Real-time, color image barrel distortion removal

Henryk Blasinski *, Wei Hai†, Frantz Lohier ‡
Logitech Inc.
6505 Kaiser Drive
Fremont, CA
* h.blasinski@stanford.edu, † wei@plantronics.com, ‡ frantz@lohier.com

***Abstract*—This paper describes a new hardware architecture for barrel distortion correction in a color video stream. Such distortion is omnipresent in images acquired with a large field of view optics: wide-angle or fish-eye lenses. The designed platform is composed of a standard image sensor, a USB video class ASIC chip, a low cost FPGA and a SDRAM memory chip. Image processing algorithms are implemented in the FPGA, which is inserted between the sensor and the ASIC. The FPGA is connected to an external SDRAM in which a frame buffer is implemented. Barrel distortion is modeled using a polynominal relationship between corrected and distorted image spaces. Combinatorial logic circuit at the frame buffer output validates correct ordering of luminance and chrominance bytes in the data stream. The proposed design is capable of removing geometric distortion from 640 $\times$ 480 pixel images at the rate of 30 frames per second. Colors in reconstructed images are within ΔE = 2 from the originals in the CIELab color space.**

I. Introduction

Digital cameras have become ubiquitous in today's world. They are present not only as devices dedicated to image acquisition such as single-lens reflex cameras, but they have migrated into many other objects such as mobile phones, computers, tablets, or even cars. Along with popularization came novel applications, for example enhanced reality or telepresence. However, the mid-range field of view (FOV) of current cameras, typically about 60°, significantly limits the user experience associated with these applications. Note that 60° of the visual field covers approximately $1/3$ of the human visual system FOV. This problem may be partially solved by using special optics, such as wide-angle or fish-eye lenses. Unfortunately, these lenses introduce geometric distortion into the image, which seriously impacts the perceptual image quality. Barrel distortion affects straight lines from the real world and represents them as curves in the image.

Past research into barrel distortion removal focused on two aspects. The first one described mathematical models approximating the nature of this phenomenon and proposing image restoration algorithms [1]–[5]. Computational complexity was rarely analyzed though. The second aspect addressed the issue of efficient, real-time computation on designated hardware platforms [6]–[9]. Implementations varied in terms of logic requirements or frame rates often achieving good performance; however, only grayscale images were analyzed.

H.B. is currently with Stanford University, CA. W.H. is currently with Plantronics Inc., CA, F.L. is currently with Kudelski Group.

In many commercial applications, color is often represented in the luminance and subsampled chrominance format, for example YUV 4:2:2. This means that the spatial sampling frequency of color information is two times lower, thus chrominance data is shared between neighboring pixels. Such a solution significantly lowers the bandwidth requirement, but complicates image manipulation. Recently a real-time barrel distortion system for color images was proposed [10], but it processed very small, QQVGA (160×120) images that severely limited any practical use.

This paper is organized as follows. Section II presents a high level overview of the distortion model used. Section III describes the proposed architecture; it is followed by experimental results discussed in section IV. Conclusions are presented in section V.

II. Distortion removal

In this paper we are modeling the barrel distortion as having two components: radial, affecting the distance R of a given point from the image centre, and tangential, which affects the angle θ between the radius and the reference direction [11]. In general, the relationship between corrected (R_c, θ_c) and distorted (R_d, θ_d) parameters can be expressed by m and n-th order polynominals:

$$\theta_c = \sum_{k=0}^{m} a_k \cdot \theta_d^k = p(\theta_d) \tag{1}$$

$$R_c = \sum_{k=0}^{n} b_k \cdot R_d^k = q(R_d) \tag{2}$$

Direct implementation of this approach in hardware can be problematic due to polar to rectangular conversion. For this reason, the coefficient method proposed by [10] was implemented. In brief, this method assumes that the radial distortion is negligible ($\theta_d = 0$), and relates distorted and corrected coordinates of image pixels to a coefficient C_f. The value of this coefficient is approximated with a n-th order polynominal function of the square of the pixel distance from the image center (i.e. $x^2 + y^2$). We refer to the original paper for more details. In this implementation, a 2^{nd} order polynominal was used; its coefficient values are given in Table I.

Numerical values taken from [10]

TABLE I
MAPPING ENGINE PARAMETERS.

Lens type	Resolution	Scaling		2^{nd} order polynominal		
		m	n	a	b	c
Fish-eye	QQVGA	8	-10	0.2272	-10.863	239.24
	VGA	8	-10	0.0016	-0.8979	239.24

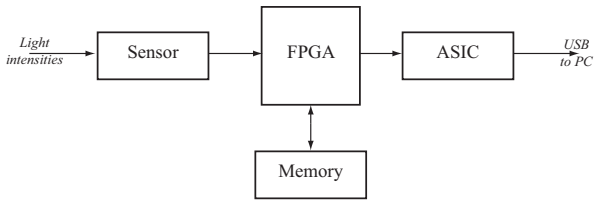


Fig. 1. Overall system architecture.

III. SYSTEM ARCHITECTURE

To facilitate verification, the system was designed to be implemented in a Field Programmable Gate Array (FPGA) chip. The FPGA was inserted between the acquisition sensor, and the USB Video Class ASIC chip. In this way, standard components could have been used, and the presence of the additional FPGA circuit would be transparent. The external memory chip was accessible by the FPGA only, as presented in Fig. 1.

The nature of the barrel distortion removal algorithm requires an image buffer, which was implemented in the external memory. Due to low cost, in high volume commercial applications dynamic memories are usually chosen. While they provide large storage space and high bandwidth, the latter can be significantly reduced when accessing memory cells in a non contiguous fashion. Such memory access schemes are characteristic of geometric distortion correction algorithms and become an issue in real time applications.

A. Full-frame DDR2 SDRAM buffer

The DDR2 SDRAM does not allow for simultaneous read/write operations, thus making it necessary to multiplex access time, which can be done on a per line basis. In order to access the memory efficiently, it is necessary to implement a layer of 'cache' between the incoming and outgoing data and the memory (Fig. 2). Incoming image data is temporarily stored in a *Data in FIFO*, as each consecutive word represents the next pixel, a simple counter is sufficient to compute addresses in the frame buffer. While input data is being written into the memory, addresses corresponding to output pixels are computed and stored in the *Address FIFO*. First, the coordinates in the corrected image space are generated, then they are mapped onto the distorted image space as described in Sec. II. Distorted coordinates serve as the basis for buffer address computations. In parallel to addresses, parity bits are also calculated and stored into the *Parity FIFO*. A parity bit indicates whether there is a mismatch between the parities of the x coordinate in the distorted and corrected image spaces. Once one line is processed this way, the multiplexer switches

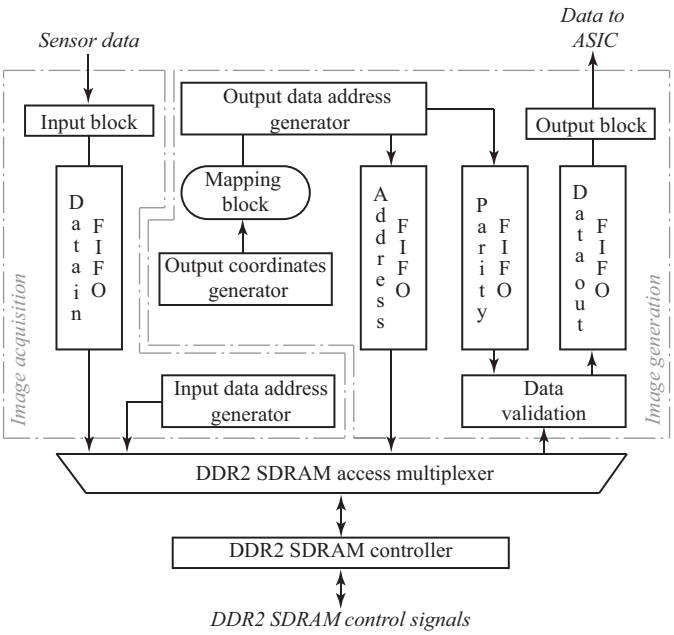


Fig. 2. DDR2 SDRAM frame buffer architecture. The memory access multiplexer is preceded by a layer of FIFOs constituting a cache for incoming and outgoing image data.

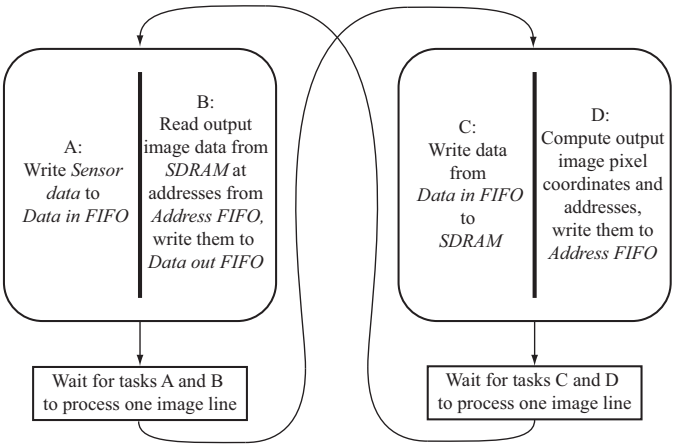


Fig. 3. High level overview of the finite state machine. One state (task B) corresponds to SDRAM read, the other (task C) to SDRAM write.

to the memory read operation, generating output image line data. At the same time, incoming pixel values are written into the *Data in FIFO* (Fig. 3).

B. DDR2 SDRAM Color restoration

To achieve FPGA circuit transparency, the data sent to the ASIC chip has to follow the same pattern as the incoming sensor data. In a YUV 4:2:2 encoding, for a given image line, luminance data bytes are interleaved with chrominance data: Cr and Cb. The correct byte pattern may be as follows: $Y_1, Cr_1, Y_2, Cb_1, Y_3, Cr_3, Y_4, Cb_3, ...$ If chrominance data is switched, or just one component is repeated, color distortion occurs. An example of an erroneous byte ordering would be: $Y, Cb, Y, Cb, Y, Cb, ...$

In order to restore the original YUV 4:2:2 colors, redundant data obtained during burst read operations was used. In a burst transaction a memory is fed a single address, in response to which it fetches data corresponding to that particular address and n consecutive ones, where n denotes the burst length. Assume that consecutive addresses in the memory encode image pixels in a single line. This means that issuing a memory read request with an address A corresponding to pixel coordinates (x, y), will produce pixel data from coordinates: $(x, y); (x + 1, y); (x + 2, y); ...; (x + n, y)$. Since any two neighboring pixels in a single line store all chrominance information, it is now possible to assure proper order of color bytes.

These operations, performed in the *Data validation* block, are schematically presented in Fig. 4. Although it is assumed that the data bus is 32 bits wide, generalizations to other widths can easily be made. Since just the first memory word has both chrominance values, the remaining $n - 1$ words can be discarded. The decision as to which color information byte to use is contingent upon the value of the parity bit stored earlier in the *Parity FIFO*.

Parity values are obtained by comparing the x coordinate of the distorted and corrected images. If parities are the same, then the parity bit is set to '0' and the color information read from that particular pixel matches the order in the output bitstream. In the opposite case, the parity bit is set to '1' and read chrominance byte ordering is reversed.

IV. Results

A. Experimental setup

The architecture was implemented using a Xilinx Spartan 3 700AN FPGA with about 13,000 logic cells and 20 SRAM memory blocks 18 kbits each. This FPGA was available on a starter board equipped with a 32M $\times$ 16 bit DDR2 SDRAM block in which the frame buffer was realized. The circuit was interfaced with the FPGA using a SDRAM controller intellectual property (IP) block provided by Xilinx. A 2 megapixel 1/3.2 inch MT9D111 sensor by Aptina and a USB Video Class ASIC chip constituted the remaining blocks of the system.

B. Buffer implementation

Using a full frame dynamic memory buffer, it was possible to process VGA (640×480) images at the rate of 30 frames per second (fps). The design was balanced in terms of resources utilization, which is detailed in Table II. In general, the architecture used 28% of logic, 25% of embedded memory, and 25% of multipliers, available in the FPGA. The largest block was the SDRAM controller that had to operate at high frequencies, as a result, it required many pipelining stages. The presented solution is considerably smaller than other designs. For example [6] used approximately 18,000 logic elements, however their system was optimized for performance rather than compact size.

TABLE II
Spartan-3 700AN resources utilization.

Module	Logic slices		Memory blocks		Multipliers	
	Used	[%]	*Used*	[%]	*Used*	[%]
Input block	50	0.8%	0	0	0	0
Data in FIFO	158	2.7%	1	5%	0	0
Input addr. gen.	48	0.8%	0	0	0	0
DDR2 multiplexer	109	1.8%	0	0	0	0
DDR2 controller	472	8.0%	0	0	0	0
Output coord. gen.	74	1.6%	0	0	0	0
Mapping	166	2.8%	0	0	10	50%
Output addr. gen.	12	0.2%	0	0	0	0
Address FIFO	169	2.8%	2	10%	0	0
Parity FIFO	146	2.5%	1	5%	0	0
Data validation	24	0.4%	0	0	0	0
Data out FIFO	125	2.1%	1	5%	0	0
Output block	101	1.7%	0	0	0	0
Total	**1657**	**28.1%**	**5**	**25%**	**10**	**50%**

C. Color restoration

Appropriate selection of chrominance bytes, as described in section III-B, performs a very visually pleasing color restoration. Colors were measured by acquiring an image of a standard Macbeth color checker in three scenarios: original image without barrel distortion removal, image with barrel distortion removed but without color restoration and finally an image with both color and distortion corrected. Table III presents average RGB values of four different, uniformly colored patches shown in Fig. 5. This table also provides average perceptual differences ΔE in the CIELab color space between the original image and images without and with color restoration; ΔE_{nr} and ΔE_r respectively. RGB to CIELab conversion was computed with the ISET software where typical LCD display characteristics were selected [12]. Very small ΔE values confirm that perceptually there is almost no difference between colors in original and reconstructed images. By contrast, the same parameters are much higher for images without the color restoration algorithm implemented, and hence by human observers the colors will be recognized as different.

V. Conclusions

This paper presented an architecture for real-time barrel distortion correction of color VGA images. The system's logic was entirely contained within a standard low-cost FPGA chip. The only necessary external component was a memory chip in which a frame buffer was implemented. The proposed solution was capable of processing images at the rate of 30 frames per second. A significant decrease of ΔE values in the CIELab color space was obtained after implementation of the color reconstruction algorithm. This decrease translates into a better visual resemblance of original and corrected colors.

References

[1] K. Wonjun and K. Changick, "An efficient correction method of wide-angle lens distortion for surveillance systems," in *Proc. IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, May 2009, pp. 3206–3209.

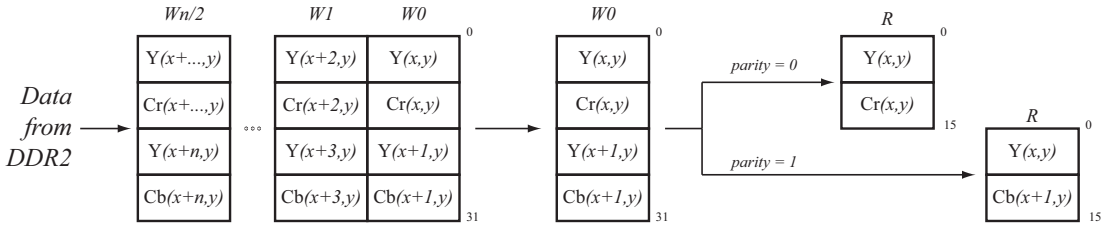
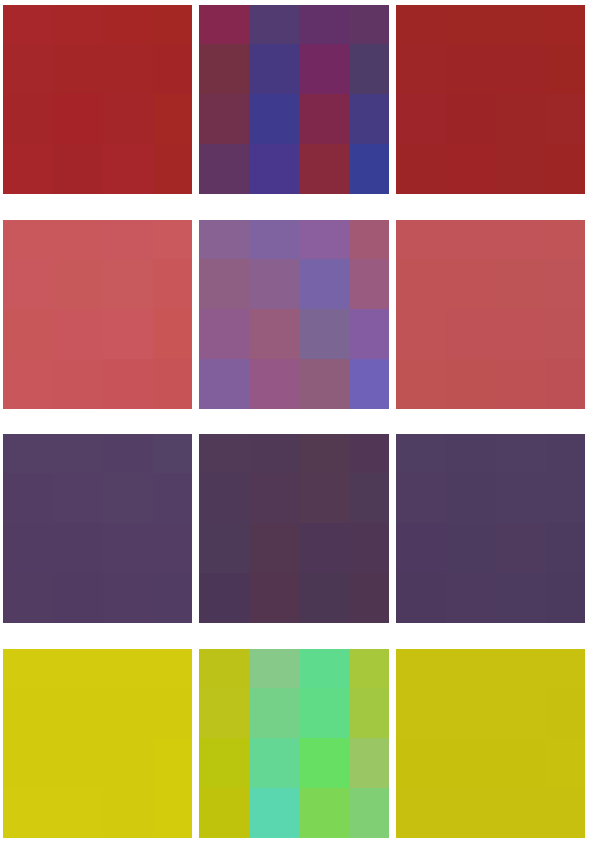


Fig. 4. Color restoration process in the *Data validation* block. Data nonadjacent to the pixel (x, y) is discarded (words $W1$ to $Wn/2$) and depending on the parity appropriate 16 bit word R is formed and send to *Data out FIFO*. Note that depending on the pixel location in the line: x coordinate, Cr and Cb bytes will interchange.

TABLE III
QUANTITATIVE ANALYSIS OF COLOR RESTORATION ALGORITHM.

Region	Average intensity	R	G	B
#1	Original, I_o	166	40	40
	No restoration, I_{nr}	103	49	100
	With restoration, I_r	158	38	37
	$\Delta E_{nr} = \lvert I_o - I_{nr}\rvert_{CIELab}$	22.7		
	$\Delta E_r = \lvert I_o - I_r\rvert_{CIELab}$	**1.9**		
#2	Original, I_o	202	90	93
	No restoration, I_{nr}	139	95	142
	With restoration, I_r	191	183	86
	$\Delta E_{nr} = \lvert I_o - I_{nr}\rvert_{CIELab}$	15.1		
	$\Delta E_r = \lvert I_o - I_r\rvert_{CIELab}$	**1.3**		
#3	Original, I_o	83	64	102
	No restoration, I_{nr}	81	57	85
	With restoration, I_r	78	60	96
	$\Delta E_{nr} = \lvert I_o - I_{nr}\rvert_{CIELab}$	3.1		
	$\Delta E_r = \lvert I_o - I_r\rvert_{CIELab}$	**1.0**		
#4	Original, I_o	211	203	13
	No restoration, I_{nr}	131	209	105
	With restoration, I_r	201	193	15
	$\Delta E_{nr} = \lvert I_o - I_{nr}\rvert_{CIELab}$	26.2		
	$\Delta E_r = \lvert I_o - I_r\rvert_{CIELab}$	**2.0**		



(a) Original (b) No color restoration (c) With color restoration

Fig. 5. Qualitative evaluation of color restoration mechanism (best viewed in color). Patches correspond to regions #1 (top) through #4 (bottom) in Table III.

[2] J. Courbon, Y. Mezouar, L. Eck, and P. Martinet, "A generic fisheye camera model for robotic applications," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, IROS*, October-November 2007, pp. 1683–1688.

[3] A. Basu and S. Licardie, "Modeling fish-eye lenses," in *Proc. of the 1993 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, July 1993, pp. 1822–1828.

[4] F. Devenray and O. Faugeras, "Straight lines have to be straight," *Machine Vision and Applications*, vol. 13, no. 1, pp. 14–24, August 2001.

[5] C. Ishii, Y. Sudo, and H. Hashimoto, "An image conversion algorithm from fish eye image to perspective image for human eyes," in *Proceedings IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, vol. 2, 20-24 2003, pp. 1009–1014 vol.2.

[6] H. T. Ngo and V. K. Asari, "A pipelined architecture for real-time correction of barrel distortion in wide-angle camera images," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 14, no. 3, pp. 436–444, March 2005.

[7] H. Ngo and V. Asari, "Developing a FPGA-based high performance, power-aware architecture for real-time radial lens distortion correction of video stream," *ICGST Intl. J. on Programmable Devices, Circuits and Systems, PDCS*, vol. 7, pp. 33–41, 2007.

[8] K. Gribbon, C. Johnston, and D. Bailey, "A real-time FPGA implementation of a barrel distortion correction algorithm with bilinear interpolation," in *Proc. of Image and Vision Computing New Zealand*, November 2003, pp. 408–413.

[9] A. Hernandez, A. Gardel, L. Perez, I. Bravo, R. Mateos, and E. Sanchez, "Real-time image distortion correction using FPGA-based system," in *Proc. IEEE 32nd Annual Conf. on Industrial Electronics IECON*, November 2006.

[10] H. Blasinski, W. Hai, and F. Lohier, "FPGA architecture for real-time barrel distortion correction of colour images," in *Proc. IEEE Intl. Conf. on Multimedia and Expo (ICME)*, July 2011, pp. 1–6.

[11] S. Shah and J. Aggarwal, "A simple calibration procedure for fish-eye (high distortion) lens camera," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, no. 4, May 1994, pp. 3422–3427.

[12] J. E. Farrell, F. Xiao, P. B. Catrysse, and B. A. Wandell, "A simulation tool for evaluating digital camera image quality," in *Proc. SPIE 5294*, vol. 124, 2003.