ON Semiconductor

Is Now

To learn more about onsemi™, please visit our website at www.onsemi.com

onsemi and onsemi and other names, marks, and brands are registered and/or common law trademarks of Semiconductor Components Industries, LLC dba "**onsemi**" or its affiliates and/or subsidiaries in the United States and/or other countries. **onsemi** owns the rights to a number of patents, trademarks, copyrights, trade secrets, and other intellectual property. A listing of **onsemi** product/patent coverage may be accessed at www.onsemi.com/site/pdf/Patent-Marking.pdf. **onsemi** reserves the right to make changes at any time to any products or information herein, without notice. The information herein is provided "as-is" and **onsemi** makes no warranty, representation or guarantee regarding the accuracy of the information, product features, availability, functionality, or suitability of its products for any particular purpose, nor does **onsemi** assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation special, consequential or incidental damages. Buyer is responsible for its products and applications using **onsemi** products, including compliance with all laws, regulations and safety requirements or standards, regardless of any support or applications information provided by **onsemi**. "Typical" parameters which may be provided in **onsemi** data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. **onsemi** does not convey any license under any of its intellectual property rights nor the rights of others. **onsemi** products are not designed, intended, or authorized for use as a critical component in life support systems or any FDA Class 3 medical devices or medical devices with a same or similar classification in a foreign jurisdiction or any devices intended for implantation in the human body. Should Buyer purchase or use **onsemi** products for any such unintended or unauthorized application, Buyer shall indemnify and hold **onsemi** and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that **onsemi** was negligent regarding the design or manufacture of the part. **onsemi** is an Equal Opportunity/Affirmative Action Employer. This literature is subject to all applicable copyright laws and is not for resale in any manner. Other names and brands may be claimed as the property of others.

AND9181/D

Sparse Color Filter Pattern Image File and Processing

ON Semiconductor®

www.onsemi.com

APPLICATION NOTE

Overview

This document gives guidelines for opening and processing a Sparse Color Filter Pattern (or Sparse CFA) image in Sensor Studio. Many of the requirements described in this document can also be found under the Help: Sensor Studio and Python Help Menu in Sensor Studio.

SPARSE CFA PROCESSING CODE OVERVIEW

ON Semiconductor provides several avenues of support for the Sparse Color Filter Pattern. The first is Sensor Studio software, which can process the Sparse CFA pattern and provide a RGB image. This allows users to process a raw Sparse CFA file with a known algorithm. The second form of support is ‘C’ source code which is available with a no-charge license agreement that can be compiled into an executable implementing the Sensor Studio Sparse CFA algorithm.

The purpose of the Sparse CFA image processing path in Sensor Studio and its corresponding ‘C’ code is to provide a simple algorithm based upon known computational methods that could be used to directly compare the imaging performance of a sensor with the Sparse CFA to an identical sensor with the Bayer CFA using the same processing technique. Note that the code has not been optimized in any way for speed performance, or to take advantage of alternative interpolation techniques.

There are a few techniques used in the Sparse CFA processing path that were employed to make sure that the integrity of the resolution was maintained. For example, the panchromatic channel is downsampled to match the color pixels so that the center of weighting for the panchromatic pixels matches the center of weighting of the color pixels when computing the color difference channel, helping to avoid aliasing artifacts. The color difference channel is then upsampled (after noise cleaning) and the full resolution panchromatic channel is applied to maintain full resolution.

Computing luminance and color difference channels is common practice because the human visual system is more sensitive to luminance variations than chrominance variations. This allows noise cleaning algorithms to apply different weights to the luminance channel than the color difference channel. It also allows sharpening to be performed on the luminance channel separately from the colors channel. Noise cleaning and sharpening algorithms are dependent upon the application and can be very simple or very complex. The noise cleaning and sharpening code that was used in the Sensor Studio processing path was intended to be common practice so that this code could be shared easily with customers.

The book “Single-Sensor Imaging: Methods and Applications for Digital Cameras” edited by Rastislav Lukac has a section that discusses the processing path.

EXPOSURE

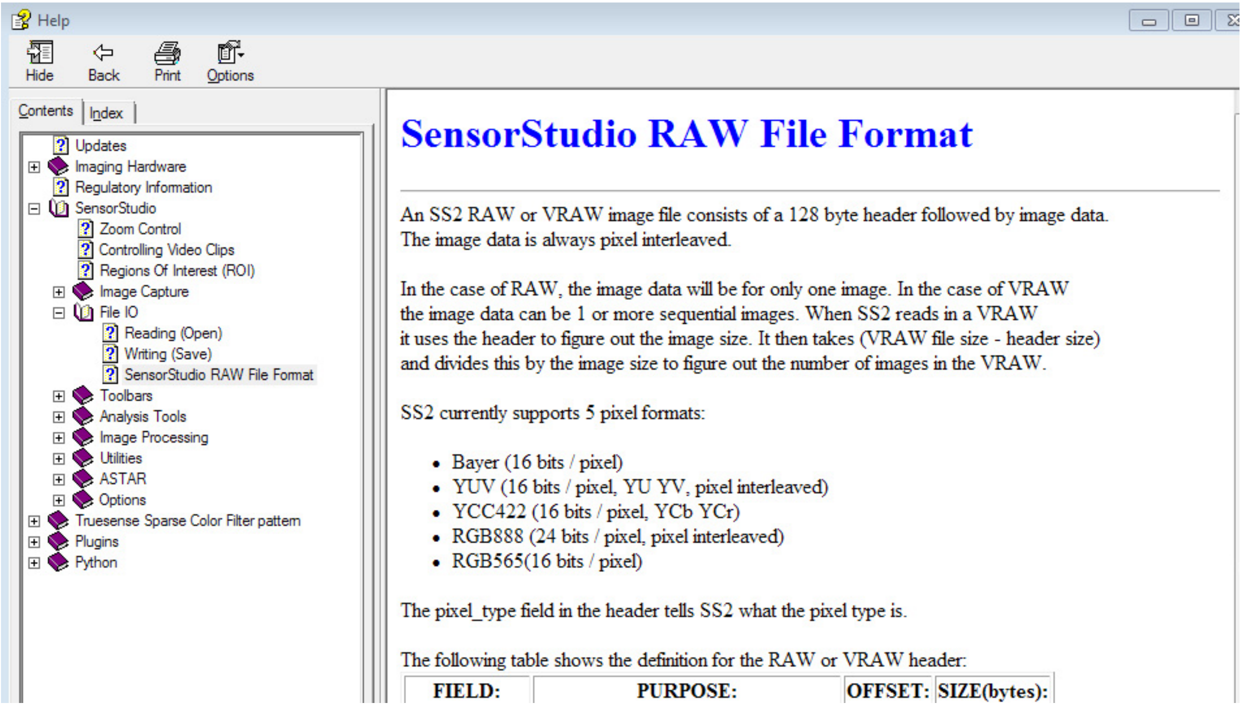
Setting the proper exposure during image capture for the Sparse CFA is important for achieving the full dynamic range. The panchromatic pixel is twice as sensitive as the color pixels and will saturate first. Therefore, the panchromatic pixels should be used to determine the exposure. For example, setting the exposure on an 18% gray for a 12-bit imager with a whitepoint of 106% and a dark offset of 30 should give a raw digital code value of 720 according to the following calculations:

$$\text{code value} = \left(2^{(\text{number of bits})} - \text{dark offset}\right) \times \left(\frac{0.18}{\text{whitepoint}}\right) + \text{dark offset} \quad \text{(eq. 1)}$$

$$\text{code value} = \left(2^{(12)} - 30\right) \times \left(\frac{0.18}{1.06}\right) + 30 \quad \text{(eq. 2)}$$

SENSOR STUDIO AND RAW IMAGE FILE FORMAT

The description for the raw file format can be found in the Help Menu as shown below.



SensorStudio RAW File Format

An SS2 RAW or VRAW image file consists of a 128 byte header followed by image data. The image data is always pixel interleaved.

In the case of RAW, the image data will be for only one image. In the case of VRAW the image data can be 1 or more sequential images. When SS2 reads in a VRAW it uses the header to figure out the image size. It then takes (VRAW file size - header size) and divides this by the image size to figure out the number of images in the VRAW.

SS2 currently supports 5 pixel formats:

- Bayer (16 bits / pixel)
- YUV (16 bits / pixel, YU YV, pixel interleaved)
- YCC422 (16 bits / pixel, YCb YCr)
- RGB888 (24 bits / pixel, pixel interleaved)
- RGB565(16 bits / pixel)

The pixel_type field in the header tells SS2 what the pixel type is.

The following table shows the definition for the RAW or VRAW header:

FIELD:	PURPOSE:	OFFSET:	SIZE(bytes):

Figure 1. Portion of Sensor Studio Raw File Format Help Menu

File types supported for single capture image are

- 16-bit RAW (Native to Sensor Studio)
- 16-bit PNG
- Processed BMP
- Processed JPEG

The Processed BMP and JPEG are 8-bit full color images.

Sensor Studio File Format Bit Depth

Although Sensor Studio does support 16-bit images, the Sparse CFA processing path only takes 12-bit data (unpacked 12-bit encoded as 16-bit). If you are unsure of the bit depth of a Sparse CFA image, use the "Pixel Code Values" to determine the bit-depth as shown in Figure 2.

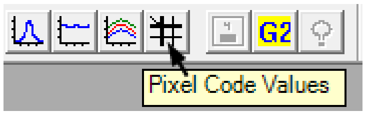


Figure 2. Pixel Code Values Button

An example raw scene is shown in Figure 3 to demonstrate the use of the Pixel Code Values function.

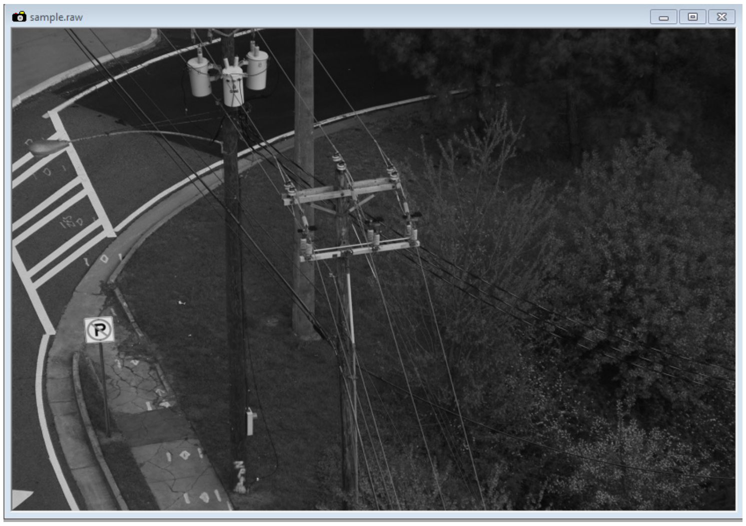
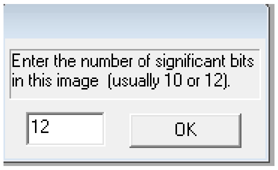


Figure 3. Example Raw Scene Image as it Should Appear

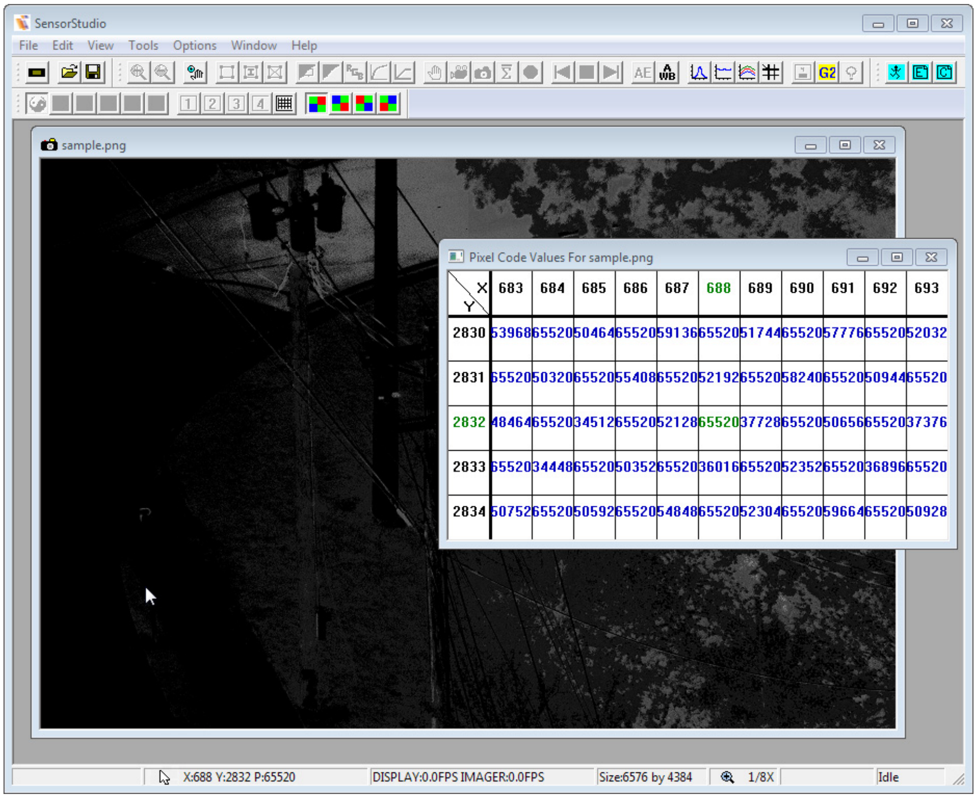
When opening a raw image that was not captured natively in Sensor Studio, the pop-up shown in Figure 4 is displayed. This gives the users the opportunity to specify the bit depth of the image. In this example, the user has selected a bit depth of 12.

In Figure 5, the raw image opens but does not appear normal. The “no parking” sign should be white. A potential cause of this effect is that Sensor Studio is not processing the image file using the correct bit depth. By using the Pixel Code Values tool, it is shown that the data is 16-bit since the values are in the range of 65535. In this case, the 12-bit data has been provided to Sensor Studio with the 12-bit data in the twelve upper bits and padded with zeros in the lower bits.


Enter the number of significant bits in this image (usually 10 or 12).

12

Figure 4. Pop-up



X / Y	683	684	685	686	687	688	689	690	691	692	693
2830	53968	65520	50464	65520	59136	65520	51744	65520	57776	65520	52032
2831	65520	50320	65520	55408	65520	52192	65520	58240	65520	50944	65520
2832	48464	65520	34512	65520	52128	65520	37728	65520	50656	65520	37376
2833	65520	34448	65520	50352	65520	36016	65520	52352	65520	36896	65520
2834	50752	65520	50592	65520	54848	65520	52304	65520	59664	65520	50928

X:688 Y:2832 P:65520 | DISPLAY:0.0FPS IMAGER:0.0FPS | Size:6576 by 4384 | 1/8X | Idle

Figure 5. Sample Image Pixel Code Values

In order to convert the data to 12-bit, the pixel values must be shifted to the right by 4 positions (divide by 16). This can be done within Sensor Studio as follows:

- Go to the Tools Menu and Select “Python Command Window”
- Use the BitShiftImage() Function to Convert to 12 bit (Refer to Help:Python:Extensions:Read or Modify Pixel Data)
- Do File−>Save Image As and Save it as a RAW such that the Original PNG is Not Disturbed
- Open Up the New Raw. It Should Display Normally and will be 12 bit
- Color Process Using the Device Used for Capture, for Example, “KAI−29050 SPARSE CFA Setup”

If the raw image data is less than 12-bit, Sensor Studio will automatically convert it to the appropriate bit depth with all of the data in the lower bits. For an 8-bit raw file, the upper bits would be converted to all zeros and the data would all be in the lower 8 bits.

SPARSE CFA ‘C’ Code Bit Depth

If the SPARSE CFA ‘C’ code is being used as a standalone package (not with Sensor Studio), the data needs to be 12-bit data (unpacked 12-bit encoded as 16-bit). The data is expected to be in the lower bits.

Converting Raw Data File Formats

Third party packages such as ImageJ (search internet for “ImageJ” for download) can be used to convert different raw data file formats into a format that is compatible with Sensor Studio and the Sparse CFA processing.

An example for using ImageJ is given below. For raw files, open the file as Raw, then convert to Image: Type: 16-bit, and then save as PNG.

ImageJ can also be used to convert DNG file-type images to PNG.

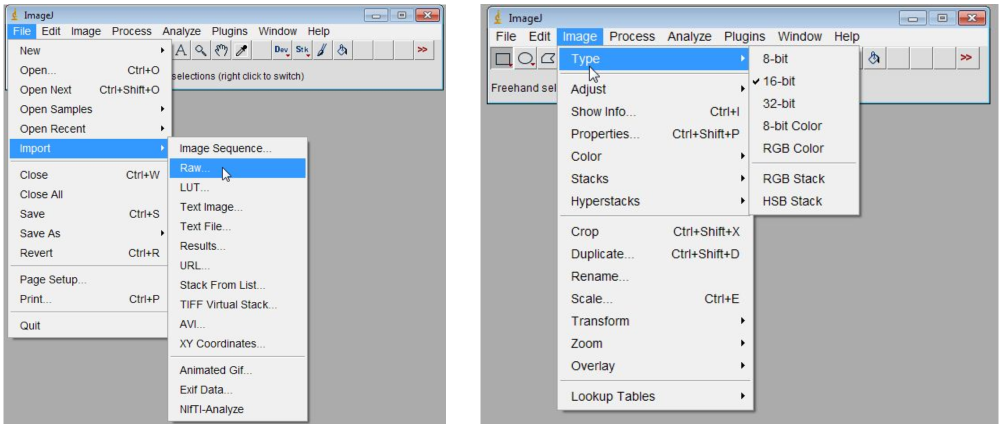


Figure 6. Image J Menus

SENSOR STUDIO IMAGE PROCESSING OPTIONS

The Image Processing Options has preset setups for various sensors. Choose the family (“KAI 5.5” for devices in the 5.5-micron interline transfer CCD Family, or “KAI 7.4” for devices in the 7.4-micron interline transfer CCD Family) then select the sensor or group that has the Sparse CFA.

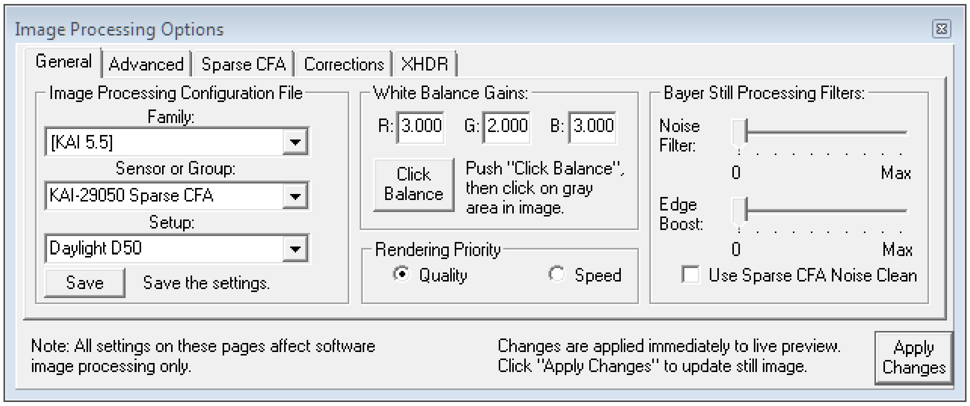


Image Processing Configuration File

Family: [KAI 5.5]

Sensor or Group: KAI-29050 Sparse CFA

Setup: Daylight D50

Save the settings.

White Balance Gains:

R: 3.000 G: 2.000 B: 3.000

Push "Click Balance", then click on gray area in image.

Rendering Priority

● Quality ○ Speed

Bayer Still Processing Filters:

Noise Filter: 0

Edge Boost: 0

□ Use Sparse CFA Noise Clean

Note: All settings on these pages affect software image processing only.

Changes are applied immediately to live preview. Click "Apply Changes" to update still image.

Figure 7. Image Processing Options

Selecting the Dark Offset

On the sensor itself, there are shielded pixels surrounding the active pixel region. These can be used to set the dark offset or dark floor for the sensor. Enter the dark offset value in the Image Processing Option: Advanced: Dark Offset. Typical values range from 20–40 for a 12-bit image.

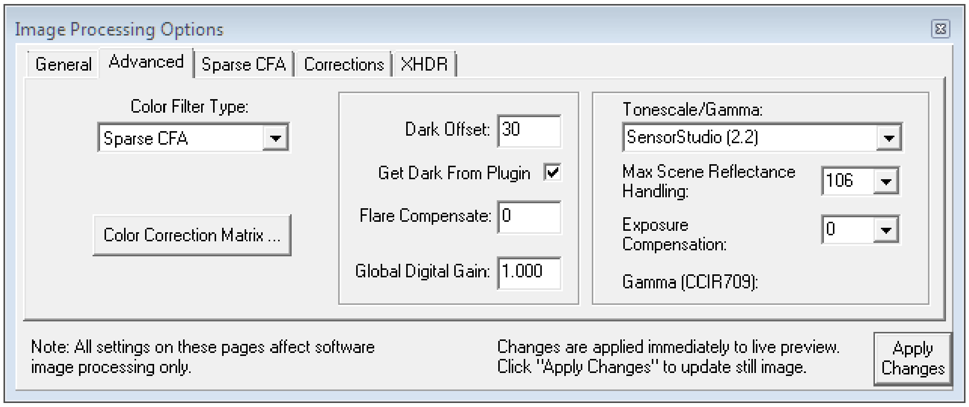


Image Processing Options

General | Advanced | Sparse CFA | Corrections | XHDR

Color Filter Type: Sparse CFA

Color Correction Matrix ...

Dark Offset: 30

Get Dark From Plugin ☑

Flare Compensate: 0

Global Digital Gain: 1.000

Tonescale/Gamma: SensorStudio (2.2)

Max Scene Reflectance Handling: 106

Exposure Compensation: 0

Gamma (CCIR709):

Note: All settings on these pages affect software image processing only.

Changes are applied immediately to live preview. Click "Apply Changes" to update still image.

Apply Changes

Figure 8. Dark Offset

Selecting the Sparse CFA Pattern

In the Sparse CFA pattern, the panchromatic pixels are interspersed as a checkboard pattern. Like colors are averaged together in the processing. Therefore, the underlying color pattern is still a Bayer-like pattern. The underlying color pattern may be arranged in any of the configurations shown below. Sensor Studio provides options to allow the user to process full or cropped images. The Sparse CFA pattern is chosen by clicking one of the buttons.

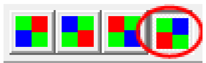


Figure 9. Set GBRG Button

The default for the Sparse CFA pattern in the setups is usually the one shown circled above. However, if the image has been cropped, one of the other buttons may be the correct configuration. Unclick the button before trying a different pattern.

White Balance Correction

The White Balance Gains default values may only be initial values. Sensor Studio provides a “click balance” function for the active image to achieve a more accurate white balance by selecting a ‘known’ medium grey region within the image and then setting the R,G,B gains to make this region ‘grey’. For natural scenes, using a medium gray works best – such as blacktop-road surfaces. For a properly exposed image, the green pixel gain after white balance is typically around 2 because it is half the sensitivity of the panchromatic pixels.

Luma Noise Cleaning Option in Image Processing Options Menu

The Luma Noise injects high resolution panchromatic data into the image. When it is off (setting high and low equal to 0), the image processing path ONLY uses cleaned and upsampled LOW resolution pan data. This is sometimes useful if the analog gain is set above 28 and the image is extremely photon starved.

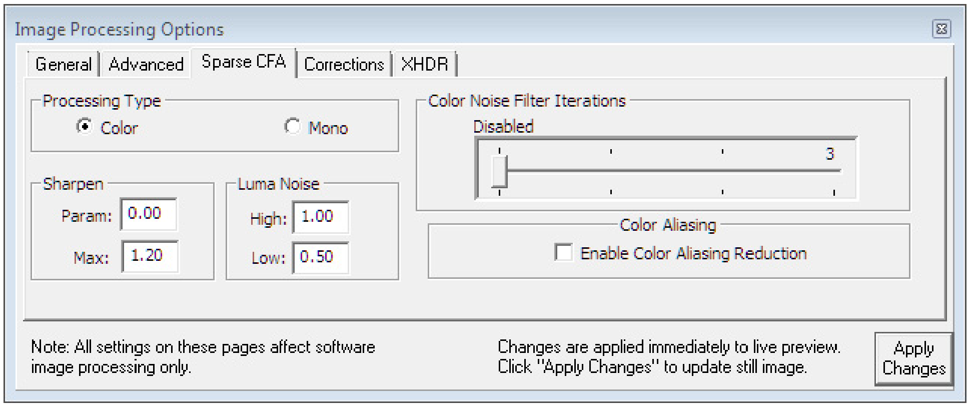


Image Processing Options

General | Advanced | Sparse CFA | Corrections | XHDR

Processing Type: ● Color ○ Mono

Sharpen: Param: 0.00 Max: 1.20

Luma Noise: High: 1.00 Low: 0.50

Color Noise Filter Iterations: Disabled 3

Color Aliasing: □ Enable Color Aliasing Reduction

Note: All settings on these pages affect software image processing only.

Changes are applied immediately to live preview. Click "Apply Changes" to update still image.

Apply Changes

Figure 10. Luma Noise

The default setting is High = 1, Low = 0.5. This adds a Blend percentage of the difference between the interpolated high resolution pan pixel and the upscaled, cleaned low resolution pan pixel. The amount that is added in is based upon the High and Low values and a gamma type curve.

The default settings of High = 1, Low = 0.5 give the most pleasing results in general. The function is called Luma Noise because it is the difference between the cleaned low res pan pixel and the uncleaned pan pixel. But it really is detail and noise. The equation also tends to weight the blend by the gamma curve so that the highlights will get more and the darks will get less. By setting High = Low, then the gamma curve weighting is not used and the whole image will get a Blend percentage based on only the Low setting.

OPTIONS FOR IMPROVING PROCESSING SPEED FOR REAL-TIME APPLICATIONS

As mentioned in the overview sections, the code provided has not been optimized for speed for real-time applications. This section gives a few suggestions for improving the processing speed.

Turn Off Color Noise Cleaning

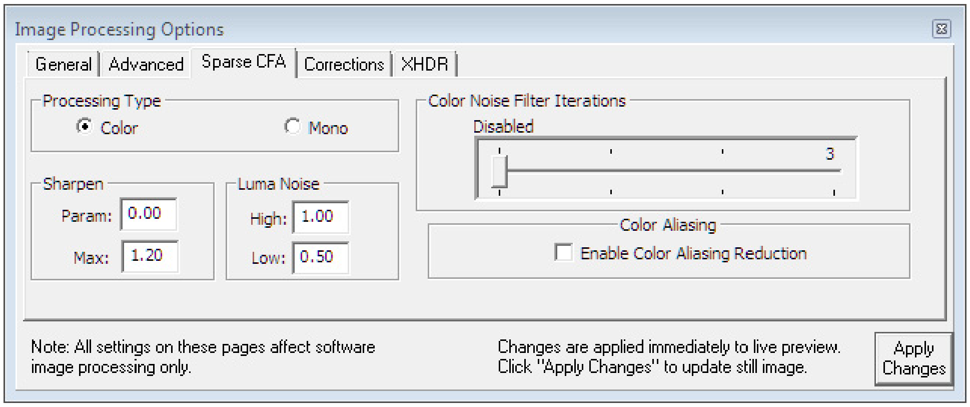
The color noise cleaning step is the slowest step in the processing path. The noise cleaning step can be iterated several times. It also requires the largest chunk of memory. The color noise cleaning step may not be necessary for all applications. It also can be turned off to allow a faster preview.

The color noise cleaning tables are derived from the analog gain that was used during capture. The analog gain is typically increased in order to achieve full dynamic range in night or extremely low light captures. If the analog gain is not increased, less color noise cleaning should be required.

The color noise cleaning can be skipped in Sensor Studio by setting the color noise filter iterations to Disabled in the Image Processing Options Menu.

The color noise cleaning step can be skipped in the ‘C’ code at skip_noise instruction by setting all of the iColor Radius parameters to 0 as shown below

```
skip_noise = ((pm->iColorRadius0 == 0) && (pm->iColorRadius1 == 0) && (pm->iColorRadius2 == 0));
```



Processing Type: ● Color ○ Mono

Sharpen: Param: 0.00 Max: 1.20

Luma Noise: High: 1.00 Low: 0.50

Color Noise Filter Iterations: Disabled 3

Color Aliasing: ☐ Enable Color Aliasing Reduction

Note: All settings on these pages affect software image processing only.

Changes are applied immediately to live preview. Click "Apply Changes" to update still image.

Figure 11. Color Noise Filter Iterations

Turn Off Sharpening Algorithm

The sharpening algorithm can be turned off or skipped in rev2.0 ‘C’ code SparseCFA_c_rev2.0 in order to improve processing speed. This can be done by commenting out the “Sharpening” call or by adding a skip similar to what is done for the color noise algorithm.

Note that the sharpening algorithm cannot be skipped in rev1.0 of this code because the Alg_HighResPanBlending function was performed within the sharpening code for the initial version and disabling sharpening will only produce a low resolution image.

High Resolution Mono Processing

For a high resolution monochrome image (only using pan pixels), only a few of the routines need to be called. This is the fastest processing path using code revision 2.0. In SparseCFA_c_rev2.0, the routines that should be called would be

- Dark Subtract − Alg_SubtractDarkFloor
- White Balance − Alg_WhiteBalance
- Interpolate − Alg_InterpolatePan
- Tone Map − pToneScale

Low Resolution Color Preview

The fast preview mode in Sensor Studio uses a subset of the commands which may be sufficient for real-time viewing for most applications. Here is the list of subroutines to call.

- Dark Subtract − Alg_SubtractDarkFloor
- White Balance − Alg_WhiteBalance
- Interpolate − Alg_InterpolatePan
- Low Res Color − Alg_GenerateLow
- Tone Map − Apply pToneScale

Fast Low Resolution Color Preview

A low resolution color preview can be created by using the routines that digitally bin the color pixels and then using standard Bayer processing. In SparseCFA_c_rev2.0, the routines that should be called would be:

- Dark Subtract − Alg_SubtractDarkFloor
- White Balance − Alg_WhiteBalance
- Low Res Color − Taking the Image from Alg_GenerateLow (dataGenerateLow_lBayer) and
- Tone Map − Running it through the pToneScale to Make the Final Image

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Literature Distribution Center for ON Semiconductor
19521 E. 32nd Pkwy, Aurora, Colorado 80011 USA
Phone: 303−675−2175 or 800−344−3860 Toll Free USA/Canada
Fax: 303−675−2176 or 800−344−3867 Toll Free USA/Canada
Email: orderlit@onsemi.com

N. American Technical Support: 800−282−9855 Toll Free USA/Canada
Europe, Middle East and Africa Technical Support:
Phone: 421 33 790 2910

ON Semiconductor Website: **www.onsemi.com**

Order Literature: http://www.onsemi.com/orderlit

For additional information, please contact your local Sales Representative