ANALOG DEVICES
Technical notes on using Analog Devices DSPs, processors and development tools
Visit our Web resources http://www.analog.com/ee-notes and http://www.analog.com/processors or e-mail processor.support@analog.com or processor.tools.support@analog.com for technical support.

Interfacing Red/Clear Sensors to ADSP-BF609® Blackfin Processors

Contributed by Gopal Karanam

Rev 1 – May 28, 2013

Introduction

The ADSP-BF609 dual-core Blackfin® processor series is optimized for embedded vision and video analytics applications, such as automotive advanced driver assistance systems (ADAS), machine vision and robotics for industrial manufacturing systems, and security and surveillance system analytics. This Engineer-to-Engineer Note discusses the advanced features available for interfacing application specific cameras to the ADSP-BF609 Blackfin processor.

Vision cameras have photosensors that convert light intensity into an electronic signal. Typical photosensors have little or no sensitivity to wavelength of incident light and therefore cannot separate color information. In order to capture color information, a mosaic of tiny color filters is placed over the pixel sensors of an image sensor.

One of the most common color filters used in consumer cameras is the Bayer filter, which gives information about the intensity of light in red, green and blue wavelength regions. One possible arrangement of the color filter is shown in Figure 1. The raw image data captured by the image sensor is then converted to full-color image by a demosaicing algorithm with intensities of all three primary colors (red, green, blue) represented at each pixel. Naturally, the demosaicing algorithm has to be tailored for each type of color filter.

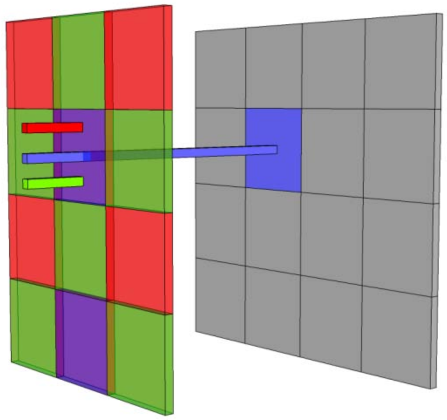


Figure 1. Bayer Color Filter Array (CFA)

Vision based ADAS use multiple cameras to capture the scene around the car to assist during the driving process. Typical machine vision algorithms use only the intensity of the light (Luma). Hence, sensor manufactures produce monochrome sensors exclusively for ADAS applications. But, in order to meet stringent quality requirements the application needs access to some color information as well. One of the key color information is the red channel, which will help localize the region of interest (ROI) of the image, such as traffic light, car rear-light, etc. For example, Traffic Sign Recognition (TSR) applications can identify possible locations of red colored speed limit signs in the captured image.

Hence, sensor manufacturers have designed Red/Clear (RCCC) color filter array specifically for automotive use. Unlike Bayer sensors, RCCC sensors use clear filters instead of the blue and the two green filters in the 2x2 pixel neighborhood. The red filter remains the same. A “clear filter” is the same concept as monochrome

Copyright 2013, Analog Devices, Inc. All rights reserved. Analog Devices assumes no responsibility for customer product design or the use or application of customers' products or for any infringements of patents or rights of others which may result from Analog Devices assistance. All trademarks and logos are property of their respective holders. Information furnished by Analog Devices applications and development tools engineers is believed to be accurate and reliable, however no responsibility is assumed by Analog Devices regarding technical accuracy and topicality of the content provided in Analog Devices Engineer-to-Engineer Notes.

sensors. Hence, the output generated by RCCC sensors is almost as detailed as a monochrome output and still provides the red color information as shown in Figure 2.

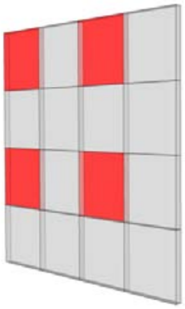


Figure 2. Red/Clear Color Filter Array

De-Mosaicing Algorithm

The red pixels form a quarter (width/2 x height/2) of the full resolution of the sensor. On the other hand, the clear pixels form $3/4^{th}$ of the sensor resolution. In order to generate the full resolution clear or monochrome pixels, a de-mosaicing algorithm has to generate the missing "C" pixels at "R" locations, commonly referred to as the red pixel substitution.

The quality of the algorithm can be estimated by simulating a Red/Clear color filter, applying the de-mosaicing algorithm and then comparing against the reference image. A test image shown in Figure 3 with lots of detail in color, texture and edges is chosen for analysis.

Figure 3. Test Image[2]

The image has red, green, and blue values for each pixel and is converted to reference monochrome image using Equation 1:

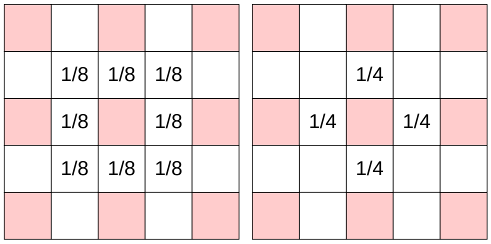
$$Y = 0.59 * G + 0.3 * R + 0.11 * B$$

Equation 1. Conversion to Monochrome Image

A Red/Clear color filter is simulated by replacing 1(one) luma pixel out of 2x2 pixel region in the reference monochrome image with a red pixel. The demosaicing algorithm will then generate the missing clear pixel. The resulting image is compared visually as well as using peak-signal-to-noise ratio (PSNR) metric.

Legacy De-Mosaicing Method

The clear pixel is generated by a convolution operation. The filter kernel is only applied at "R" locations to compute the missing "C" pixel. A popular method to compute the missing clear pixel is to use the mean of the neighboring clear pixels. The filter kernel for such an algorithm is shown in Figure 4.



🟥		🟥		🟥
	1/8	1/8	1/8	
🟥	1/8	🟥	1/8	🟥
	1/8	1/8	1/8	
🟥		🟥		🟥

🟥		🟥		🟥
		1/4		
🟥	1/4	🟥	1/4	🟥
		1/4		
🟥		🟥		🟥

Figure 4. 8-Pixel and 4-Pixel Averaging Kernel

Proposed Method

The legacy methods discard the red value at the red pixel locations. It has most valuable information and can be used to correct the mean value of the computed clear pixel. The estimate of the red pixel can be computed from the neighboring red pixels, and the difference with the center value represents the luminance change at the location. A fraction of this estimate error can be added to the mean of the clear pixels as described in Figure 5.

$$\begin{bmatrix} & & & & \\ & & 1/4 & & \\ & 1/4 & & 1/4 & \\ & & 1/4 & & \\ & & & & \end{bmatrix} + \zeta * \begin{bmatrix} & & -1/4 & & \\ & & & & \\ -1/4 & & 1 & & -1/4 \\ & & & & \\ & & -1/4 & & \end{bmatrix}$$

Figure 5. 4-Pixel Averaging Kernel & Correction Term

The parameter zeta (ζ) controls the amount of correction. Zeta is varied from 0 (4-clear pixel averaging kernel) to 1 in steps of 0.05. The PSNR of the resulting images is plotted in MATLAB as shown in Figure 6.

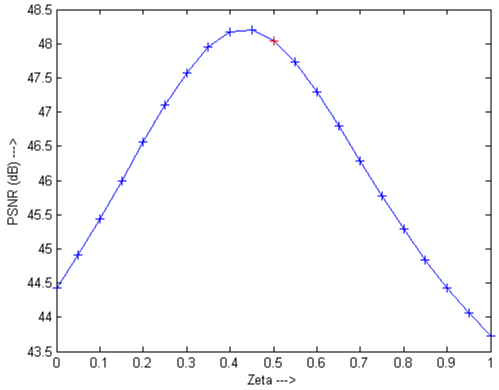


Figure 6. PSNR Plot of De-Mosaiced Images

It is clear from the graph that adding a correction term improves PSNR significantly and it peaks around zeta equals 0.5. The optimal kernel is shown in Figure 7.

		-1/8		
		1/4		
-1/8	1/4	1/2	1/4	-1/8
		1/4		
		-1/8		

Figure 7. Optimal Red/Clear De-Mosaicing Kernel

Table 1 demonstrates that adding the correction term results in a PSNR improvement of 3.6dB:

Filter Kernel	PSNR (dB)
Mean (1st kernel in Figure 4)	43.4
Mean (2nd kernel in Figure 4)	44.4
Mean + Correction term (Figure 7)	48

Table 1. PSNR table for different demosaic kernels

The visual quality improves as shown in Figure 8.

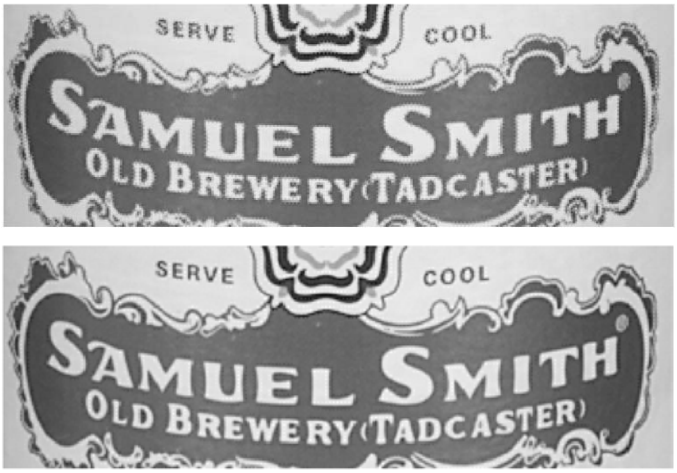


Figure 8. Section of De-Mosaiced image using mean kernel (Top) & optimal kernel (bottom)

Implementation of the De-Mosaicing Algorithm on the ADSP-BF609 Blackfin Processor

The Pipelined Video Processor (PVP) hardware engine on ADSP-BF609/8 derivatives[5] is designed to process images from Red/Clear color filter sensors.

The PVP contains four Convolution blocks with each of them supporting any arbitrary 5x5 coefficient matrix. One of the Convolution blocks, CNV1, provides a unique feature, which performs convolution operation, shift and saturate only at the red pixel location. The convolution and normalization coefficients are freely programmable.

The following parameters are proposed for the discussed de-mosaicing algorithm:

$$Coeff_{CNV1} = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ -1 & 2 & 4 & 2 & -1 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix}$$

The output is normalized by setting the `PVP_CNV1_CTL.SHIFT` value to 3. If the output needs to be clipped between a particular range, then the Threshold-Histogram-Compression (THCx) block can be used.

The quarter resolution red image can be separated using the Input Formatter IPF0. Color separation grants concurrent access to multiple color components. It is enabled by setting the `PVP_IPF0_CTL.BFRMT0` bit. Color separation mode supports two arrangements of color filter arrays as controlled by `PVP_IPF0_BFRMT1` bit. The two types are shown in Figure 9.

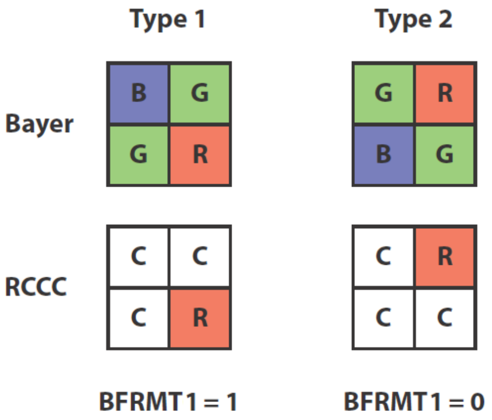


Figure 9. Color Separation for Type 1 & 2

In color separation mode, IPF0 output port 2 outputs the red pixels.

Naturally, the PVP performs the operation on in-coming data real time, without consuming any MIPS (Millions of Instructions Per Second) on the Blackfin cores, available for further signal processing tasks.

References

[1] *Wikipedia (http://en.wikipedia.org/wiki/Color_filter_array).* May 2013.

[2] *Sigma SD1 Merrill Thumbnails (http://www.imaging-resource.com/PRODS/SD1/SD1THMB.HTM). May 2013.*

[3] *H.S. Malvar, L.-W. He, and R. Cutler, High-quality linear interpolation for demosaicing of Bayer-patterned color images,* IEEE Int Conf (ICASSP '04), Proceedings on Acoustics, Speech, and Signal Processing 3, 2004, 485-8.

[4] *ADSP-BF60x Blackfin Processor Hardware Reference.* Rev 0.5, February 2013. Analog Devices, Inc.

[5] *ADSP-BF606/607/608/609 Blackfin Dual Core Embedded Processors Data Sheet.* Rev PrG, April 2013. Analog Devices, Inc.

Document History

Revision	Description
Rev 1 – May 28, 2013 *by Gopal Karanam*	Initial release.