on the manufacturing process and cannot be avoidable ultimately but can be eliminated as much as possible.

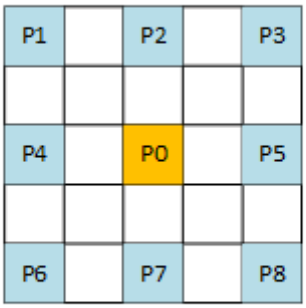
Based on the value of defects, the defective pixels can be classified into three types, dead (always low), hot (always high) and stuck (always a value between low and high). Based on the status of defective pixels, they are divided into two types as static dead pixel (always constant) and dynamic dead pixel (changed by some conditions like exposure or temperature).

There are two steps to realize the dead pixel correction function. The first one is to detect defective pixels. The second one is to replace the defects with interpolated values.



For stuck defects, it is easy to find the locations by capturing black or white images. Then the locations of known defects are stored into ISP memory for correction.

For dynamic defects, it is often estimated by using the neighborhood of defective pixels.



The differences between center pixel and neighbored 8 pixels are calculated.

$$diff(x) = abs(P_{ij} - P_{mn}) \tag{1}$$

One threshold `thres` is predefined to charge the pixel is dead or not.

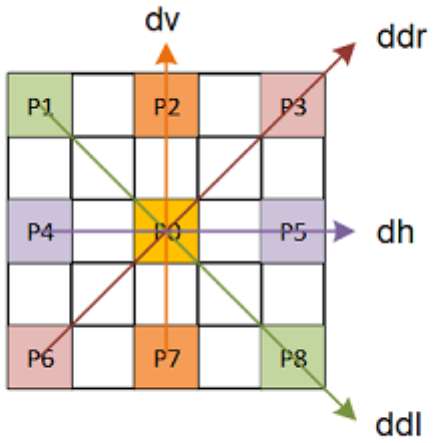
$$P_{ij} = \begin{cases} dead & \text{if \&(diff(x) } > \text{ thres) where x is 0-8} \\ not\ dead & \text{others} \end{cases} \tag{2}$$

Once the automatic dead pixel detection is done, the correction can be executed. Here lists two interpolation methods. One is mean filter, another one is gradient-based filter.

For mean filter, it is pretty easy and simple

$$P_{ij} = \frac{1}{4} \sum_{m=-2,n=-2}^{m=2,n=2} P_{i+m,j+n} \tag{3}$$

For gradient-based filter, first calculate the gradient of adjacent pixels on different directions.



From above figure, there are four directions gradient calculated.

$$\begin{aligned} dv &= |2P_0 - P_2 - P_7| \\ dh &= |2P_0 - P_4 - P_5| \\ ddr &= |2P_0 - P_1 - P_8| \\ ddl &= |2P_0 - P_3 - P_6| \end{aligned} \tag{4}$$

The output pixel is the average of adjacent pixels on the selected direction.

$$P_{ij} = \begin{cases} (P_{i,j-2} + P_{i,j+2} + 1)/2 & \text{if min(dv,dh,ddr,ddl)} == \text{dv} \\ (P_{i-2,j} + P_{i+2,j} + 1)/2 & \text{if min(dv,dh,ddr,ddl)} == \text{dh} \\ (P_{i-2,j-2} + P_{i+2,j+2} + 1)/2 & \text{if min(dv,dh,ddr,ddl)} == \text{ddl} \\ (P_{i+2,j-2} + P_{i-2,j+2} + 1)/2 & \text{others} \end{cases} \tag{5}$$

Black Level Compensation

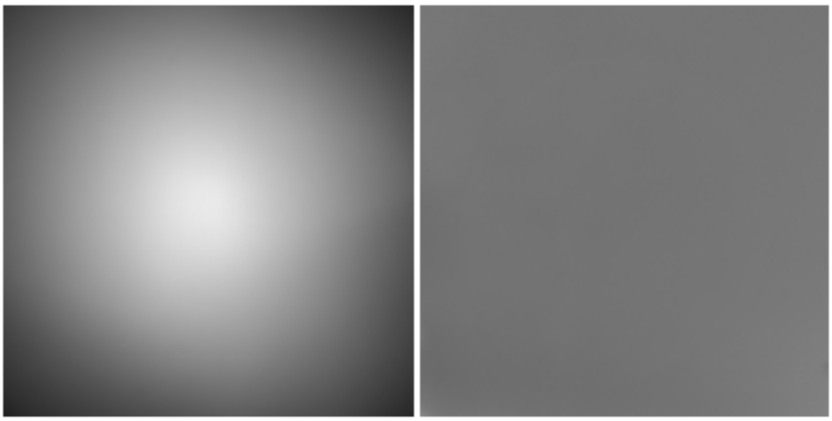
From the sensor characterization, the lowest output voltage of sensor is the black level voltage. Due to circuit design, the black level voltage cannot be zero. This voltage can be measured under the condition by setting exposure time and all gain (analog and digital) to smallest value. This leads to the fact that the image is not pure dark even the exposure time and gain are set to lowest value. The black level compensation is also called optical black correction. So the black level voltage should be corrected to result in a pure black image. The adjustment is simple as shown below.

$$R^{'} = R + R\_offset \tag{6}$$
$$Gr^{'} = Gr + Gr\_offset + \alpha R \tag{7}$$
$$Gb^{'} = Gb + Gb\_offset + \beta B \tag{8}$$
$$B^{'} = B + B\_offset \tag{9}$$

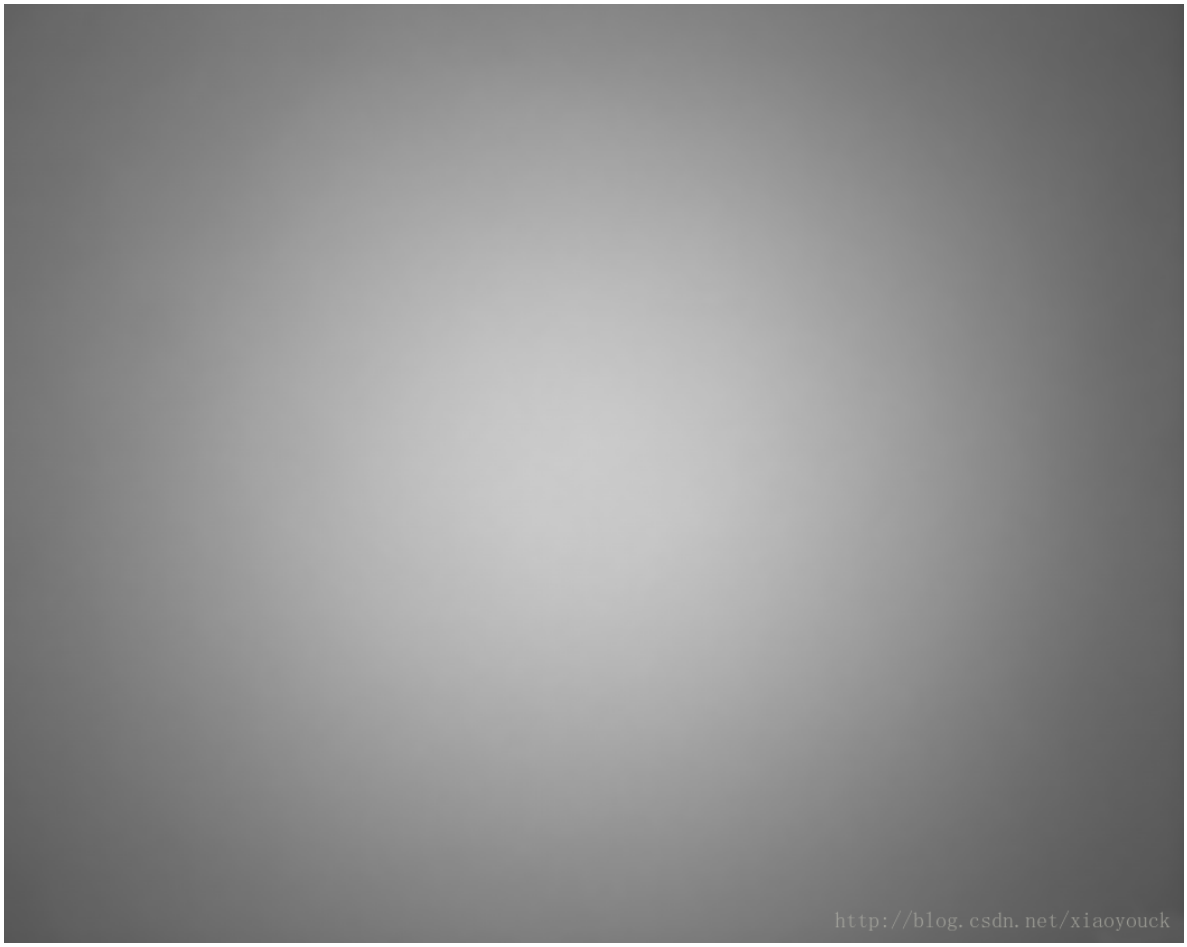
where R_offset, Gr_offset, Gb_offset, B_offset, α, $beta$ are configurable parameters.

Lens Shading Correction

For camera module, the sensor receives the image via lens. However, at the edge of lens, the light has a large angle with the optical axis of lens. The intensity of light degrades with the increase of distance from optical axis. Meanwhile, the refractive index of different colors are different. As a result, lens shading causes two shadings, luma shading and color shading.



Luma shading causes the intensity of image is not uniform. The corners of image are dark compared to the center of image.



Color shading causes the color distortion of image. This usually reflects the inconsistency of the color in image center and corners.

Lens shading correction aims to correct the luma and color distortion. The correction method is simple, which is to multiply a gain on each pixel at different locations. It is impossible to store all gains on all pixel locations. The easiest way is to do interpolation based on certain points.

There are several steps to get the parameters.

1. Take a image with uniform light condition. And divide the image into $m * n$ blocks. Four points of each block have a correction coefficient. Store the coefficients into look-up table.
2. Based on the pixel location, calculating the pixel falls into which block. Then get four coefficient of the block from LUT.
3. Calculating the correction gain of the pixel by interpolation.
4. Multiplying the correction gain by the pixel.

Usually step 1 is done by tuning software.

- [ ] Investigate several LSC algorithms
- [ ] Implement one of LSC algorithm

Anti-aliasing Noise Filter

- [ ]

HDR/Tone Mapping

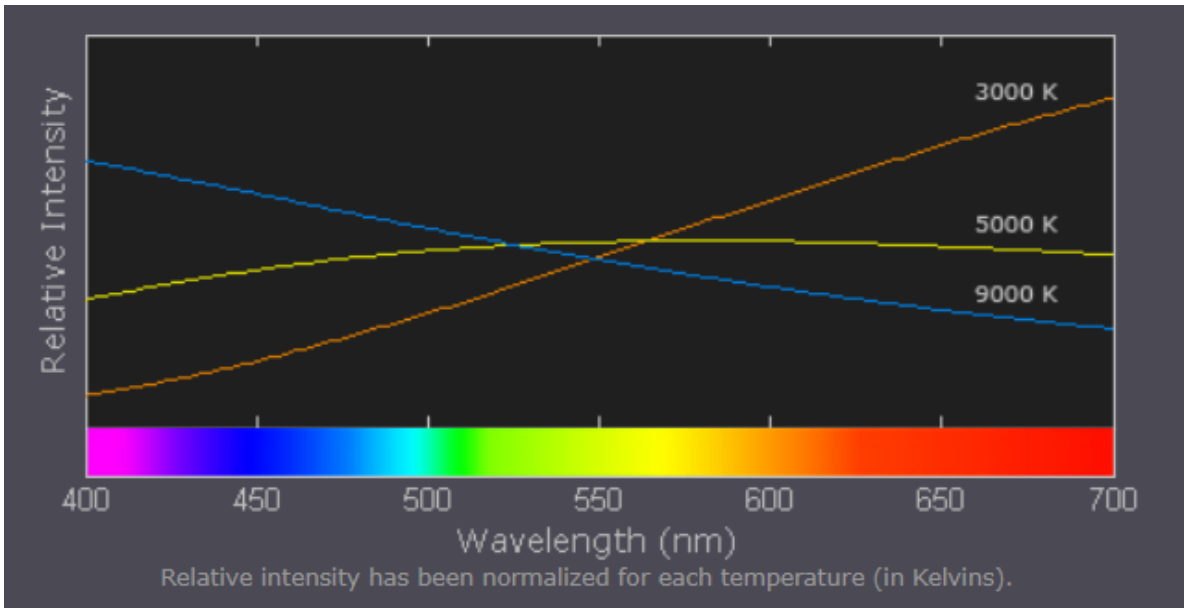
- [ ]

Auto White Balance Gain Control

White balance (WB) is the process of removing unrealistic color casts, so that objects which appear white in person are rendered white in your photo. Proper camera white balance has to take into account the "color temperature" of a light source, which refers to the relative warmth or coolness of white light. Our eyes are very good at judging what is white under different light sources, but digital cameras often have great difficulty with auto white balance (AWB) — and can

create unsightly blue, orange, or even green color casts. Understanding digital white balance can help you avoid these color casts, thereby improving your photos under a wider range of lighting conditions.

Color temperature describes the spectrum of light which is radiated from a "blackbody" with that surface temperature. A blackbody is an object which absorbs all incident light — neither reflecting it nor allowing it to pass through. A rough analogue of blackbody radiation in our day to day experience might be in heating a metal or stone: these are said to become "red hot" when they attain one temperature, and then "white hot" for even higher temperatures. Similarly, blackbodies at different temperatures also have varying color temperatures of "white light." Despite its name, light which may appear white does not necessarily contain an even distribution of colors across the visible spectrum:



Why is color temperature a useful description of light for photographers, if they never deal with true blackbodies? Fortunately, light sources such as daylight and tungsten bulbs closely mimic the distribution of light created by blackbodies, although others such as fluorescent and most commercial lighting depart from blackbodies significantly. Since photographers never use the term color temperature to refer to a true blackbody light source, the term is implied to be a "correlated color temperature" with a similarly colored blackbody. The following table is a rule-of-thumb guide to the correlated color temperature of some common light sources:

Color Temperature (K)	Light Source
1000 - 2000	Candlelight
2500-3500	Tungsten Bulb (household variety)
3000-4000	Sunrise/Sunset (clear sky)
4000-5000	Fluorescent Lamps
5000-5500	Electronic Flash
5000-6500	Daylight with Clear Sky (sun overhead)
6500-8000	Moderately Overcast Sky
9000-10000	Shade or Heavily Overcast Sky

To realize auto white balance function, it requires to estimate the color temperature. Under different color temperature, the gain differs and should be calculated based on the situation. The estimation of color temperature is complex and done by software or firmware. For hardware, only applying separate gain on different channels on bayer image. Usually the gain consists of white balance gain and exposure gain. The image statistics are acquired on bayer domain and calculated by firmware.

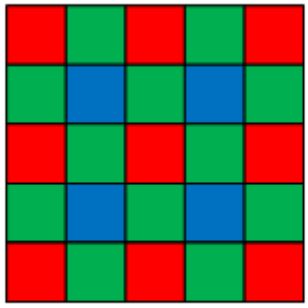
$$\begin{bmatrix} R' \\ Gr' \\ Gb' \\ B' \end{bmatrix} = \begin{bmatrix} R & 0 & 0 & 0 \\ 0 & Gr & 0 & 0 \\ 0 & 0 & Gb & 0 \\ 0 & 0 & 0 & B \end{bmatrix} * \begin{bmatrix} \frac{R_{avg}}{G_{avg}} * Gain \\ Gain \\ Gain \\ \frac{B_{avg}}{G_{avg}} * Gain \end{bmatrix} \tag{10}$$

Where $R_{avg}, G_{avg}, B_{avg}$ is the average of different channels on bayer domain. $Gain$ is the digital gain applied to all channels to keep the image brightness constant.

Color Filter Array Interpolation

A color image requires at least three color samples at each pixel location. Computer images often use red, green, and blue. A camera would need three separate sensors to completely measure the image. Using multiple sensors to detect different parts of the visible spectrum requires splitting the light entering the camera so that the scene is imaged onto each sensor. Precise registration is then required to align the three images. These additional requirements add a large expense to the system. Thus, many cameras use a single sensor array with a color filter array. The color filter array allows only one part of the spectrum to pass to the sensor so that only one color is measured at each pixel. This means that the camera must estimate the missing two color values at each pixel. This process is known as demosaicing.

Several patterns exist for the filter array. The most common array is the Bayer color filter array. The Bayer array measures the green image on a quincunx grid and the red and blue images on rectangular grids. The green image is measured at a higher sampling rate because the peak sensitivity of the human visual system lies in the medium wavelengths, corresponding to the green portion of the spectrum.



The simplest approach to demosaicing is bilinear interpolation, in which the three color planes are independently interpolated using symmetric bilinear interpolation from the nearest neighbors of the same color. As expected, bilinear interpolation generates significant artifacts, especially across edges and other high-frequency content, since it doesn't take into account the correlation among the RGB values. Practical demosaicing algorithms take such correlation into account, either with better linear filters [4], or with nonlinear filters that adapt interpolation smoothness to a measure of image activity or edginess.

Malvar et al proposed a simple linear demosaicing filter with better performance and low complexity than others. This demosaicing algorithm has been widely used in ISP and adapted in Matlab.

Malvar algorithm is designed based on bilinear interpolation filter. The bilinear demosaicing the green value $g(i,j)$ at a pixel location (i,j) that falls in a red or blue pixel, is computed by the average of the neighboring green values in a cross pattern.

$$\hat{g}(i,j) = \frac{1}{4}\sum g(i+m, j+n) \tag{11}$$

Where $(m,n) = \{(0,1),(0,-1),(-1,0),(1,0)\}$.

Specificially, to interpolate G values at a red location, a portion of estimated luminance change is added.

$$\hat{g}(i,j) = \hat{g}_B(i,j) + \alpha\Delta_R(i,j) \tag{12}$$

Where $\Delta_R(i,j)$ is the gradient of R at the location, computed by

$$\Delta_R(i,j) \triangleq r(i,j) - \frac{1}{4}\sum r(i+m, j+n) \tag{13}$$

Where $(m,n) = \{(0,2),(0,-2),(-2,0),(2,0)\}$.

Malvar demosaicing is a gradient-corrected bilinear interpolated approach, with a gain parameter to control how much correction is applied. For interpolating G at blue pixels, the same formula is used, but corrected by $\Delta_B(i,j)$.

For interpolating R at green pixels, $\Delta_G(i,j)$ is determined by a 9-point region.

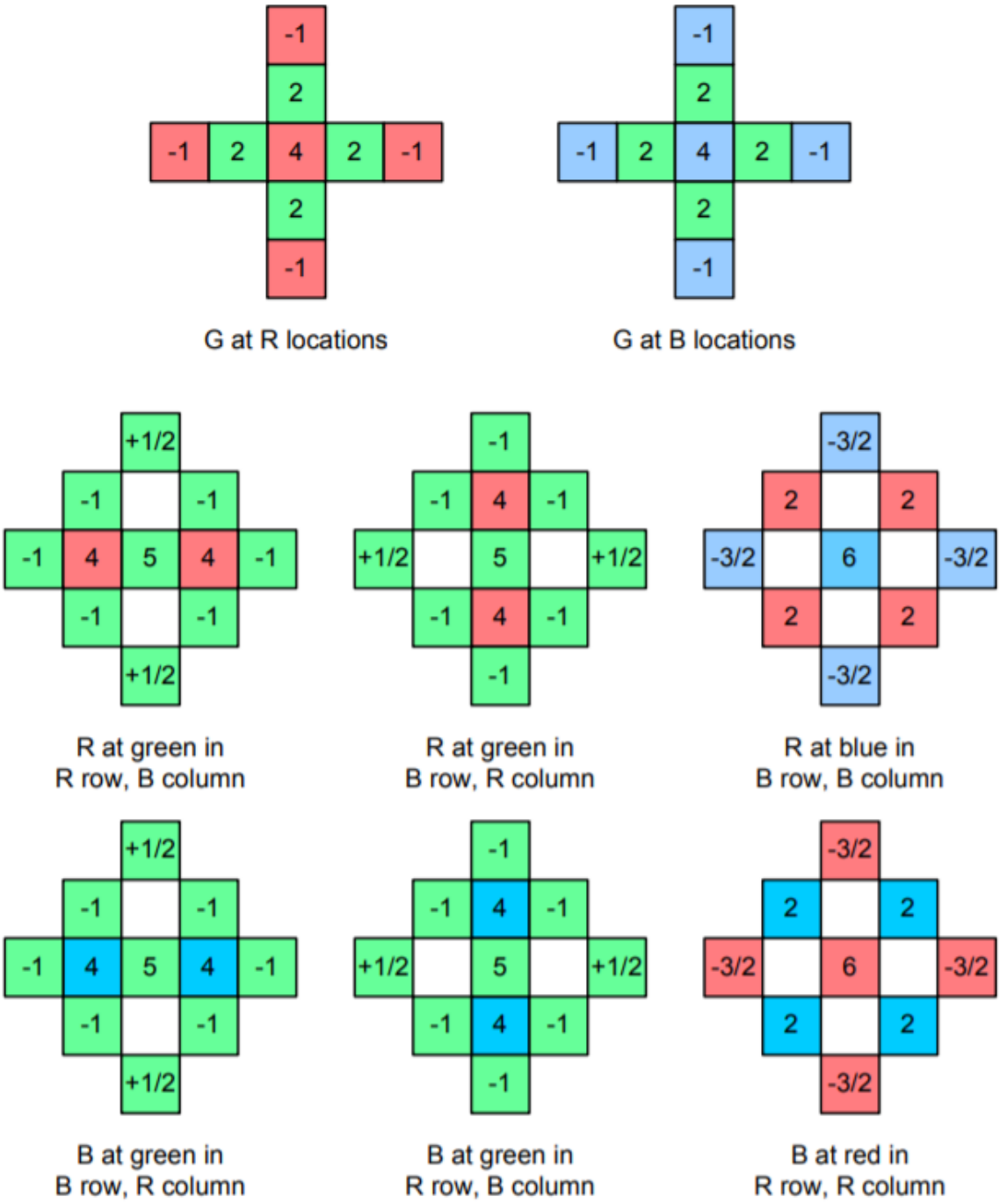
$$\hat{r}(i,j) = \hat{r}_B(i,j) + \beta\Delta_G(i,j) \tag{14}$$

For interpolating R at blue pixels, $\Delta_B(i,j)$ is determined by a 5-point region.

$$\hat{r}(i,j) = \hat{r}_B(i,j) + \gamma\Delta_B(i,j) \tag{15}$$

In order to determine the appropriate values for gain parameters $\{\alpha, \beta, \gamma\}$, Wiener approach is used. First, calculating the values leads to minimum mean-square error interpolation. Then approximating the optimal Wiener coefficients by integer multiples of small powers of $\frac{1}{2}$, with the final result $\alpha = \frac{1}{2}$, $\beta = \frac{5}{8}$ and $\gamma = \frac{3}{4}$.

The final filters are listed below.



RGB Domain Processing

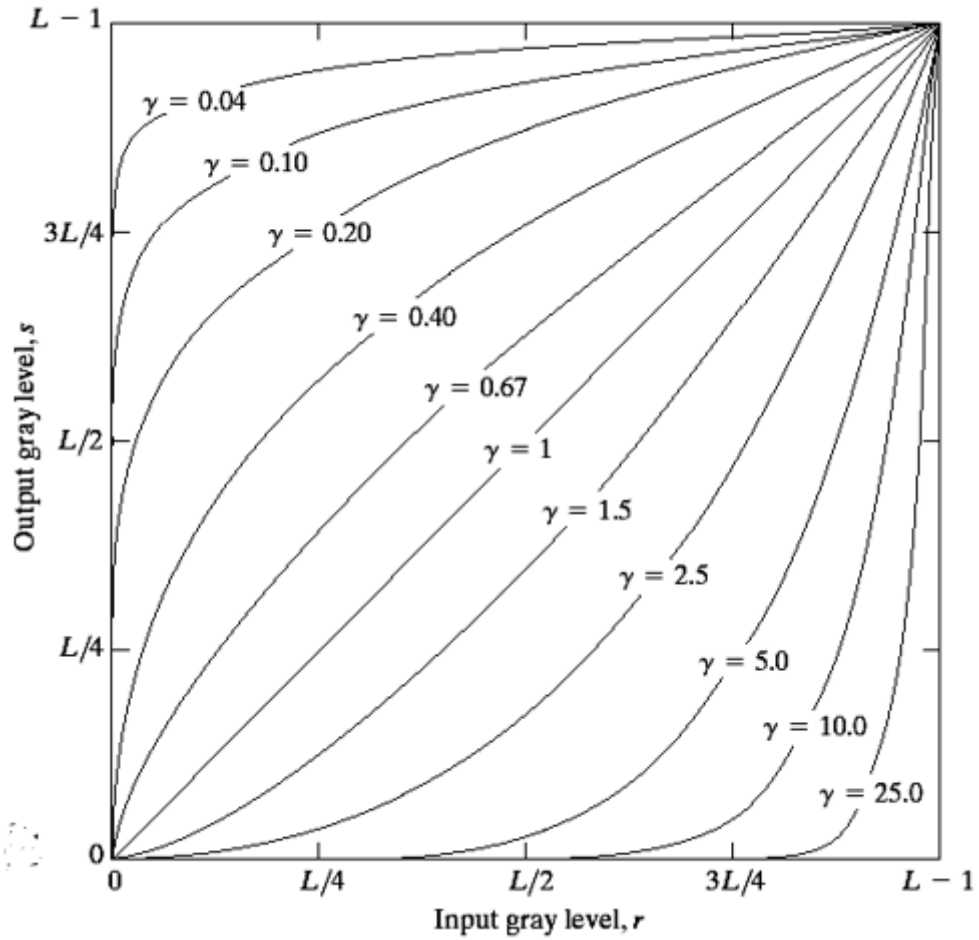
Gamma Correction

Gamma correction is a nonlinear operation used to encode and decode linear luminance or RGB values to match the non-linear characteristics of display devices.

Gamma correction helps to map data into a more perceptually uniform domain, so as to optimize perceptual performance of a limited signal range, such as a limited number of bits in each RGB component. Gamma correction is, in the simplest cases, defined by

$$V_{out} = AV_{in}^{\gamma} \tag{16}$$

Where A is a gain multiply to input. $\gamma < 1$ is called gamma compression and $\gamma > 1$ is called gamma expansion.



Nowadays, gamma correction is not only used as a method to match the nonlinear response of display devices, but a tool to adjust the input by mapping to nonlinear response to extend or compress the dynamic range of image.

Gamma correction is often realized in hardware by a look-up table, which is easy to change and program.

Color Correction Matrix

There are many variations that cause difficulties in accurately reproducing color in imaging systems. These can include:

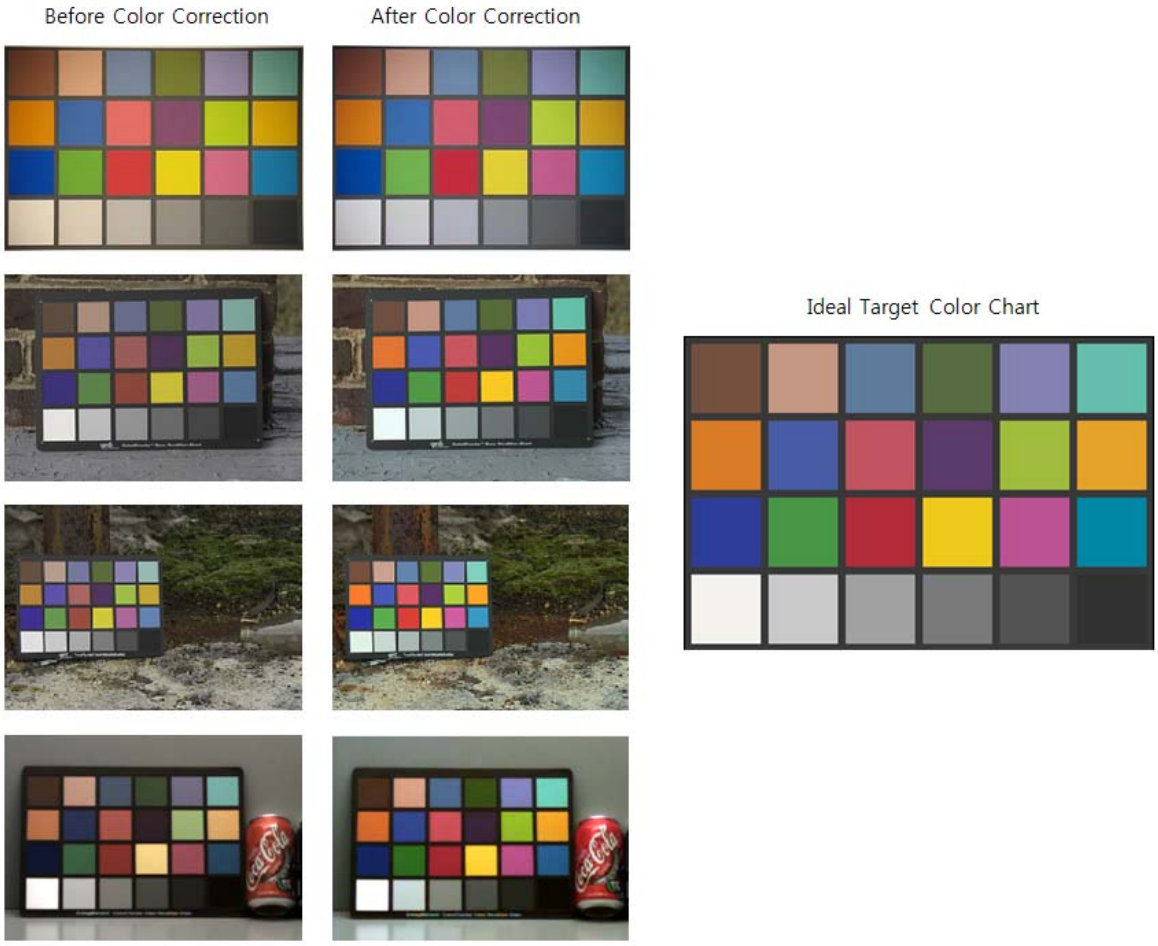
- Spectral characteristics of the optics (lens, filters)
- Lighting source variations like daylight, fluorescent, or tungsten
- Characteristics of the color filters of the sensor

Color correction is a process used in stage lighting, photography, television, cinematography, and other disciplines, which uses color gels, or filters, to alter the overall color of the light. Typically the light color is measured on a scale known as color temperature, as well as along a green–magenta axis orthogonal to the color temperature axis. Different color temperature results in different color correction matrix.

Color correction matrix is used to correct the differences between source imaging system and destination display or receiver system. For ISP system, the color space of sensor is different from the color space of display device or our human eyes. CCM is dedicated to transfer the sensor RGB color space to sRGB color space.

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} RR & RG & RB \\ GR & GG & GB \\ BR & BG & BB \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} R_{offset} \\ G_{offset} \\ B_{offset} \end{bmatrix} \tag{17}$$

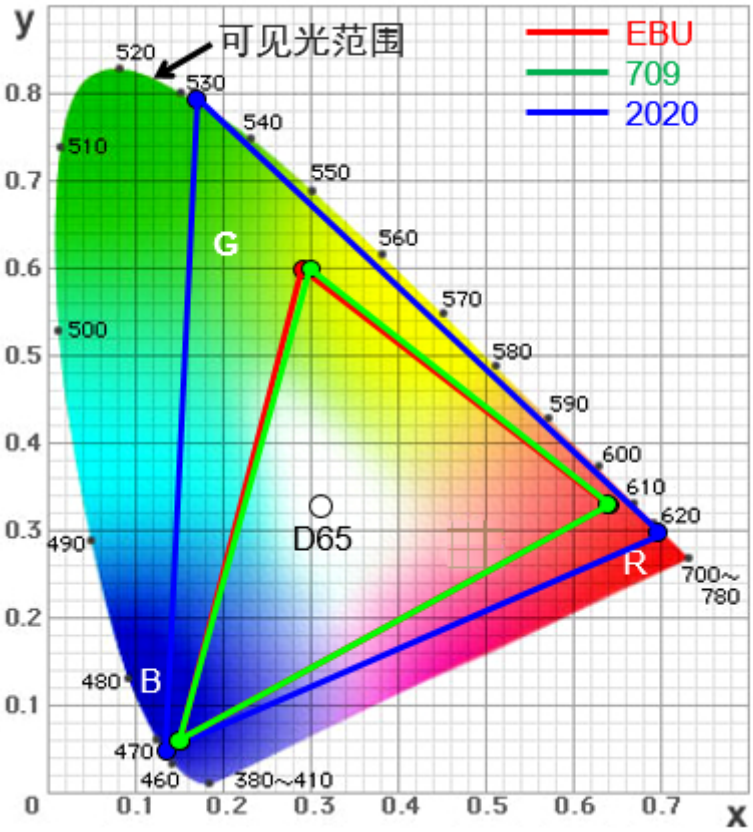
The images before color correction and after color correction are shown.



3D Lookup table is also used to map one color space to another. A 3D LUT is a 3D lattice of output RGB color values that can be indexed by sets of input RGB color values. Each axis of the lattice represents one of the three input color components and the input color thus defines a point inside the lattice. Since the point may not be on a lattice point, the lattice values must be interpolated; most products use trilinear interpolation. Cubes may be of various sizes and bit depths. Often 33×33×33 cubes are used as 3D LUTs. The most common practice is to use RGB 10-bit/component log images as the input to the 3D LUT. Output is usually RGB values that are to be placed unchanged into a display device's buffer.

Color Space Conversion

YCbCr is one of color spaces, widely used in modern color TV system. Usually RGB is transferred to luma (Y) and chroma (CbCr) via conversion matrix. The transmitter sends the encoded luma and chroma signal on one coaxial. The advantage of YCbCr color space is that the luma (Y) and chroma (CbCr) are separated. So that luma (Y) signal is compatible with color and black/white TVs.



There are three common color spaces, like BT.601 (for SDTV), BT.709 (for HDTV), BT.2020 (for UHDTV). Under different color space, the conversion matrix is different.

BT.601color space conversion

$$
\begin{aligned}
Y &= 0.2568 * R + 0.5041 * G + 0.0979 * B + 16 \\
Cb &= -0.1482 * R - 0.291 * G + 0.4392 * B + 128 \\
Cr &= 0.4392 * R - 0.3678 * G - 0.0714 * B + 128
\end{aligned} \tag{18}
$$

The fixed 8bit point color space conversion matrix is

$$
RGB2YCbCr = 256 * \begin{bmatrix} 0.2568 & 0.5041 & 0.0979 \\ -0.1482 & -0.2910 & 0.4392 \\ 0.4392 & -0.3678 & -0.0714 \end{bmatrix} = \begin{bmatrix} 66 & 129 & 25 \\ -38 & -74 & 112 \\ 112 & -94 & -18 \end{bmatrix} \tag{19}
$$

The fixed 8bit point RGB2YCrbCr conversion matrix is

$$
\begin{aligned}
Y &= 66 * R + 129 * G + 25 * B + 16 \\
Cb &= -38 * R - 74 * G + 112 * B + 128 \\
Cr &= 112 * R - 94 * G - 18 * B + 128
\end{aligned} \tag{20}
$$

The inverse conversion matrix is

$$
YCbCr2RGB = 256 * \begin{bmatrix} 1.1644 & -0.0001 & 1.5960 \\ 1.1644 & -0.3917 & -0.8130 \\ 1.1644 & 2.0173 & -0.0001 \end{bmatrix} = \begin{bmatrix} 298 & 0 & 409 \\ 298 & -100 & -208 \\ 112 & 516 & 0 \end{bmatrix} \tag{21}
$$

BT.709 color space conversion

$$\begin{aligned} Y &= 0.1826 * R + 0.6142 * G + 0.062 * B + 16 \\ Cb &= -0.1006 * R - 0.3386 * G + 0.4392 * B + 128 \\ Cr &= 0.4382 * R - 0.398 * G + 0.0402 * B + 128 \end{aligned} \tag{22}$$

The fixed 8bit point color space conversion matrix is

$$RGB2YCbCr = 256 * \begin{bmatrix} 0.1826 & 0.6142 & 0.062 \\ -0.1006 & -0.3386 & 0.4392 \\ 0.4382 & -0.398 & -0.0402 \end{bmatrix} = \begin{bmatrix} 47 & 157 & 16 \\ -26 & -87 & 112 \\ 112 & -102 & -10 \end{bmatrix} \tag{23}$$

The fixed 8bit point RGB2YCrbCr conversion matrix is

$$\begin{aligned} Y &= 47 * R + 157 * G + 16 * B + 16 \\ Cb &= -26 * R - 87 * G + 112 * B + 128 \\ Cr &= 112 * R - 102 * G - 10 * B + 128 \end{aligned} \tag{24}$$

The inverse conversion matrix is

$$YCbCr2RGB = 256 * \begin{bmatrix} 1.1644 & -0.0001 & 1.7969 \\ 1.1644 & -0.2133 & -0.5342 \\ 1.1644 & 2.1125 & -0.0002 \end{bmatrix} = \begin{bmatrix} 298 & 0 & 460 \\ 298 & -55 & -137 \\ 112 & 541 & 0 \end{bmatrix} \tag{25}$$

BT.2020 color space conversion

$$\begin{aligned} Y &= 0.2256 * R + 0.5832 * G + 0.0509 * B + 16 \\ Cb &= -0.1227 * R - 0.3166 * G + 0.4392 * B + 128 \\ Cr &= 0.4392 * R - 0.4039 * G + 0.0353 * B + 128 \end{aligned} \tag{26}$$

The fixed 8bit point color space conversion matrix is

$$RGB2YCbCr = 256 * \begin{bmatrix} 0.2256 & 0.5832 & 0.0509 \\ -0.1227 & -0.3166 & 0.4392 \\ 0.4392 & -0.4039 & -0.0353 \end{bmatrix} = \begin{bmatrix} 58 & 149 & 13 \\ -31 & -81 & 112 \\ 112 & -103 & -9 \end{bmatrix} \tag{27}$$

The fixed 8bit point RGB2YCrbCr conversion matrix is

$$\begin{aligned} Y &= 58 * R + 149 * G + 13 * B + 16 \\ Cb &= -31 * R - 81 * G + 112 * B + 128 \\ Cr &= 112 * R - 103 * G - 9 * B + 128 \end{aligned} \tag{28}$$

The inverse conversion matrix is

$$YCbCr2RGB = 256 * \begin{bmatrix} 1.1632 & 0.0002 & 1.6794 \\ 1.1632 & -0.1870 & -0.6497 \\ 1.1634 & 2.1421 & 0.0008 \end{bmatrix} = \begin{bmatrix} 298 & 0 & 430 \\ 298 & -48 & -166 \\ 298 & 548 & 0 \end{bmatrix} \tag{29}$$

YUV Domain Processing

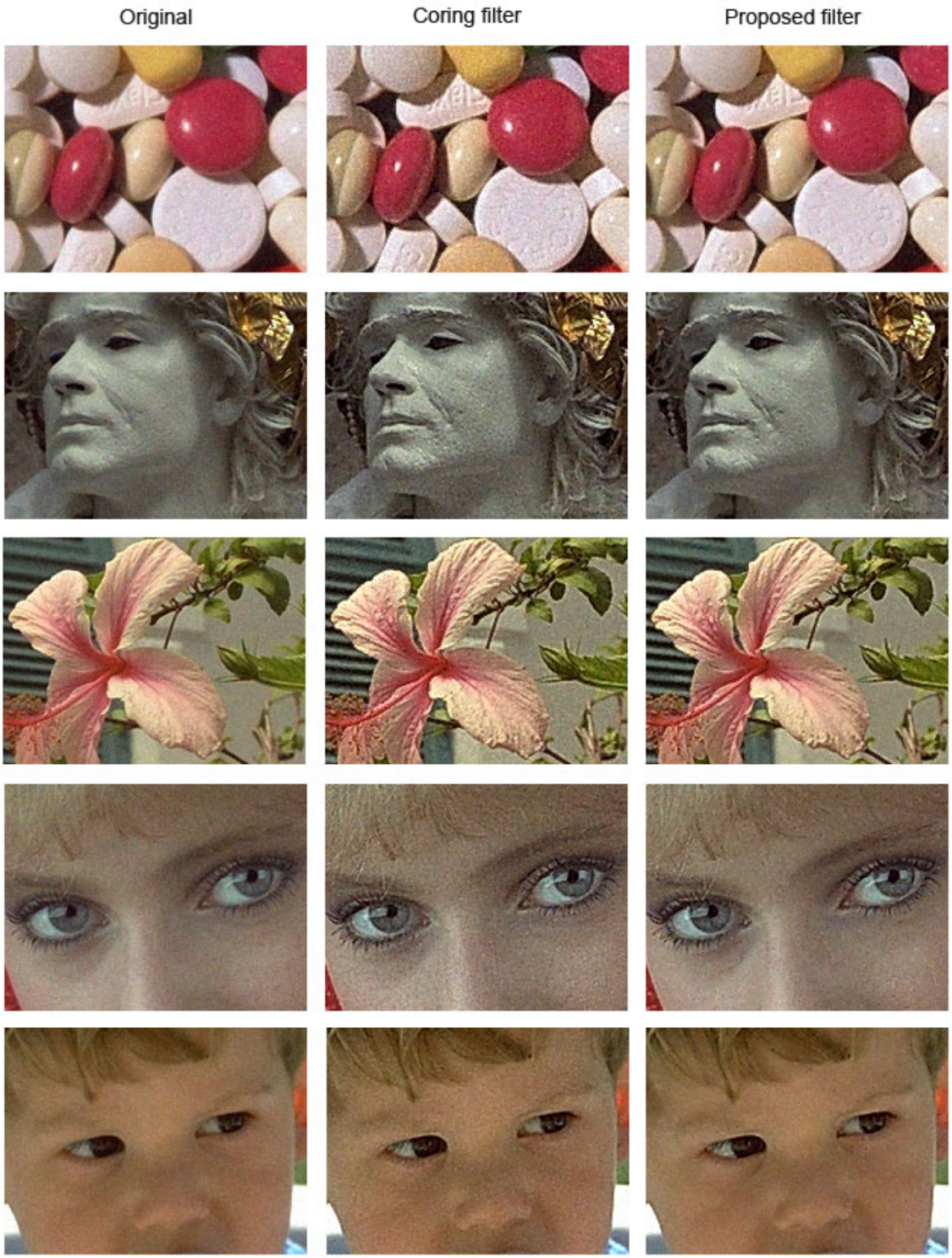
Noise Filter for Luma/Chroma

- [ ]

Edge Enhancement

Edge enhancement is an image processing filter that enhances the edge contrast of an image or video in an attempt to improve its acutance (apparent sharpness). The filter works by identifying sharp edge boundaries in the image, such as the edge between a subject and a background of a contrasting color, and increasing the image contrast in the area immediately around the edge. This has the effect of creating subtle bright and dark highlights on either side of any edges in the

image, called overshoot and undershoot, leading the edge to look more defined when viewed from a typical viewing distance.



Edge enhancement applied to an image can vary according to a number of properties; the most common algorithm is unsharp masking, which has the following parameters:

- *Gain*. This controls the extent to which contrast in the edge detected area is enhanced.
- *Radius* or *aperture*. This affects the size of the edges to be detected or enhanced, and the size of the area surrounding the edge that will be altered by the enhancement. A smaller radius will result in enhancement being applied only to sharper, finer edges, and the enhancement being confined to a smaller area around the edge.
- *Threshold*. Where available, this adjusts the sensitivity of the edge detection mechanism. A lower threshold results in more subtle boundaries of color being identified as edges. A

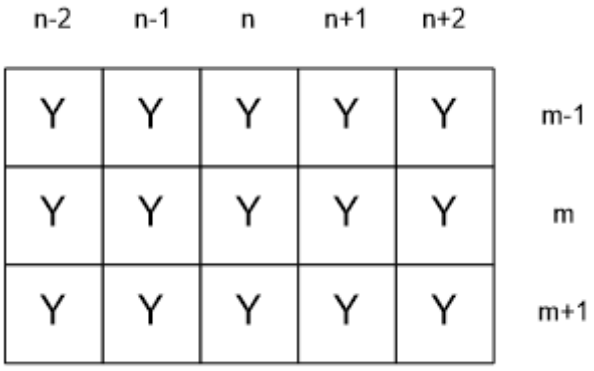
threshold that is too low may result in some small parts of surface textures, film grain or noise being incorrectly identified as being an edge.

In some cases, edge enhancement can be applied in the horizontal or vertical direction only, or to both directions in different gains.

Edge extraction is done on Y channel, to extract the edge map for image sharpening. The filter is selected as 3 x 5.

$$edge\_filter = \begin{bmatrix} -1 & 0 & -1 & 0 & -1 \\ -1 & 0 & 8 & 0 & -1 \\ -1 & 0 & -1 & 0 & -1 \end{bmatrix} * \frac{1}{8} \tag{30}$$

The filter is applied on 3 x 5 region.

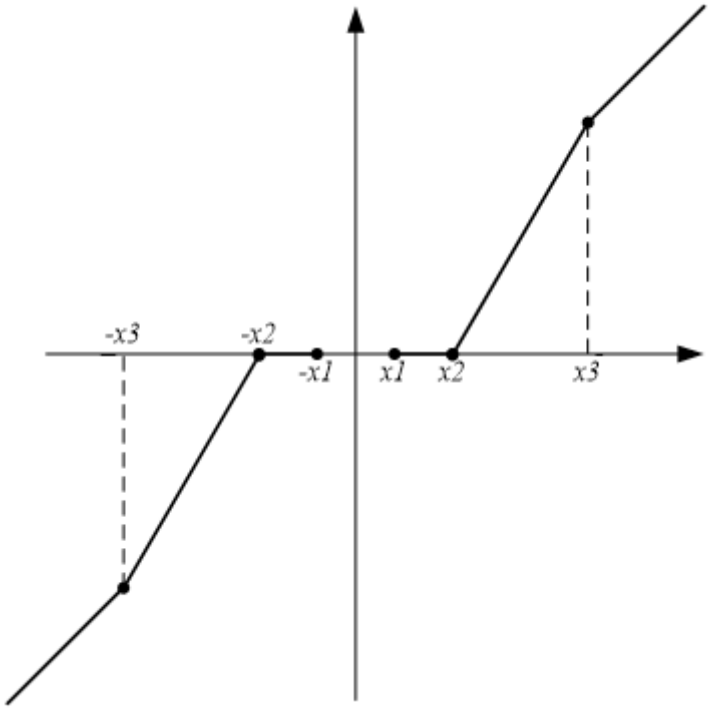


The extracted edge map is given by

$$EM = \frac{1}{8} \sum_{i,j=-2}^{i,j=2} Y_{m+i,n+j} * edge\_filter_{i,j} \tag{31}$$

Where i,j ranges from -2 to 2 with interval of 1.

The edge map (EM) is further modified through a lookup table (EMLUT) as



$$EMLUT(x) = \begin{cases} m_1 x & x \leq -x_3 \\ m_1 x_3 \times (\frac{x+x_2}{x_3-x_2}) & -x_3 < x \leq -x_2 \\ 0 & -x_2 < x \leq -x_1 \\ -m_2 x \ or \ Avg - Y & -x_1 < x \leq x_1 \\ 0 & x_1 < x \leq x_2 \\ m_1 x_3 \times (\frac{x-x_2}{x_3-x_2}) & x_2 < x \leq x_3 \\ m_1 x & x_3 \leq x \end{cases} \tag{32}$$

Where m_1, m_2 is the gain for different thresholds. When $-x_1 < x \leq x_1$, the pixel is noise not the edge. There are two methods to calculate the Y value:

$$Y = Y - m_2 x \tag{33}$$
$$Y = Avg \tag{34}$$

The Avg is calculated by the gradient of neighbor pixels, which is almost same in DPC. Four directions are calculated.

$$\begin{aligned} DDH &= abs(2Y_{m,n} - Y_{m,n-1} - Y_{m,n+1}) \\ DDV &= abs(2Y_{m,n} - Y_{m-1,n} - Y_{m+1,n}) \\ DDL &= abs(2Y_{m,n} - Y_{m-1,n-1} - Y_{m+1,n+1}) \\ DDR &= abs(2Y_{m,n} - Y_{m-1,n+1} - Y_{m+1,n-1}) \end{aligned} \tag{35}$$

Find the minimum of (DDH, DDV, DDL, DDR), then Avg takes the value of that direction.

$$\begin{aligned} AvgH &= \frac{Y_{m,n} + Y_{m,n-1} + Y_{m,n+1}}{3} \\ AvgV &= \frac{Y_{m,n} + Y_{m-1,n} + Y_{m+1,n}}{3} \\ AvgL &= \frac{Y_{m,n} + Y_{m-1,n-1} + Y_{m+1,n+1}}{3} \\ AvgR &= \frac{Y_{m,n} + Y_{m-1,n+1} + Y_{m+1,n-1}}{3} \end{aligned} \tag{36}$$

The $EMLUT(x)$ is further clipped before adding to the original Y channel.

$$Y' = Y + clip(EMLUT(x)) \tag{37}$$

To avoid over-shoot on edges, the Y' is further clipped for output.

$$Y' = clip(Y', Y_{max}, Y_{min}) \tag{38}$$

False Color Suppression

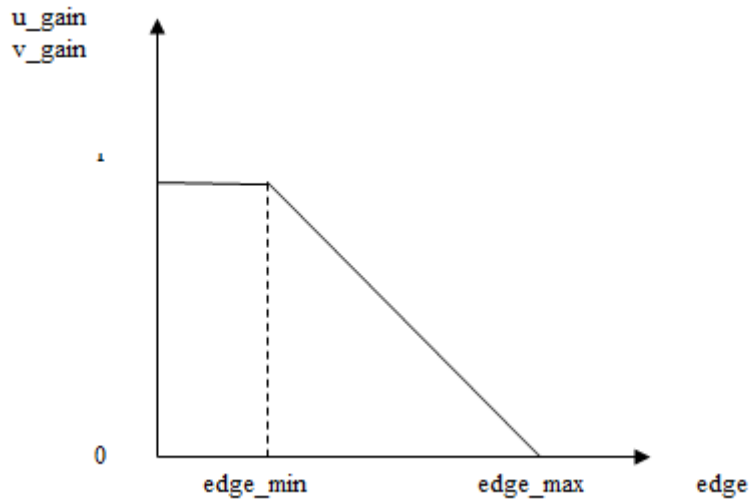
False colors (speckles) could be introduced during the demosaicing phase where very fine detail is resolved. False color suppression is similar to color smoothing. The luminance channel is not affected by this suppression. False colors are generally more apparent in images from cameras without anti-aliasing filters. Nowadays, despite the false color introduced by demosaicing, other factors also leads to false color like lens, optical filter, dynamic compression etc. False color suppression is used to reduce or eliminate the false color introduced during sensor imaging or isp pipeline. Another function of false color suppression is used to control the chroma saturation level while doing sharpening and chroma noise under different light condition, like low light or dark light condition.

False color suppression is conducted on chroma (Cb, Cr) channel. Luma (Y) is not influenced. The false color suppression is also based on the piece-wise linear function.

The edge map acquired from Edge Enhancement is taken by absolute value.

$$EM' = abs(EM) \tag{39}$$

Then clipping the absolute edge map into $\{edge\_min, edge\_max\}$.



$$EM'' = clip(edge\_min, edge\_max) \tag{40}$$

The chroma gain is calculated by

$$chroma\_gain = K\_edge * (edge\_max - EM'') \tag{41}$$

Where $K\_edge = \frac{65536}{(edge\_max - edge\_min)}$.

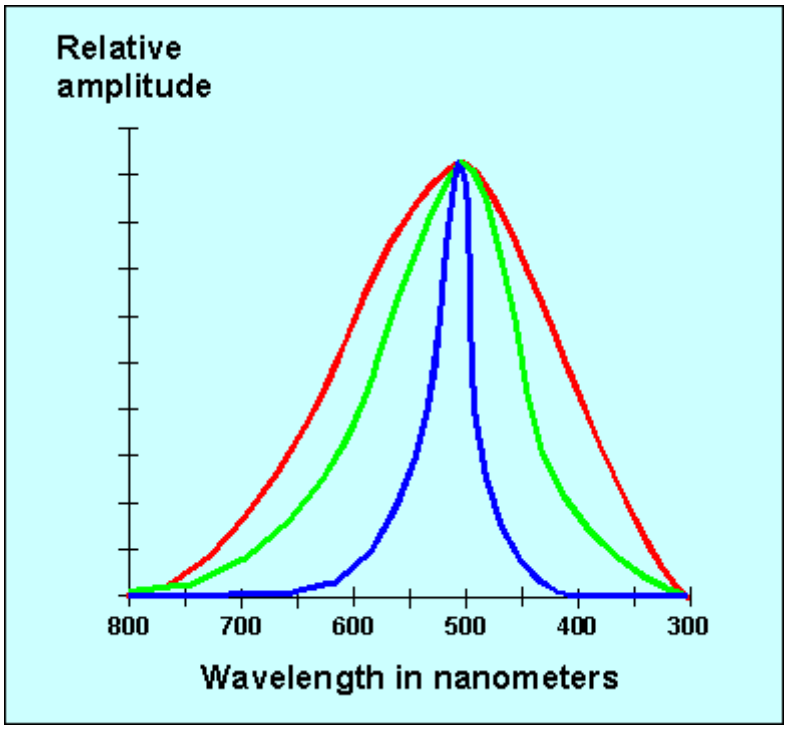
Once the gain is computed, then doing false color suppression on chroma channel when edge is larger than $edge\_min$.

$$(Cb', Cr') = \begin{cases} (Cb, Cr) & edge \le edge\_min \\ \frac{chroma\_gain * (EM'' - 128))}{65536} + 128 & edge\_min < edge \le edge\_max \\ (0, 0) & edge\_max < edge \end{cases} \tag{42}$$

Hue/Saturation Control

Hue, saturation, and brightness are aspects of color in the RGB scheme. Hue is one of the main properties (called color appearance parameters) of a color, defined technically , as "the degree to which a stimulus can be described as similar to or different from stimuli that are described as red, green, blue, and yellow" (which in certain theories of color vision are called unique hues). Hue can typically be represented quantitatively by a single number, often corresponding to an angular position around a central or neutral point or axis on a colorspace coordinate diagram (such as a chromaticity diagram) or color wheel, or by its dominant wavelength or that of its complementary color. Most sources of visible light contain energy over a band of wavelengths. Hue is the wavelength within the visible-light spectrum at which the energy output from a source is greatest. This is shown as the peak of the curves in the accompanying graph of intensity versus wavelength. The saturation of a color is determined by a combination of light intensity and how much it is distributed across the spectrum of different wavelengths. The saturation is represented by the steepness of the slopes of the curves. Here, the red curve represents a color having low saturation, the green curve represents a color having greater saturation, and the blue curve represents a color with fairly high saturation. As saturation increases, colors appear more "pure." As saturation decreases, colors appear more "washed-out". Saturation is the purity or

vividness of a color, expressed as the absence of white. A color with 100% saturation contains no white. A color with 0% saturation corresponds to a shade of gray.



Hue and saturation adjustment are both applied on chroma channel. Hue adjustment involves the change of hue angle, which is implemented by cos, sin function.

$$\begin{aligned} Cb' &= (Cb - 128)cos\theta + (Cr - 128)sin\theta + 128 \\ Cr' &= (Cr - 128)cos\theta - (Cb - 128)sin\theta + 128 \end{aligned} \tag{43}$$

Where cos, sin is realized by lookup table, computed by software and programmed into memory.

$$(Cb'', Cr'') = clip((Cb', Cr'), 0, 255) \tag{44}$$

The final (Cb, Cr) are clipped to $[0, 255]$.

Saturation adjustment is easy to implement.

$$(Cb', Cr') = k * ((Cb, Cr) - 128) + 128 \tag{45}$$

Where k is the saturation ratio.

Brightness/Contrast Control

Brightness is a relative expression of the intensity of the energy output of a visible light source. It can be expressed as a total energy value (different for each of the curves in the diagram), or as the amplitude at the wavelength where the intensity is greatest (identical for all three curves). In the RGB color model, the amplitudes of red, green, and blue for a particular color can each range from 0 to 100 percent of full brilliance. These levels are represented by the range of decimal numbers from 0 to 255, or hexadecimal numbers from 00 to FF. brightness is the perception elicited by the luminance of a visual target. It is not necessarily proportional to luminance. This is a subjective attribute/property of an object being observed and one of the color appearance parameters of color appearance models. Brightness refers to an absolute term and should not be confused with Lightness. Contrast in visual perception is the difference in appearance of two or more parts of a field seen simultaneously or successively (hence: brightness contrast, lightness contrast, color contrast, simultaneous contrast, successive contrast, etc.). The contrast is defined as the ratio of the luminance of the brightest color (white) to that of the darkest color (black) that

the system is capable of producing. A high contrast ratio is a desired aspect of any display. It has similarities with dynamic range.

Brightness and contrast are both applied on luma channel.

$$Y' = Y + brightness \tag{46}$$

Where $brightness$ ranges from $[-128, 127]$.

$$Y' = Y + (Y - 127) * contrast \tag{47}$$

Where $contrast$ is the adjustment ratio, ranges from $[0, 1.992x]$.

ISP Tuning Algorithm

Lens Shading Correction

Color Correction Matrix

Auto Exposure/Auto White Balance

ISP Tuning Tool

Load RAW/RGB/YUV Images (.raw/.rgb888/.yuv444)

Load configuration files (.csv)

Individually run each algorithm

Sequentially execute all algorithm

References

1. http://www.babelcolor.com/index_htm_files/AN-9a%20How%20to%20derive%20and%20use%20a%20Color%20Correction%20Matrix.pdf
2. https://blog.csdn.net/u014470361/article/details/88692390
3. https://zhuanlan.zhihu.com/p/34562544
4. https://en.wikipedia.org/wiki/Color_space
5. https://zh.wikipedia.org/wiki/YCbCr
6. https://en.wikipedia.org/wiki/Edge_enhancement
7. http://www.asicfpga.com/site_upgrade/asicfpga/isp/edge_enhancement1.html
8. https://rawpedia.rawtherapee.com/Demosaicing
9. http://help.corel.com/paintshop-pro/v20/main/en/documentation/index.html#page/Corel_PaintShop_Pro/Adjusting_hue_and_saturation.html
10. https://whatis.techtarget.com/definition/hue-saturation-and-brightness
11. https://en.wikipedia.org/wiki/Display_contrast
12. https://en.wikipedia.org/wiki/Brightness