Infinite-ISP Algorithm Design Model v1.1

Description of Algorithms

11th September 2023

About this Document:

Purpose

This document is a supplement provided with the Infinite-ISP Algorithm Design Model, it provides algorithm details of all modules in the ISP as well as the explanation of configuration parameters.

Revision History

Version	Released Date	Change Description
1.00	2022-12-15	Initial Draft – Algorithm Development Model V1
1.1	-	Added Sharpening Module Optimized JBF Algorithm in BNR Module Optimized Algorithm for DPC Module

Contents

List of Figures

List of Tables

Introduction

Infinite-ISP Algorithm Design Model (aka Model) is a collection of 19 Python modules which convert an input RAW image from a sensor to an output YUV image. All the modules are individually configurable via various parameters. Some of the module's configurable parameters are determined by using a separate software tuning tool while others are more directly determined. The model pipeline is shown in Figure 1.

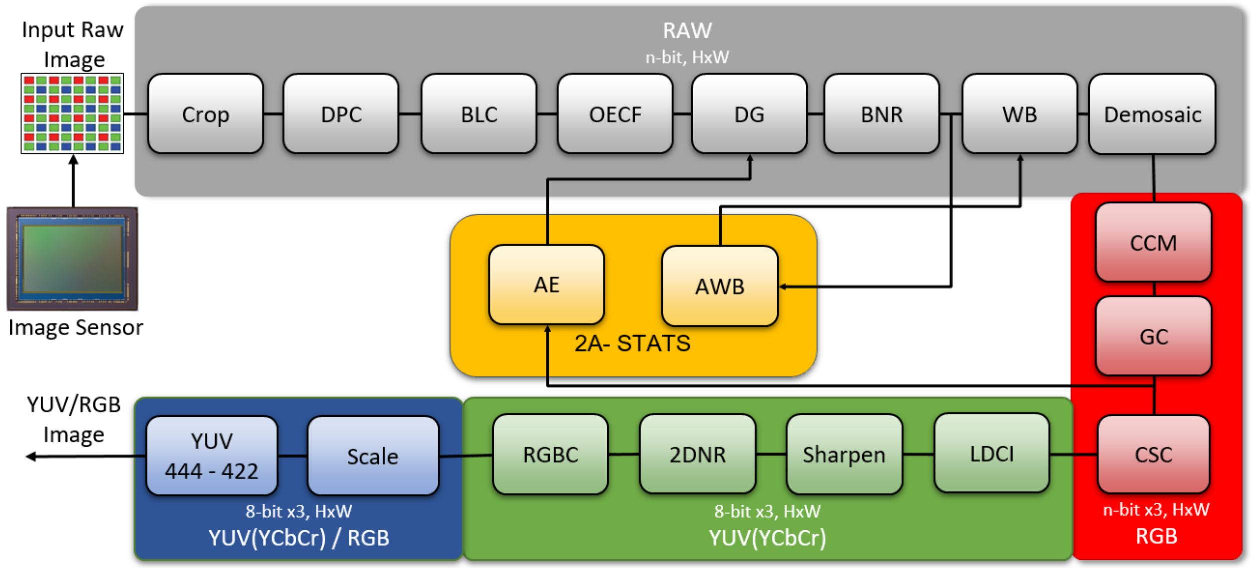


Figure 1: Infinite-ISP Algorithm Design Model pipeline

Motivation

The real motivation behind Infinite-ISP is to streamline procedures, enhance current approaches, and develop advance algorithms in the ISP and computational photography field. This project provides a platform for relevant developers to work on the most exciting and challenging hardware-related issues that an ISP engineer encounters. The "open innovation" approach, enables anyone with the right skills, time, and interest to contribute generously to the Infinite-ISP project.

Modules Description

Currently the model consists of the following 18 modules,

- Crop
- Dead Pixel Correction (DPC)
- Black Level Compensation (BLC)
- Optoelectronic Conversion Function (OECF)
- Digital Gain (DG)
- Bayer Noise Reduction (BNR)
- White Balance (WB)
- Auto White Balance (AWB)
- Demosaic
- Color Correction Matrix (CCM)
- Gamma Correction (GC)
- Auto Exposure (AE)
- Color Space Conversion (CSC)
- Local Dynamic Contrast Improvement (LDCI)
- Sharpening
- 2d Noise Reduction (2DNR)
- RGB Conversion (RGBC)
- Scale
- YUV Conversion Format (YUV 444 - 422)

Some functions like Local Lens Shading Correction (LSC), High Dynamic Range Imaging HDR Stitching, Tone Mapping (TM), Dynamic Contrast Improvement (LDCI) and Sharpening or Edge Enhancement (EE) will be added to the design in the future.

Crop

The crop module is used to extract a rectangular region from an image by selecting an area at the center of the image with a specific size. This is useful for focusing on a specific portion of the image or more importantly removing unwanted regions from the edges.

Algorithm Explanation

A straightforward cropping algorithm has been implemented, ensuring the preservation of the Bayer pattern. This is achieved by verifying that the number of rows and columns to be cropped are divisible by 4, and subsequently cropping an equal number of pixels from all sides of the image.

Cropping is applied on image only when both of the following conditions are met:

- Parameters [$\boldsymbol{new\_height}$, $\boldsymbol{new\_width}$] provided to the crop module are even numbers
- The difference between image dimensions and crop parameters is divisible by 4:

$$(image\_height - new\_height)\ mod\ 4 = 0$$

$$(image\_width - new\_width)\ mod\ 4 = 0$$

Configuration Parameters

Parameters	Details
$\boldsymbol{is\_enable}$	When enabled it crops the image if the conditions listed above are met: False: Disable True: Enable
$\boldsymbol{new\_width}$	New width of the input RAW image after cropping
$\boldsymbol{new\_height}$	New height of the input RAW image after cropping

Table 1: Crop configuration parameters

Dead Pixel Correction

Image sensors sometimes come with manufacturing flaws that result in some malfunctioning pixels in the image. Despite advanced manufacturing techniques photo sensor arrays like CCD and CMOS sensor arrays include defective pixels because of noise, dust, or fabrication flaws. Hence identifying the defective pixels in the images and then replacing them with approximate values are two important phases of the DPC module.

Defective pixels can be categorized into two groups:

1. Dead: consistently low output
2. Hot: consistently high output

.

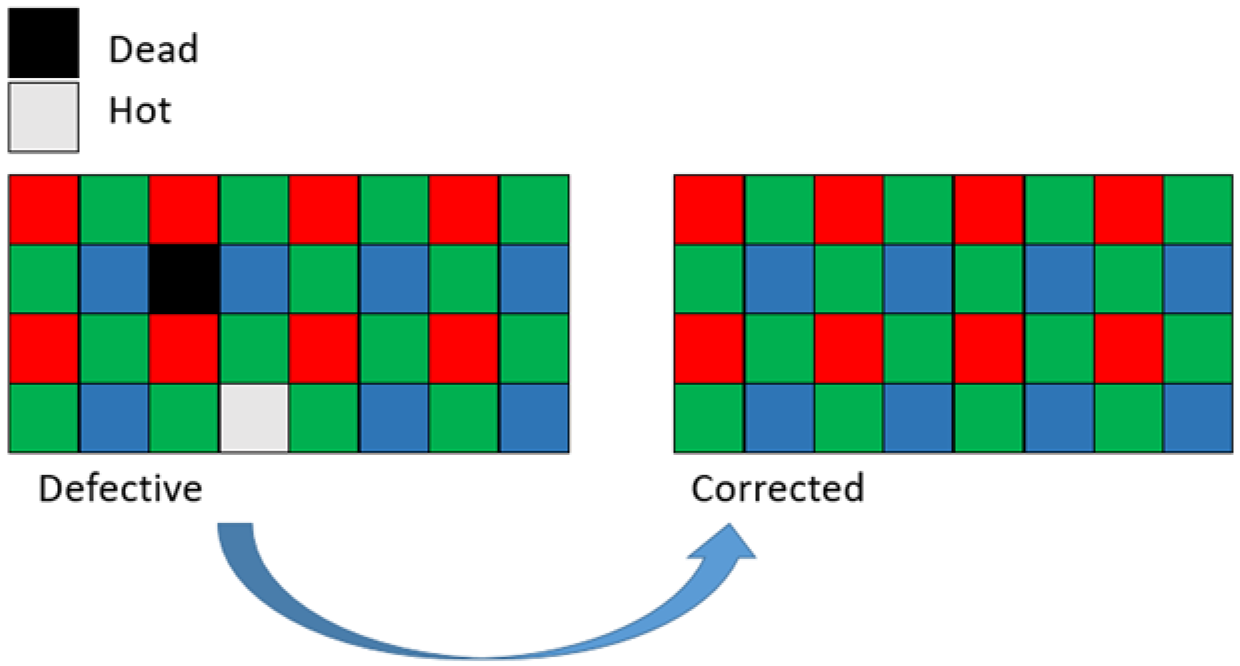


Figure 2: Defective and corrected pixels

Dynamic DPC Approach

Each pixel of the image is tested by the DPC algorithm. To find defective pixels a 3×3 neighborhood of the *same* color channel surrounding the pixel being tested in a 5×5 window is created. The pixel is considered defective if it meets both of the following conditions:

1. The pixel value is less than the minimum pixel value of the all the pixels in the 3×3 neighborhood or greater than the maximum pixel value of all the pixels in the 3×3 neighborhood.

2. The difference between the pixel's value and each one of the pixels in the 3 x 3 neighborhood is greater than the programmed threshold parameter.

Once the pixel is identified as defective, its value is corrected by first computing four gradients (horizontal, vertical, left diagonal, and right diagonal) which pass through the pixel. The corrected value is then computed as the average of the two neighbors in the *minimum* gradient direction.

[Add optimization details for V1.1]

Configuration Parameters

Parameters	Details
is_enable	When enabled, apply the DPC algorithm: False: Disable True: Enable
dp_threshold	The threshold used for determining defective pixels. A low threshold results in more pixels being detected as dead and hence corrected

Table 2: DPC configuration parameters

Black Level Correction

This module is responsible for setting the image's pure black color. Setting the exposure time and various programmable gains (analog and digital) to their minimum values is one of the techniques to create a pure black image; however, even under these conditions, the resulting image may not be entirely black and as such the black level needs to be corrected. The four offset parameters, one for each channel, needed by this algorithm are generated by using the Tuning tool for a specific image sensor. Once the parameters have been generated, they are used by this module to correct the image’s black level.

Algorithm Explanation

To apply the black level correction, subtract the corresponding parameters from each of the pixels in the image. For a RGGB Bayer image the new values are computed as follows:

$$R' = R - r\_offset$$

$$Gr' = Gr - gr\_offset$$

$$Gb' = Gb - gb\_offset$$

$$B' = B - b\_offset$$

Additionally, by applying these offsets the range of values that a pixel generates is scaled down. This can optionally be corrected (enabled via the $\boldsymbol{is\_linear}$ parameter) by applying a linearization function over the desired range of values. Black level correction with linearization is computed as follows:

$$R' = \frac{R - r\_offset}{r\_sat - r\_offset} 2^n$$

$$Gr' = \frac{Gr - gr\_offset}{gr\_sat - gr\_offset} 2^n$$

$$Gb' = \frac{Gb - gb\_offset}{gb\_sat - gb\_offset} 2^n$$

$$B' = \frac{B - b\_offset}{b\_sat - b\_offset} 2^n$$

Configuration Parameters

Parameters	Details
$\boldsymbol{is\_enable}$	When enabled, apply the BLC algorithm: False: Disable True: Enable

Parameters	Details
r_offset	Red channel offset
gr_offset	Gr channel offset
gb_offset	Gb channel offset
b_offset	Blue channel offset
is_linear	Enables or disables linearization. When enabled a linearization function is applied to scale the values over the desired range of values
r_sat	Red channel saturation level
gr_sat	Gr channel saturation level
gb_sat	Gb channel saturation level
b_sat	Blue channel saturation level

Table 3: BLC configuration parameters

Opto-Electronic Conversion Function—OECF

The relationship between the sensor output (photocell voltages) and incident light lux is called the optoelectronic conversion function (OECF). The sensor usually has an approximately linear response in that the detected signal is proportional to the incident light lux as imaged by the lens. Linearity is essential for achieving good white balance and color accuracy. The OECF module implements lookup curves for voltage re-mapping for a *specific* sensor. There are four lookup curves, one for each color channel which are generated through the Tuning tool.

Algorithm Explanation

The Tuning tool uses ISO 14524 to generate the four curves. It uses a digital reflective camera contrast chart to plot density response vs. pixel voltage values.

Configuration Parameters

Parameters	Details
is_enable	When enabled, applies the OECF curve: False: Disable True: Enable
r_lut	The lookup table for the OECF curve. This curve is sensor dependent and is calculated by the Tuning tool

Table 4: OECF configuration parameters

Bayer Noise Reduction

The Bayer Noise Reduction (BNR) block suppresses noise in the Bayer domain before demosaicing. As a noise reduction block in the ISP pipeline, it is primarily tasked with reducing noise without effectively blurring the image details.

Joint Bilateral Filter (JBF)

For the BNR block, a Joint/Cross Bilateral Filter (JBF) has been selected as the appropriate candidate for RAW image denoising. The algorithm has been optimized through the adoption of the shifted array approach. This method is more efficient than the traditional technique of iterating over each pixel individually using loops. By leveraging array shifts, computational overhead is reduced, leading to faster processing times.

1. Green channel interpolation is performed on the input Bayer image to obtain a complete image of the green pixels using green interpolation kernels derived from the Malwar-He-Cutler algorithm.
2. Input Bayer image is deconstructed into R and B sub-images of half the size compared to the original Bayer image.
3. Interpolated G image based on a window (size determined by the file window parameter) is deconstructed into G pixels just at the R pixel locations. Additionally, the interpolated G image is also deconstructed into G pixels just at the B pixel locations. This produces R and B guided images.
4. JBF is applied on the R and B sub-images (produced in step 2) using their respective guide images (produced in step 3). JBF is also applied on the interpolated G image using itself as the guide image.
5. JBF uses two types of kernels i.e. range and spatial. A look-up table weighing scheme is created for the range kernel to facilitate the algorithm's hardware implementation.
6. The output Bayer image is reconstructed by joining the R, G, and B pixels from the JBF outputs (step 4).

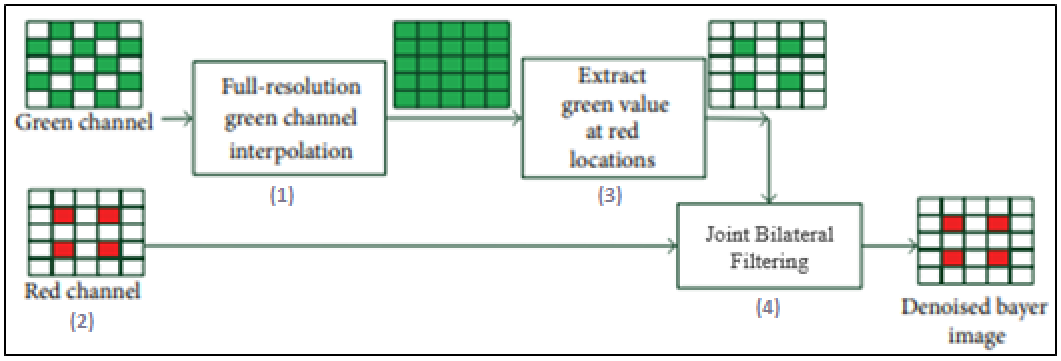

Green channel
Full-resolution green channel interpolation
(1)
Extract green value at red locations
(3)
Red channel
(2)
Joint Bilateral Filtering
(4)
Denoised bayer image

Figure 3: BNR algorithm

Configuration Parameters

Parameters	Details
is_enable	When enabled, applies the BNR algorithm: False: Disable True: Enable
filt_window	Filter size for BNR This should be an odd window size e.g., 3 x 3, 5x5, etc.
r_std_dev_s	Red channel Gaussian kernel strength Higher strength values will result in increased blurring while a value of 0 is not permissible
r_std_dev_r	Blue channel range kernel strength Higher strength values will result in increased blurring while a value of 0 is not permissible
g_std_dev_s	Gr and Gb Gaussian kernel strength. Higher strength values will result in increased blurring while a value of 0 is not permissible
g_std_dev_r	Gr and Gb range kernel strength Higher strength values will result in increased blurring while a value of 0 is not permissible
b_std_dev_s	Blue channel Gaussian kernel strength Higher strength values will result in increased blurring while a value of 0 is not permissible
b_std_dev_r	Blue channel range kernel strength Higher strength values will result in increased blurring while a value of 0 is not permissible

Table 5: BNR configuration parameters

Digital Gain

Digital gain works with the auto exposure module to choose an appropriate constant value to be multiplied to all the channels to improve the image exposure.

Algorithm Explanation

In the digital gain module, the raw image I is multiplied by an integer g, also known as Gain.

$$I' = I * g$$

$$g = gain\_array[current\_gain]$$

An array of permissible gains, $\boldsymbol{gain\_array}$, is defined in the configuration, and the index of the current gain is stored in the $\boldsymbol{current\_gain}$ parameter, with its default value being zero. Gain selection for the Digital Gain module also depends on the $\boldsymbol{ae\_feedback}$ parameter:

- $\boldsymbol{ae\_feedback = 0}$ means image exposure is fine, so no change in gain is made.
- $\boldsymbol{ae\_feedback = 1}$ means the image is overexposed, and the current gain is updated accordingly to decrease the gain.

$$current\_gain = \begin{cases} current\_gain, & current\_gain \ = 0 \\ current\_gain - 1, & otherwise \end{cases}$$

- $\boldsymbol{ae\_feedback = -1}$ means the image is underexposed, and the current gain is updated accordingly to increase the gain.

$$current\_gain = \begin{cases} current\_gain, & current\_gain \ = size(gain\_array) \\ current\_gain + 1, & otherwise \end{cases}$$

Configuration Parameters

Parameters	Details
$\boldsymbol{is\_auto}$	Flag to adjust digital gain according to AE Feedback False : $\boldsymbol{current\_gain}$ can only be changed manually True : $\boldsymbol{current\_gain}$ is adjusted according to $\boldsymbol{ae\_feedback}$
$\boldsymbol{gain\_array}$	List of permissible gains
$\boldsymbol{current\_gain}$	Index of the $\boldsymbol{gain\_array}$ for the current gain which is being used. The index value starts at 0
$\boldsymbol{ae\_feedback}$	AE feedback parameter 0 : Correct Exposure 1 : Overexposed -1 : Underexposed

Table 6: DG configuration parameters

White Balance

This module adjusts the white balance of raw images by applying appropriate gains, ensuring that white colors appear true to life. If gains are not available, they must first be determined through an auto-white balance process.

Algorithm Explanation:

The white balance module selectively applies the provided gains to the red and blue channels, while the green channel remains unaffected. This process effectively restores color balance, making white colors appear accurate and natural.

$$R' = R * r\_gain$$

$$B' = B * b\_gain$$

Configuration Parameters

Parameters	Details
$\boldsymbol{is\_enable}$	Applies user-given white balance gains when enabled: False: Disable True: Enable
$\boldsymbol{is\_auto}$	Flag to adjust white balance gains according to AWB Algorithm 0 : $\boldsymbol{r\_gain}$ and $\boldsymbol{b\_gain}$ are applied on the raw image 1 : Gains from AWB module are applied
$\boldsymbol{r\_gain}$	Red channel gain.
$\boldsymbol{b\_gain}$	Blue channel gain.

Table 7: WB configuration parameters

3A STATS

The 3A Stats module encompasses three essential aspects of the ISP (Image Signal Processing) pipeline: Auto White Balance (AWB), Auto Focus (AF), and Auto Exposure (AE). These 3A algorithms do not modify the image directly, but instead provide feedback in the form of 3A statistical parameters which are used by other pipeline modules. The AF module will be implemented in the future.

Auto White Balance

AWB computes white balance gains as 3A statistics using the Gray World algorithm. In the White Balance (WB) module the AWB calculated gains are applied to the raw image. The AWB module calculates gains that adjusts the color balance in an image to restore the accurate colors of gray pixels. After researching a variety of algorithms, the following algorithms were selected based on their performance and computational complexity.

1. Gray World Algorithm
2. Norm-2 Gray World
3. PCA Illuminant Estimation

Before calculating white balance gains, a pre-processing pixel filtering step is performed, in which a percentage of underexposed and overexposed pixels are removed from the image. The percentage of underexposed and overexposed pixels is provided through AWB configuration parameters ($\boldsymbol{underexposed\_percentage}$, $\boldsymbol{overexposed\_percentage}$). These pixels are not used in the next step. The above mentioned AWB algorithms are explained below:

Gray World Algorithm

The Gray World algorithm is a white balance method based on the assumption that, on average, a scene's color is neutral gray. This assumption holds when there is a balanced distribution of colors in the scene. Given this balanced distribution, the average reflected color represents the color of the light source. To estimate the color cast of the illumination, the algorithm compares the average color to gray. The Gray World algorithm calculates an illumination estimate by computing the mean of each image channel

$$C_{avg} = \frac{\sum_{n=0}^{N} C(n)}{N}$$

$C \in \{R, G. B\}$ and N is the total number of pixels in the image. White balance gains for red and blue channels are calculated as shown below. These gains are then applied to the corresponding channels of the next frame.

$$R_{gain} = \frac{G_{avg}}{R_{avg}}$$

$$B_{gain} = \frac{G_{avg}}{B_{avg}}$$

Norm-2 Gray World

Norm-2 gray world works on the same assumption as the gray world but uses L2-norm to calculate channel averages for gain calculations.

$$C_{avg} = \frac{\sum_{n=0}^{n} (I_c(n)^p)^{1/p}}{N}$$

Where $p = 2$ for L2-Norm. After obtaining the channel averages gain calculation is same as gray world algorithm. Norm-2 Gray world performs better than gray world white balance, especially in outdoor scenes and images with larger green sections; it produces less color cast.

PCA Illuminant Estimation

PCA illuminant estimation is one of the best conventional algorithms for white balance. It shows that spatial information does not provide additional information that cannot be obtained directly from the color distributions. The algorithm is an efficient illumination estimation method that chooses bright and dark pixels using a projection distance in the color distribution and then applies PCA to estimate the illumination direction, as shown in the figure below.

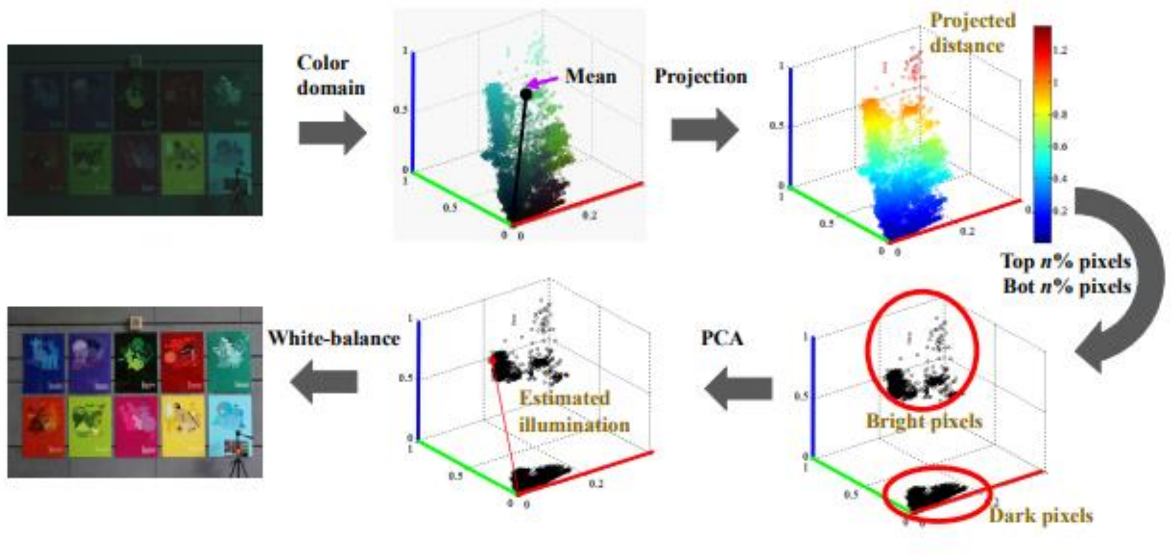


Figure 4: Visual Demonstration of PCA Illuminant Estimation

This method gives state-of-the-art results on existing general illumination datasets. The drawback of this method is its computational complexity.

Configuration Parameters

Parameters	Details
is_enable	Calculate white balance gains of the image when enabled: False: Disable True: Enable
$stats\_window\_offset$	Specifies crop dimensions to obtain a stats calculation window Should be an array of element, [Up, Down, Left, Right] Should be a multiple of 4
$underexposed\_percentage$	[0 - 100] - Set % of dark (underexposed) pixels to exclude before AWB gain calculation

$overexposed\_percentage$	[0 - 100] - Set % of saturated (overexposed) pixels to exclude before AWB gain calculation
$algorithm$	For selection of AWB algorithm, valid values are: - grey_world - norm_2 - pca
$percentage$	For PCA Illuminant Estimation Algorithm: Set the percentage of light and dark pixels for illuminant estimation

Table 8: AWB configuration parameters

Auto Exposure

AE is one of the three modules of 3A Control. Auto-exposure (AE) is a crucial feature in consumer digital cameras. AE generates a parameter based on the evaluation of the current image to auto-adjust the camera for the next frame[1]. This parameter is then used by the Digital Gain module.

High-end cameras primarily control scene brightness using exposure, aperture, and analog gain. However, video cameras have a lower exposure limit due to the required frame rate. Additionally, economical cameras often have limited sensor control, so AE control settings are mainly adjusted through gains (Analog, Digital, and ISP gain, which together contribute to the scene's ISO). In the model AE feedback is used by Digital Gain module to adjust image brightness.

Skewness for Luminance Histogram

The primary AE stat for determining scene brightness is the skewness of the scene's luminance histogram. The skewness of the luminance histogram around a central luminance provides insight into whether an image is underexposed or overexposed.

Skewness

Moments are widely used in image analysis since they can derive invariants related to specific transformation classes. A moment is a quantitative measure of the shape of a set of points. The nth moment about zero of a probability density function $f(x)$ is the expected value of $x(n)$, referred to as a raw moment. The moments about the mean (with C being the mean) are called central moments, which describe the shape of the function independently of translation.

The nth moment of a real-valued continuous function $f(x)$ of a random variable x about a value C is given by

$$\mu_n = \int_{-\infty}^{\infty} (x - C)^n f(x) dx$$

Central moments are typically used for the second and higher moments, as they offer more precise information about the distribution's shape. The first central moment μ_1 is mean, and the second central moment μ_2 is the variance. The third central moment is skewness that is defined as a measure of a distribution's lopsidedness. A symmetric distribution has a skewness of zero, while a negatively skewed distribution has a longer tail on the left and a positively skewed distribution has a longer tail on the right.

If a grayscale distribution is uniform, it has a high dynamic range, high contrast, and produces a clear image. A uniform grayscale distribution is also symmetric, with a third central moment equal to zero. Experimental results show that as a grayscale distribution becomes more uniform, the skewness of the histogram approaches zero, especially for images with non-bimodal histograms. Consequently, the skewness of a histogram is used to evaluate the uniformity of the histogram.

[1] Note that images are continuously being evaluated as they flow through the pipeline and the pipeline parameters are being adjusted till the point the image is actually taken.

In the model digital gain is applied to an image to correct the histogram skewness. The skewness as defined above using the central moment equation results in an excessively large value. So, the skewness is calculated using Fisher Pearson coefficient of skewness:

$$skewness = \frac{\mu_3}{\mu_2^{\frac{3}{2}}}$$

Where:

$$\mu_n = \int_{-\infty}^{\infty} (x - C)^n f(x)dx$$

C is provided my AE configuration parameter $\boldsymbol{center\_illuminance}$. $skewness$ is calculated above along with $\boldsymbol{histogram\_skewness}$ defines the status of the image exposure settings:

- $|skewness| \leq \boldsymbol{histogram\_skewness}$: Image has corrected exposure
- $skewness < -\boldsymbol{histogram\_skewness}$: Image is underexposed
- $skewness > \boldsymbol{histogram\_skewness}$: Image is overexposed

Configuration Parameters

Parameters	Details
$\boldsymbol{is\_enable}$	When enabled, apply the 3A - Auto Exposure algorithm: False: Disable True: Enable
$\boldsymbol{stats\_window\_offset}$	Specifies crop dimensions to obtain a stats calculation window Should be an array of element, [Up, Down, Left, Right] Should be a multiple of 4
$\boldsymbol{center\_illuminance}$	Pixel Values around which skewness is calculated The value of center illuminance for skewness calculation ranges from 0 to 255
$\boldsymbol{histogram\_skewness}$	Along with the *skewness* calculated above, it defines the exposure settings

Table 9: AE configuration parameters

Color Filter Array

The CFA module is an integral component of the ISP pipeline that transforms the black-level corrected, white-balanced, linearized 2D image into a 3D RGB image format.

A Color Filter Array (CFA) is positioned in front of the sensor to capture color information. This array typically made of three filters, restricts the sensitivity of each photocell to one part of the visible spectrum. As a result, each pixel of the CFA image only contains information about this limited range, i.e., one color response. However, to render images on a specific display device, three colors per pixel are required. Demosaicing is therefore employed to reconstruct the missing colors at each pixel location.

Malvar-He-Cutler

After thorough analysis, Malvar-He-Cutler demosaicing algorithm was selected (shown in Figure 5). Image demosaicing, or demosaicing, is the interpolation problem of estimating complete color information for an image captured through a Color Filter Array (CFA), particularly using the Bayer pattern. Demosaicing is achieved by convolving the image with a set of linear filters. There are eight different filters shown in Figure 6Figure 6 for interpolating the various color components at different locations.

The CFA module is implemented in the pipeline after DPC, BLC, and WB. Instead of using a constant or near-constant hue approach, the algorithm uses the criterion of edges to have much stronger luminance than chrominance components.

In the Malvar-He-Cutler algorithm, to interpolate a green value at a red pixel location, the algorithm compares the red pixel information with its estimate of a bilinear interpolation of the nearest red samples. If it differs from that estimate, it means there is a sharp luminance change at that pixel. Consequently, the algorithm bilinearly interpolates the green value by adding a portion of this estimated luminance change. This method is a gradient-corrected bilinear interpolation approach and its flowchart is shown below:

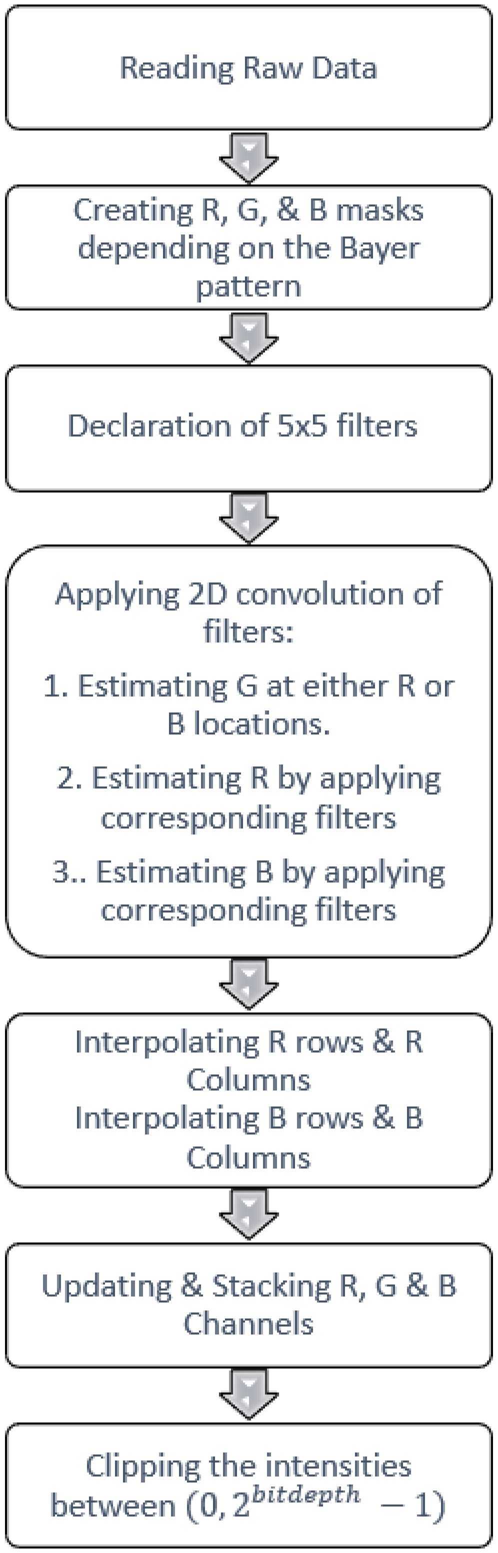

Reading Raw Data
Creating R, G, & B masks depending on the Bayer pattern
Declaration of 5x5 filters
Applying 2D convolution of filters:
1. Estimating G at either R or B locations.
2. Estimating R by applying corresponding filters
3.. Estimating B by applying corresponding filters
Interpolating R rows & R Columns
Interpolating B rows & B Columns
Updating & Stacking R, G & B Channels
Clipping the intensities between $(0, 2^{bitdepth} - 1)$

Figure 5: Flowchart for the Malwar He Cutler's Demosaicing Algorithm

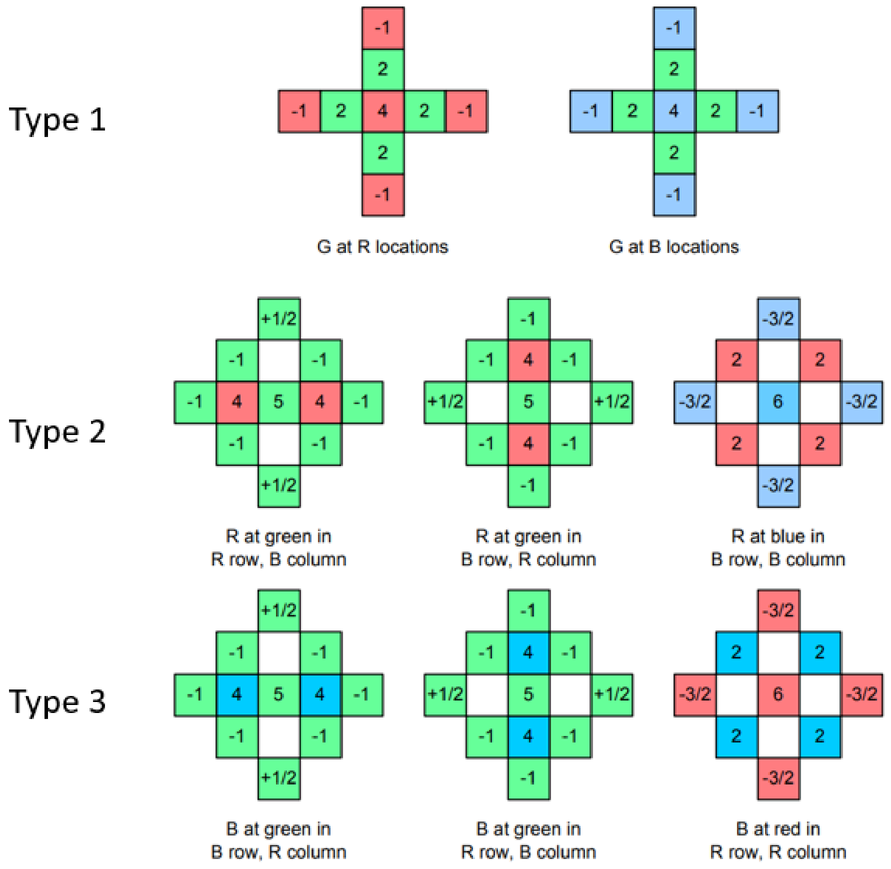


Figure 6: Filter Coefficients for linear interpolation of R, G & B data.

The Malvar-He-Cutler demosaicing algorithm is explained below:

1. **Mask Generation:** From a raw Bayer image shown in Figure 7 three r, g, and b masks are generated shown in Figure 8.

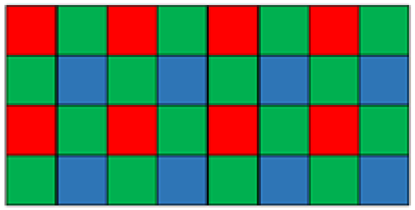


Figure 7: Raw Image with RGGB Bayer pattern

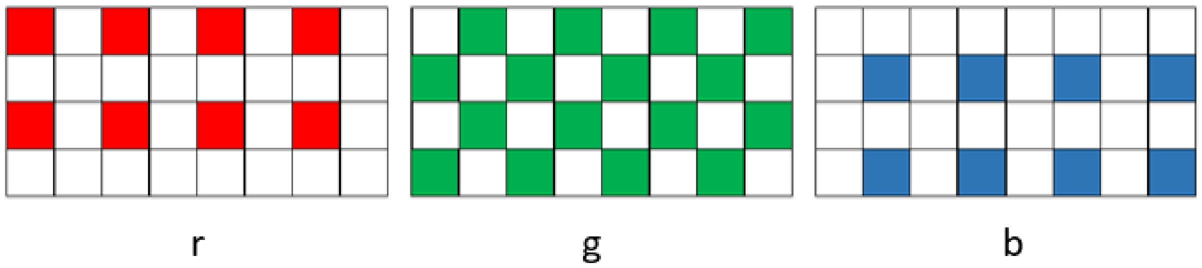


Figure 8: Masking channels

G Channel Extraction

1. On G channel, g pixels at r and b locations are estimated using the result of the convolution of filters from type 1 in Figure 6 with the raw image as shown below.

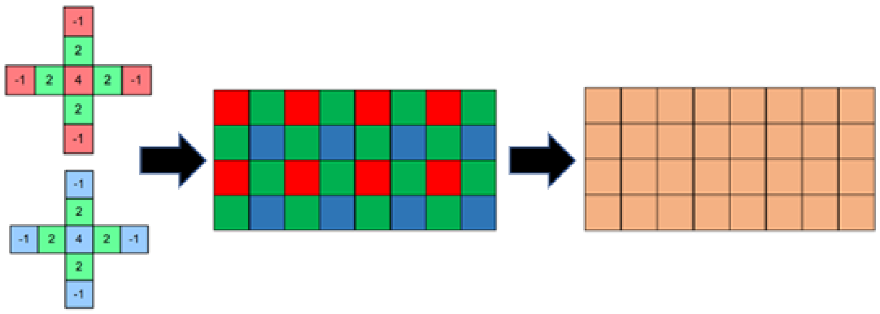


Figure 9: Convolution of a raw image with Filter Type 1

2. Result of step 1 is masked through r and b masks to get g pixels value at r and b location as shown below:

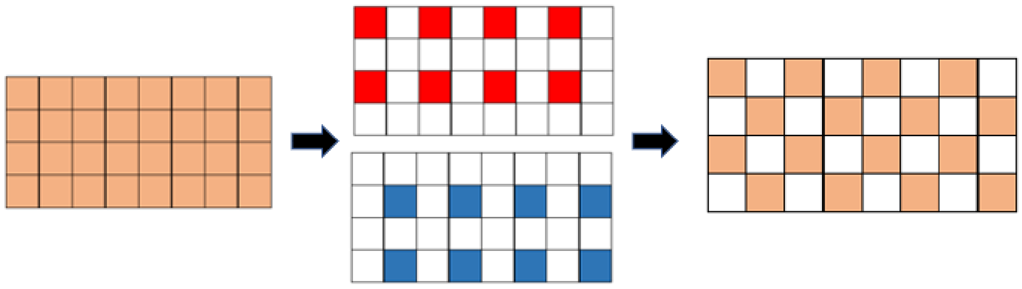


Figure 10: Estimation of g at r and b locations

3. Interpolated g values of Step 2 are added at the respective r and b location to obtain a final G channel:

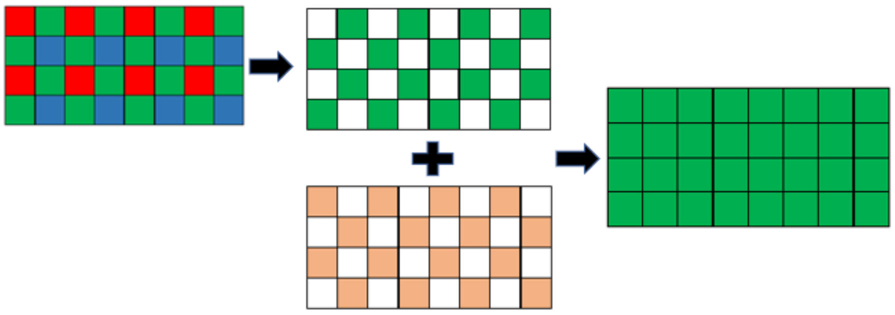


Figure 11: Final G channel

R Channel Extraction

1. Identify three locations where R has to be estimated as shown below, essentially creating three masks:

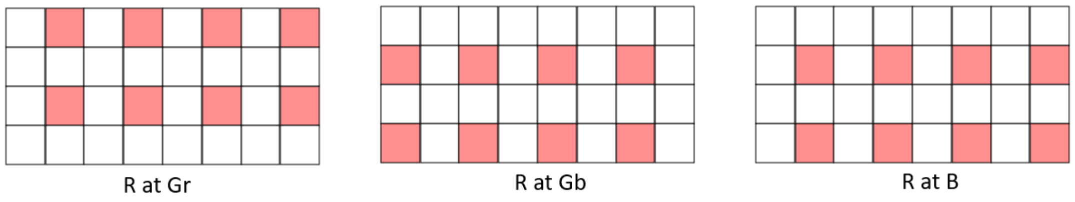


Figure 12: Location of R pixels masks

2. Convolve the raw image with each of the three types 2 filters shown in Figure 6 to produce three R estimated images.

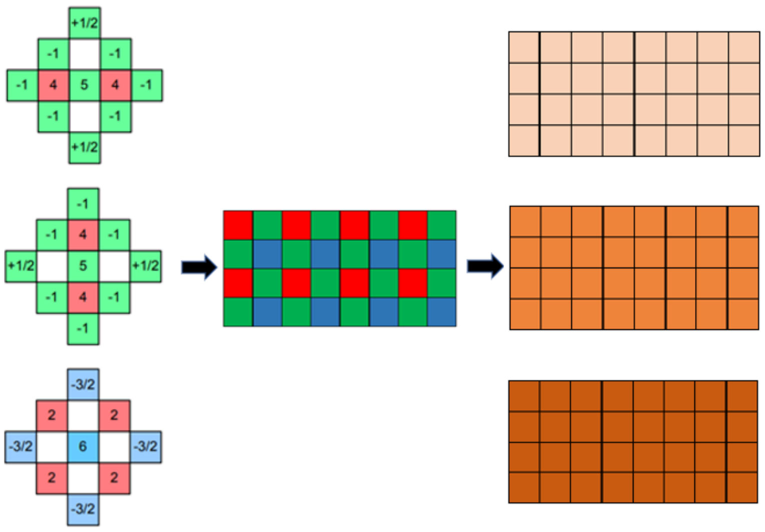


Figure 13: Convolution of Type 2 filters with raw image for R estimated images

3. Use the estimated R images produced in Step 2 along with the masks produced in Step 1 to get the estimated R channel values.

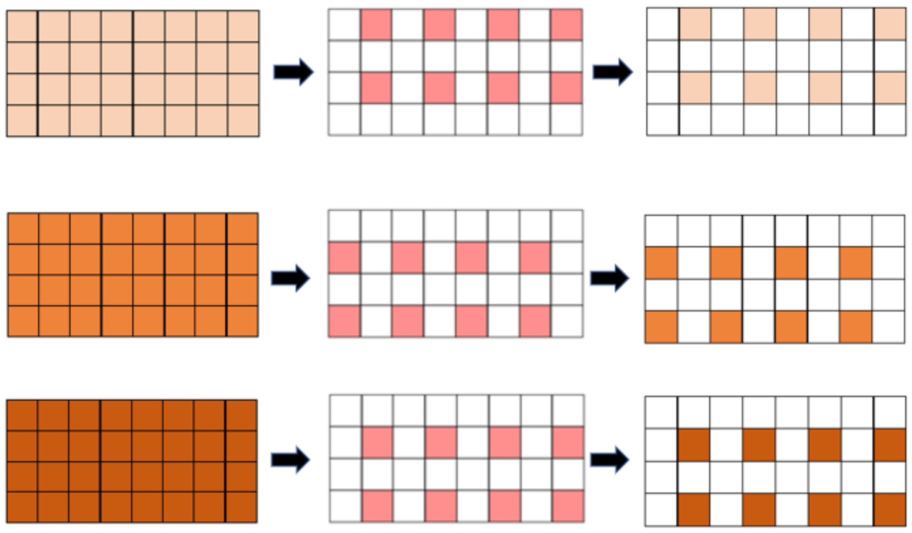


Figure 14: Extracting estimated R values

4. Estimated R channels values from step 3 together with R channel values of raw image produce the final R channel.

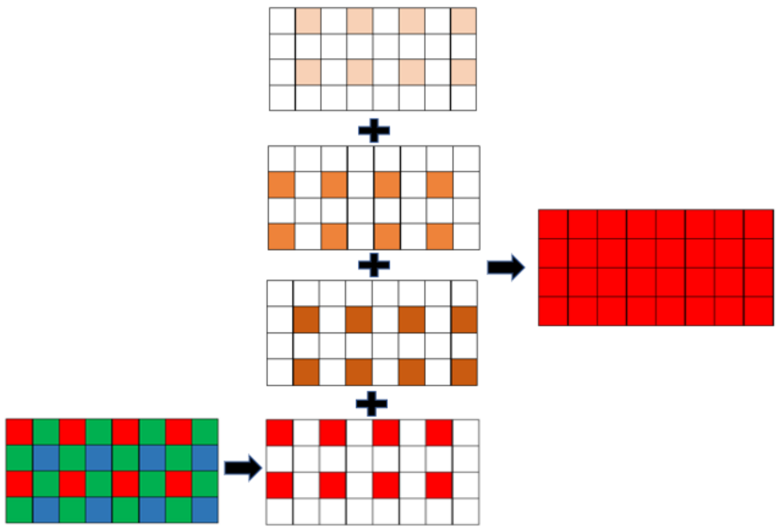


Figure 15: Final R Channel

B Channel Extraction

For B channel extraction the same procedure as for the R channel is followed. At Step 1 estimating locations are now B at Gb, B at Gr and B at R where B channel has to be estimated and filters from type 3 will be applied in Step 2.

Configuration Parameters

CFA has no configuration parameters and is always active in the pipeline.

Color Correction Matrix

To optimize color output in digital images, it is essential to address factors such as optical spectral properties (including lenses and filters), a variety of lighting sources (such as tungsten, fluorescent, and daylight), and the characteristics of the sensor's color that can lead to color inaccuracies in the captured image. The Color Correction Matrix (CCM) module plays a critical role in counteracting the effects of incorrect color representation and applies a color correction matrix to the image, ensuring a more precise display of colors as captured by the source imaging system.

Algorithm Explanation

The Color Correction Matrix (CCM) is a 3x3 matrix specifically designed to adjust the Red, Green, and Blue (RGB) color channels of an image. Each component within the matrix acts as a scaling factor, modifying the color values of each channel based on a calculated combination of the original values. This matrix is systematically applied to every pixel within the image, effectively transforming the color values in accordance with the given formula:

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = CCM * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} RR & RG & RB \\ GR & GG & GB \\ BR & BG & BB \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Configuration Parameters

Parameters	Details
is_enable	When enabled, the user given 3x3 CCM is applied to the 3D RGB image with rows sum to 1 convention: False: Disable True: Enable
corrected_red	Row 1 of CCM
corrected_green	Row 2 of CCM
corrected_blue	Row 3 of CCM

Table 10: CCM configuration parameters

Gamma Correction

Gamma correction alters the input image by translating it to a nonlinear space. It matches the nonlinear response of display devices and broadens or compresses the dynamic range of images.

Algorithm Explanation

A look-up table is a common method for implementing gamma correction in hardware and is simple to modify and encode.

Configuration Parameters

Parameters	Details
is_enable	When enabled, applies tone mapping gamma using the look-up table: False: Disable True: Enable
$gamma\_lut\_8$	The look-up table for the gamma curve for 8-bit image
$gamma\_lut\_10$	The look-up table for the gamma curve for 10-bit image
$gamma\_lut\_12$	The look-up table for the gamma curve for 12-bit image
$gamma\_lut\_14$	The look-up table for the gamma curve for 14-bit image

Table 11: GC configuration parameters

Color Space Conversion

The color space conversion transforms the RGB color representation of an image to another color space. Different color spaces have been designed to accommodate varying proportions of chroma and luma. Hardware displays typically utilize the YUV color space. This choice is primarily rooted in the history of television technology, where black and white televisions relied solely on a single luminance channel. As color television emerged, the U and V channels were introduced to transmit color information, while the Y channel continued to convey luminance, thus ensuring backward compatibility with earlier black and white televisions.

Algorithm Explanation

In the model, the color space conversions are based on ITU-R standards. There are three ITU standards for YUV conversion.

1. BT.601 – for SDTV
2. BT.709 – for HDTV

We have implemented BT.601. BT.709 while BT.2020 will be implemented in the future.

BT.601

The conversion formula uses the ITU-R standard, multiplying the YUV_{mat} to the RGB image.

$$YUV_{mat} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{bmatrix}$$

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = YUV_{mat} * \begin{bmatrix} R_{norm} \\ G_{norm} \\ B_{norm} \end{bmatrix}$$

The conversion requires a normalized RGB image, and the output YUV has a Y range from 0 to 1, whereas U and V range from -0.5 to 0.5. This conversion is not ideal for RTL implementation due to the multiplication involving floating-point numbers. Therefore, YUV_{mat} is converted into an integer matrix, also known as YCrCb, to enable digital conversion. The process for converting RGB to YC_rC_b using the $YC_rC_{b\,mat}$ involves the following steps:

1. Multiply YUV_{mat} with 2^m, where m is the matrix precision bits to get $YC_rC_{b\,mat}$. The higher the bit value, the higher the precision, resulting in multiplication with larger bit depths. m can only be an even integer from 8 to 16. For the model, m is set to 8.

$$YC_rC_{b\,mat} = YUV_{mat} * 2^m$$

$$YC_rC_{b\,mat} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{bmatrix} * 2^m$$

$$YC_rC_{b_{mat}} = \begin{bmatrix} 77 & 150 & 29 \\ 131 & -110 & -21 \\ -44 & -87 & 138 \end{bmatrix}$$

2. Multiply RGB with the $YC_rC_{b_{mat}}$ and add a digital offset to correct the black level of the YCrCb image. Divide the result of this matrix multiplication by 2^m to obtain a YCrCb image with the same bit depth as the input RGB image. Then, add a digital offset matrix to correct the black level of the YC_rC_b image.

$$\begin{bmatrix} Y \\ C_r \\ C_b \end{bmatrix} = \frac{\left(YC_rC_{b_{mat}} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} \right)}{2^m} + \begin{bmatrix} Y_{offset} \\ C_{r_{offset}} \\ C_{b_{offset}} \end{bmatrix}$$

3. This offset matrix depends on bit depth of YC_rC_b image n and it can be defined as:

$$\begin{bmatrix} Y_{offset} \\ C_{r_{offset}} \\ C_{b_{offset}} \end{bmatrix} = \begin{bmatrix} 2^{\frac{n}{2}} \\ 2^{n-1} \\ 2^{n-1} \end{bmatrix}$$

BT.709

BT.709 conversion is similar to BT.601 with different conversion matrix:

$$YUV_{mat} = \begin{bmatrix} 0.2126 & 0.7152 & 0.0722 \\ -0.1147 & -0.3854 & 0.5 \\ 0.5 & -0.4542 & -0.0458 \end{bmatrix}$$

This decimal matrix also known as analog matrix is also converted into digital matrix in the same way described in BT. 601 section following same steps to achieve the final YC_rC_b Image.

$$YC_rC_{b_{mat}} = \begin{bmatrix} 47 & 157 & 16 \\ -26 & -86 & 112 \\ 112 & -102 & -10 \end{bmatrix}$$

Note: In all the module after the CSC module, YC_rC_b image is referred as YUV image.

Configuration Parameters

Parameters	Details
conv_standard	Set standard to be used for conversion 1 : Bt.709 HD 2 : Bt.601/407

Table 12: CSC configuration parameters

Local Dynamic Contrast Improvement

The Local Dynamic Contrast Improvement (LDCI) module in the Infinite-ISP is designed to enhance the contrast of images, especially in low light scenarios. After a thorough evaluation of various algorithms, the CLAHE (Contrast Limited Adaptive Histogram Equalization) method was selected because of its superior performance and straightforward complexity.

Contrast Limited Adaptive Histogram Equalization (CLAHE)

CLAHE is an advanced method of histogram equalization, a technique used to improve the contrast in images. Unlike standard histogram equalization that applies a single transformation function to the entire image, CLAHE operates on small regions (tiles/window) in the image. This ensures that the contrast enhancement is adaptive and caters to local variations in brightness and contrast. The algorithm targets the luminance channel of the RGB image, which is derived post the CSC module processing.

1. Divide the image into small, non-overlapping regions or tiles (e.g., 8x8 or 16x16 pixels).
2. Compute the histogram for each tile, representing the distribution of pixel intensities within that region.
3. For each tile's histogram, if any bin exceeds a predefined contrast limit:
    - Clip the excess from that bin.
    - Redistribute the clipped excess uniformly among all histogram bins.
4. Apply histogram equalization to each tile independently using its clipped histogram.
5. After histogram equalization of tiles, interpolate between neighboring tiles to ensure a smooth transition and eliminate any potential artifacts at tile boundaries.
6. Combine the processed tiles to form the enhanced image with improved local contrast.

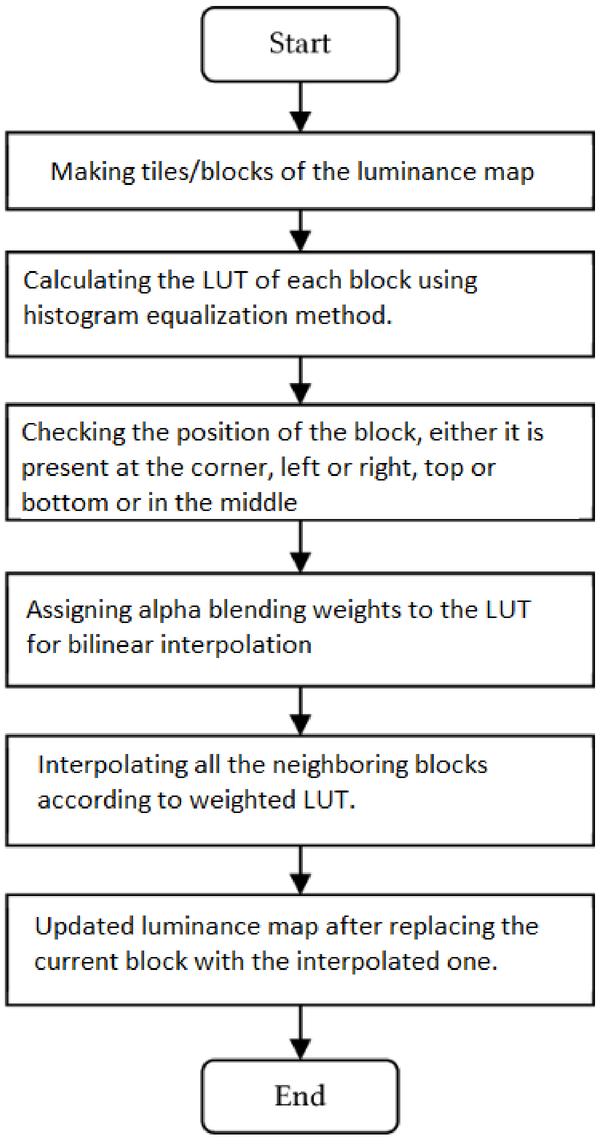

Start
Making tiles/blocks of the luminance map
Calculating the LUT of each block using histogram equalization method.
Checking the position of the block, either it is present at the corner, left or right, top or bottom or in the middle
Assigning alpha blending weights to the LUT for bilinear interpolation
Interpolating all the neighboring blocks according to weighted LUT.
Updated luminance map after replacing the current block with the interpolated one.
End

Figure 16: Flowchart for LDCI

Configuration Parameters

Parameters	Details
is_enable	When enabled, applies tone mapping gamma using the look-up table: False: Disable True: Enable
clip_limit	The clipping limit controls the amount of detail to be enhanced
wind	Window/tile size.

Table 13: LDCI Configuration Parameters

Sharpening

The Sharpen module amplifies the selected high-frequency band of the image in order to improve the edges and texture details of the image. The strength of the applied sharpening and the band selection are dependent on the 2DNR module in the pipeline to ensure high visual fidelity.

Unsharp Masking

The Unsharp Masking uses the luma component of the image. In the first step, we convolve the luminance Y component of the input image with a Gaussian filter $G(\sigma)$ where σ is the standard deviation value. Next, we use this attained low-frequency component to extract the remaining higher frequencies H from the image in the following way:

$$H = Y - (G(\sigma) * Y)$$

The extracted high frequencies of then enhanced by the given strength k and then added back into the original luminance value in the following way to achieve the sharpened luma Y_{shrp}:

$$Y_{shrp} = Y + k \cdot H$$

Configuration Parameters

Parameters	Details
$\boldsymbol{isEnable}$	When enabled, applies the sharpening to the image False: Disable True: Enable
$\boldsymbol{sharpen\_sigma}$	[1, 10] – Parameter to define the Standard Deviation of the Gaussian Filter
$\boldsymbol{sharpen\_strength}$	[0.1, 2] – Parameter controls the sharpen strength applied on the high frequency components

Table 14: Sharpening Configuration Parameters

2D Noise Reduction

A denoising tool, 2D Noise Reduction (2DNR), is incorporated into the system to minimize the impact of image noise, which deteriorates image quality. In conducting a literature review, the behavior of various spatial domain denoising methods was analyzed, along with the effects caused by adaptively chosen weights associated with them. Taking into account the hardware and computational constraints, we selected the Non-Local Mean (NLM) algorithm for denoising images.

Non-Local Means Filter

The NLM algorithm is a spatial domain denoising technique. These types of methods can be further categorized as local and non-local filters. Natural images exhibit self-similarity, and the NLM filter leverages this characteristic. NLM denoises an image by taking the weighted average values of all similar pixels throughout the entire image.

The weights assigned to these similar pixels are computed based on the Euclidean distance between the intensities of the pixels. They are sorted in decreasing order to assign more weight to the most similar pixel in the image. This concept is founded on the idea that, for denoising a single pixel in an image, the weighted average is computed using all those pixels in the entire image, which share similar intensity or gray levels, and are close to each other based on their Euclidean distance.

By considering the spatial relationships between pixels and their intensities, the NLM filter effectively reduces noise while preserving important image details. However, it may be more computationally intensive than other denoising techniques due to its non-local approach. To minimize time overhead, the shifted array approach is employed in the implementation.

Mathematically, we can represent this as:

$$I_{denoised}(x) = \frac{1}{C(x)} \sum_{y\, \in \Omega} I(y).W(x,y)$$

$$W(x,y) = \exp(-\frac{\left|\left|I(x) - I(y)\right|\right|^2}{h^2})$$

$I_{denoised}(x)$, is the denoised intensity value at x.

$C(x)$, is the normalization factor to ensure that the weights sum up to 1.

Ω, represents the search window containing all pixels in the image.

$W(x,y)$, is the weight assigned to each pixel in the search window based on its similarity with the centered pixel.

h, is the strength parameter that controls the strength of the denoising effect by influencing the assignment of weights.

$I(y)$, represents all the pixels placed in the search window.

Configuration Parameters

Parameters	Details
is_enable	When enabled, apply the non-local mean filtering: False: Disable True: Enable
window_size	Search window size for NLM filter. It should be odd 9 is set as default to avoid the computational cost.
patch_size	Window for applying Mean Filter within a Search Window It Should be odd 5 is set as default
wts	0-100] Strength parameter for NLM. The higher the value, the higher will be the smoothing or blurring Set to 5

Table 15: 2DNR configuration parameters

RGB Conversion

RGB conversion module determines the output format of the model. If enabled it performs inverse color space conversion from YUV to RGB based on conversion type specified in CSC module parameter otherwise the pipeline outputs a YUV image.

The conversion formula uses the ITU-R standard for BT-601 and BT_709, multiplying the RGB_{mat} to the YUV image.

$$RGB_{mat} = {YUV_{mat}}^{-1}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = RGB_{mat} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

RGB_{mat} matrix is a decimal therefore; it is converted into an integer matrix using the following steps:

1. Multiply YUV_{mat} with 2^m, where m is the matrix precision bits to get $YC_rC_{b\,mat}$. The higher the bit value, the higher the precision, resulting in multiplication with larger bit depths. m can only be an even integer from 8 to 16. For the model, m is set to 8

$$digital\_RGB_{mat} = RGB_{mat} * 2^m$$

$$digital\_RGB_{mat} = \begin{bmatrix} 74 & 0 & 144 \\ 74 & -13 & -34 \\ -74 & 135 & 0 \end{bmatrix}$$

2. Subtract the digital offset and multiply YUV image with the $digital\_RGB_{mat}$. Divide the result of this matrix multiplication by 2^m to obtain a RGB image with the same bit depth as the input RGB image.

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{digital\_RGB_{mat} * \left(\begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} Y_{offset} \\ C_{r\,offset} \\ C_{b\,offset} \end{bmatrix} \right)}{2^m}$$

3. This offset matrix depends on bit depth of YUV image bit depth $n = 8$ and it can be defined as:

$$\begin{bmatrix} Y_{offset} \\ C_{r\,offset} \\ C_{b\,offset} \end{bmatrix} = \begin{bmatrix} 2^{\frac{n}{2}} \\ 2^{n-1} \\ 2^{n-1} \end{bmatrix}$$

Configuration Parameters

Parameters	Details

is_enable	When enabled, converts output YUV image to RGB False : Disable – Output image format is YUV True : Enable – Output Image format is RGB

Table 16: RGBC configuration parameters

Scale

The Scale block is implemented as one of the final blocks in the ISP pipeline to downscale a full-resolution image to a supported resolution/size per the user's requirements. The Scale block is generally implemented in IP camera ISPs to create multiple channels/streams for storage and transmission in surveillance cameras.

Scaling only preserves the aspect ratio (the ratio between image height and width $\frac{h}{w}$) when height and width are scaled by the same factor $\frac{h*c}{w*c} = \frac{h}{w}$. Cropping does not preserve the aspect ratio even if same amount is cropped from both height and width $\frac{h-c}{w-c} \neq \frac{h}{w}$.

Up/Down scaling height and width at the same time is equivalent to scaling the height first and width second or vice versa. For example, downscaling with Nearest Neighbor method using a square 2×2 window is equal to downscaling by 2×1 window followed by downscaling with 1×2 window.

For scaling, following two methods are supported in our model:

- Nearest Neighbor
- Bilinear Interpolation

Nearest Neighbor

Depending upon the scale factor, say "n", every nth pixel is dropped or replicated to downscale or upscale the input image respectively.

Bilinear Interpolation:

Depending upon the scale factor, weighted averaging of neighboring pixels is employed to interpolate new pixel values or map multiple pixels values to a single pixel value to upscale or downscale an image respectively.

Hardware Friendly Approach

This approach is implemented to reduce the computational complexity of the scale module and can be enabled using the is_hardware flag.

- Input image: 3D array (RGB or YUV image) of size$2592x1536$, $2592x1440$ or $1920x1080$.
- Output image: Downscaled 3D array (RGB or YUV image). The valid output sizes corresponding to each input size are mentioned in Table 17.

The image is scaled using the following steps:

1. Down-scale the image by an integer scaling factor in {2,3,4} using bilinear interpolation.
2. Crop the image to required output-size if needed.

3. Scale the image using a rational scale factor from $\{\{2/3, 3/4, 4/7, 5/7\}\}$ using nearest neighbor or bilinear method specified by the parameter Algo.
4. Scales each channel one by one using the same steps.

The table below shows the hand-picked scaling factors and crop values that have been used in the design.

Input Size	Output size	Downscale factor		Crop value		Non-integer scale factor	
		width	height	width	height	width	height
$\mathbf{2592 \times 1944}$	2560×1440	–	–	32	24	–	$3/4$
$\mathbf{2592 \times 1944}$	1920×1080	–	–	32	54	$3/4$	$4/7$
$\mathbf{2592 \times 1944}$	1280×960	2	2	16	12	–	–
$\mathbf{2592 \times 1944}$	1280×720	2	2	16	12	–	$3/4$
$\mathbf{2592 \times 1944}$	640×480	4	4	8	6	–	–
$\mathbf{2592 \times 1944}$	640×360	4	4	8	6	–	$3/4$
$\mathbf{1920 \times 1080}$	1280×720	–	–	–	–	$2/3$	$2/3$
$\mathbf{1920 \times 1080}$	640×480	3	2	–	60	–	–
$\mathbf{1920 \times 1080}$	640×360	3	3	–	–	–	–
$\mathbf{2592 \times 1536}$	1920×1080	–	–	32	24	$3/4$	$5/7$
$\mathbf{2592 \times 1536}$	1280×720	2	2	16	48	–	–
$\mathbf{2592 \times 1536}$	640×480	4	3	8	32	–	–
$\mathbf{2592 \times 1536}$	640×360	4	4	8	24	–	–

Table 17: Scaling – Valid Output Sizes

Configuration Parameters

Parameters	Details
$\boldsymbol{is\_enable}$	When enabled, scales down the input image False: Disable True: Enable
$\boldsymbol{new\_width}$	Downscaled width of the output image

Parameters	Details
new_height	Downscaled height of the output image
algorithm	Software friendly scaling. Only used when ***is_hardware*** is disabled. - Nearest_Neighbor (default) - Bilinear
is_hardware	When true applies the hardware friendly techniques for downscaling. This can only be applied to any one of the 3 input sizes and can downscale to - 2592x1944 to 1920x1080 or 1280x960 or 1280x720 or 640x480 or 640x360 - 2592x1536 to 1280x720 or 640x480 or 640x360 - 1920x1080 to to 1280x720 or 640x480 or 640x360
upscale_method	Used only when ***is_hardware*** is enabled. Upscaling method, can be one of the above algos
downscale_method	Used only when ***is_hardware*** is enabled. Upscaling method, can be one of the above algos

Table 18: Scale configuration parameters

YUV Format – 444-422

The YUV conversion format is a way of subsampling YUV images to save the bandwidth. Subsampling of YUV is possible because the human eye is less sensitive to color differences compared to luminance differences, allowing for subsampling of the C_rC_b channel without significantly affecting the output. That is why no subsampling is done for the Y channel.

Algorithm Explanation

In YUV format, the Y, U, and V components are stored in a single array, known as the packed format. Pixels are organized into macro pixel groups, with the layout depending on the YUV format. Each YUV format described has an assigned FOURCC code, which is a 32-bit unsigned integer created by concatenating four ASCII characters:

- Y : Luma
- U : Blue-Difference
- V : Red-Difference
- A : Transparency

YUV formats implemented in the model are

1. 4:4:4
   a. AYUV
2. 4:2:2
   a. YUYV
   b. UYVY

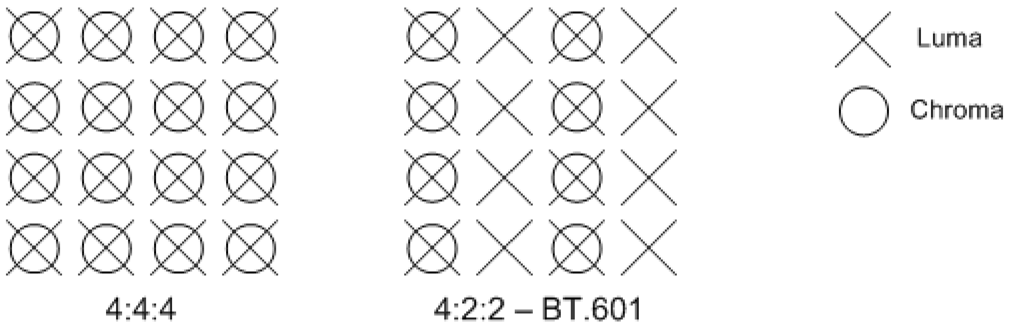


Figure 17: Pictorial representation of 4:4:4 and 4:2:2

4:4:4 Format

In this format, no subsampling is done, and YUV is saved in the following manner:

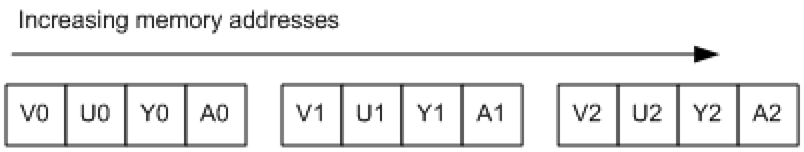
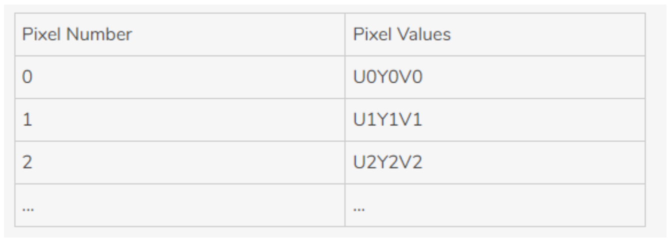


Figure 18: Memory view of bytes for packed format

The recommended FOURCC format for 4:4:4 is shown above. In FOURCC, each memory element is 32 bits with 4 bytes, each representing either Y, U, or V based on the chosen formatting. In the model, a simple 4:4:4 format is implemented, where each memory chunk has 3 bytes of Y, U, or V. Since there is no information regarding the alpha channel yet, it is not considered in this conversion.

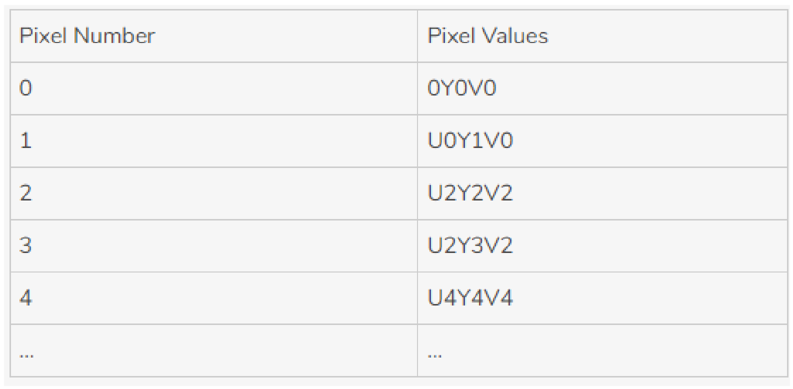


Pixel Number	Pixel Values
0	U0Y0V0
1	U1Y1V1
2	U2Y2V2
...	...

Figure 19: 4:4:4 format

4:2:2 Format

In this format, every two pixels share U and V. This is achieved by using the same U and V values for each pair of pixels. For example, the same U and V values are used for Y1 and Y2, and the same U and V values are used for Y3 and Y4.



Pixel Number	Pixel Values
0	0Y0V0
1	U0Y1V0
2	U2Y2V2
3	U2Y3V2
4	U4Y4V4
...	...

Figure 20: 4:2:2 format

In the FOURCC format, the above representation is saved as follows:

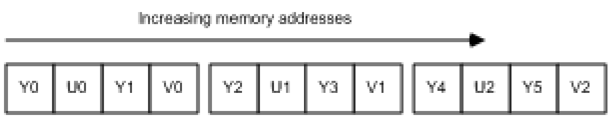


Figure 21: Memory view of 4:2:2

The first four bytes represent two pixels. Both pixels have the same U and V values but different Y values. Note that U and V in the Figure 21 are subsampled entries of the corresponding U and V channels.

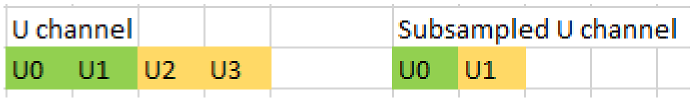


Figure 22: Subsample entries for 4:2:2

In the model, subsampling is done horizontally for each row of the U and V channels, and the YUYV format is implemented.

Configuration Parameters

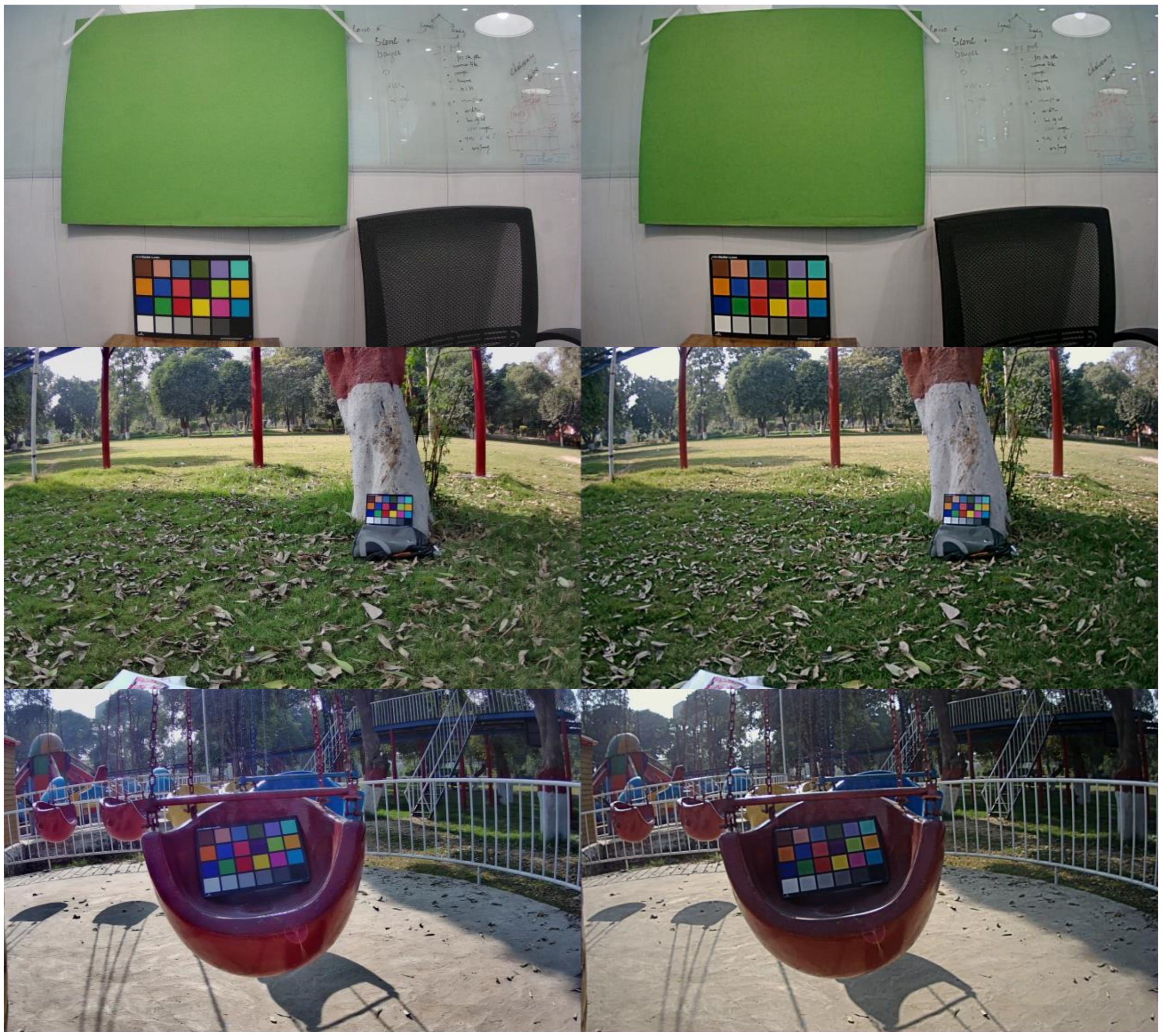
Parameters	Details
is_enable	Will enable or disable this module: False: Disable True: Enable
conv_type	Set conversion format of YC_rC_b to YUV. Can only be two values: - 444 - 422

Table 19: YUV Formats configuration parameters

Pipeline Results

Here are the results of this pipeline compared with a market-competitive ISP. The model outputs are displayed on the right, with the underlying ground truths on the left.

Ground Truths **Infinite-ISP Algorithm Design**



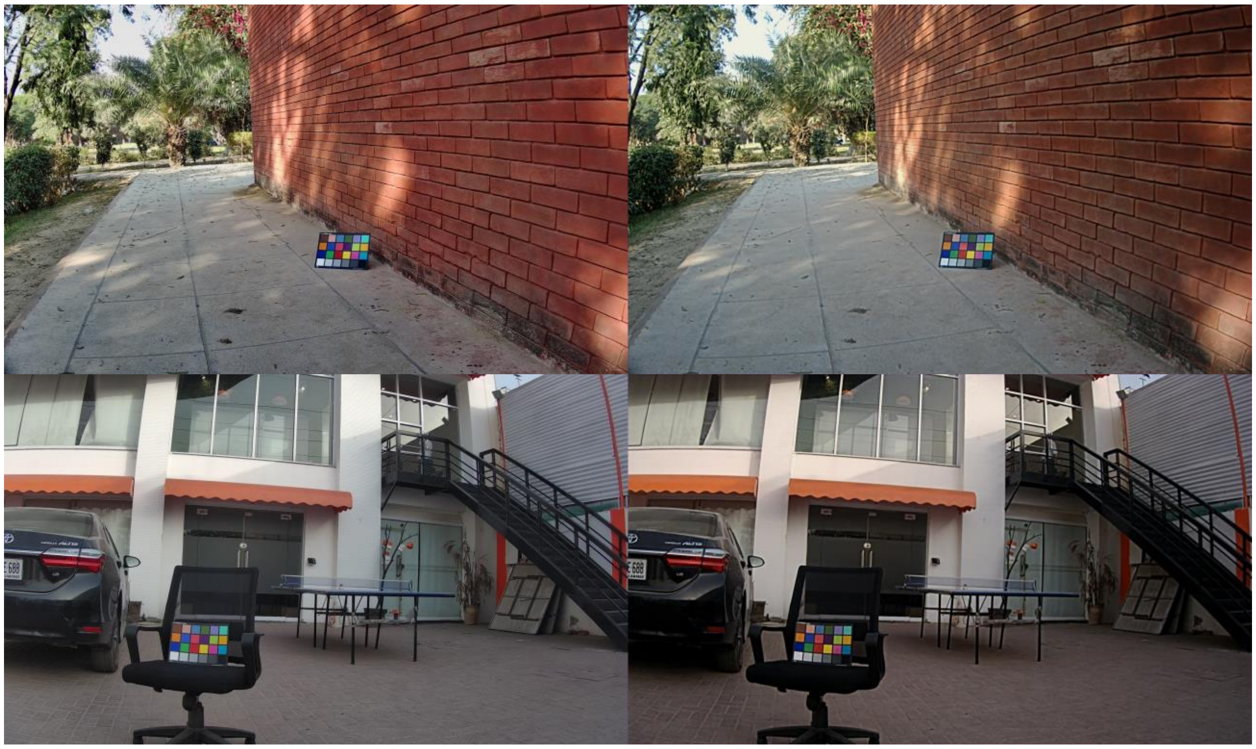


Figure 23: Pipeline Results

IQ Metrics Analysis

Images	PSNR	SSIM
Indoor1	20.0974	0.8599
Outdoor1	21.8669	0.9277
Outdoor2	20.3430	0.8384
Outdoor3	19.3627	0.8027
Outdoor4	20.7140	0.8561

Table 20: IQ metrics Analysis

References

1. Measurement for optoelectronic conversion functions (OECFs) of the digital still-picture camera - NASA/ADS (harvard.edu)
2. https://www.imatest.com/?s=oecf
3. https://www.imatest.com/docs/esfriso_instructions/
4. https://slideplayer.com/slide/5759755/
5. ISO - ISO 14524:2009 - Photography — Electronic still-picture cameras — Methods for measuring optoelectronic conversion functions (OECFs)
6. (PDF) Linearisation of RGB Camera Responses for Quantitative Image Analysis of Visible and UV Photography: A Comparison of Two Techniques (researchgate.net)
7. Linearization in detail - three different ways to linearize images (image-engineering.de)
8. https://patentimages.storage.googleapis.com/f9/11/65/a2b66f52c6dbd4/US8538199.pdf
9. https://static.aminer.org/pdf/PDF/000/319/504/large_scale_infographic_image_downsizing.pdf
10. https://www.ipol.im/pub/art/2011/g_mhcd/article.pdf
11. https://www.ipol.im/pub/art/2011/bcm_nlm/article.pdf
12. https://www.itu.int/rec/R-REC-BT.709-6-201506-I/en
13. https://www.flir.com/support-center/iis/machine-vision/knowledge-base/understanding-yuv-data-formats/
14. https://en.wikipedia.org/wiki/Chroma_subsampling
15. https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/YuY2_files/intro.htm#YV12
16. https://learn.microsoft.com/en-us/windows/win32/medfound/recommended-8-bit-yuv-formats-for-video-rendering
17. https://sci-hub.hkvisa.net/10.1109/ICAIIS49377.2020.9194921
18. https://sci-hub.hkvisa.net/10.1109/apccas.2012.6419046
19. https://www.atlantis-press.com/article/25875811.pdf
20. https://www.sciencedirect.com/science/article/abs/pii/0016003280900587
21. https://library.imaging.org/admin/apis/public/api/ist/website/downloadArticle/cic/12/1/art00008
22. https://opg.optica.org/josaa/viewmedia.cfm?uri=josaa-31-5-1049&seq=0
23. https://www.hindawi.com/journals/tswj/2014/979081/
24. https://www.semanticscholar.org/paper/Contrast-Limited-Adaptive-Histogram-Equalization-Zuiderveld/726c95fa3bd47401befc7513b6e52d1c806f26af.
25. https://web.archive.org/web/20120113220509/http://radonc.ucsf.edu/research_group/jpouliot/tutorial/HU/Lesson7.htm
26. https://github.com/cruxopen/openISP
27. https://github.com/QiuJueqin/fast-openISP