Real-Time Digital Image Stabilization for Cell Phone Cameras in Low-Light Environments without Frame Memory

Lin-bo Luo and Jong-wha Chong

This letter proposes a real-time digital image stabilization system for cell phone cameras without the need for frame memory. The system post-processes an image captured with a safe shutter speed using an adaptive denoising filter and a global color correction algorithm. This system can transfer the normal brightness of an image previewed under long exposure to the captured image making it bright and crisp with low noise. It is even possible to take photos in low-light conditions. By not needing frame memory, the approach is feasible for integration into the size-constrained image sensors of cell phone cameras.

Keywords: Digital image stabilization, denoising, color correction, frame memory, cell phone camera.

I. Introduction

Most cell phone cameras adopt a fixed aperture in which users have to extend the exposure time to obtain more photons in low ambient light conditions. Unfortunately, a long exposure time will, in all probability, generate a motion-blurred image. To reduce or remove motion blur and enhance the image quality of a phone camera, many methods have been proposed. Optical image stabilization (OIS) compensates for movement by adjusting the lens automatically or shifting the sensor of the camera [1]. Although OIS can achieve high performance, it requires additional apparatuses and space; therefore, it is not the best alternative for low-cost phone cameras. For phone cameras, digital image stabilization (DIS) algorithms are more frequently employed. One such DIS is deblurring, which estimates the point spread function (PSF) of a blurred image and tries to restore it [2]. However, it is difficult to accurately estimate the PSF of an image with a long exposure and complex motion without prior knowledge. Furthermore, deblurring algorithms require a long computational time. Several algorithms [3]-[5] have used two images: image A is previewed with a long exposure time, which is blurred but still has an acceptable brightness, and image B is captured using a safe shutter speed and high sensor sensitivity (ISO), which is dark and noisy but crisp. The algorithms first denoise image B and then transfer the tone of image A to image B. By taking advantage of the respective merits of the two images, the output image can be bright and crisp but still retain low noise. However, these algorithms [3]-[5] cannot perform in real-time processing and all of them have to use frame memory.

Therefore, based on the methods used in [3]-[5], this letter adds one more preview image and proposes a possible framework that can implement DIS in real-time requiring only a few line buffers, but not a frame buffer. Our study focuses on the architectural design of real-time implementation of DIS.

II. Proposed Method

Figure 1 illustrates the proposed framework of DIS. As shown in Fig. 1(a), three consecutive images are used, two of which are preview images, and the other is the captured image.

Manuscript received July 23, 2011; revised Oct. 23, 2011; accepted Nov. 8, 2011.

This work was supported by Hynix Semiconductor, the Brain Korea 21 Project in 2011, the MKE Korea, the ITRC support program supervised by the NIPA (NIPA-2011-C1090-1100-0010), and ETRI System Semiconductor Industry Promotion Center, Human Resource Development Project for SoC Convergence.

Lin-bo Luo (phone: +82 10 5858 6499, luolb@htomail.com) is with the Department of Electronics and Computer Engineering, Hanyang University, Seoul, Rep. of Korea, and also with the Faculty of Mechanical and Electronic Information, China University of Geosciences, Wuhan, China.

Jong-wha Chong (jchong@hanyang.ac.kr) is with the Department of Electronics and Computer Engineering, Hanyang University, Seoul, Rep. of Korea.

http://dx.doi.org/10.4218/etrij.12.0211.0338

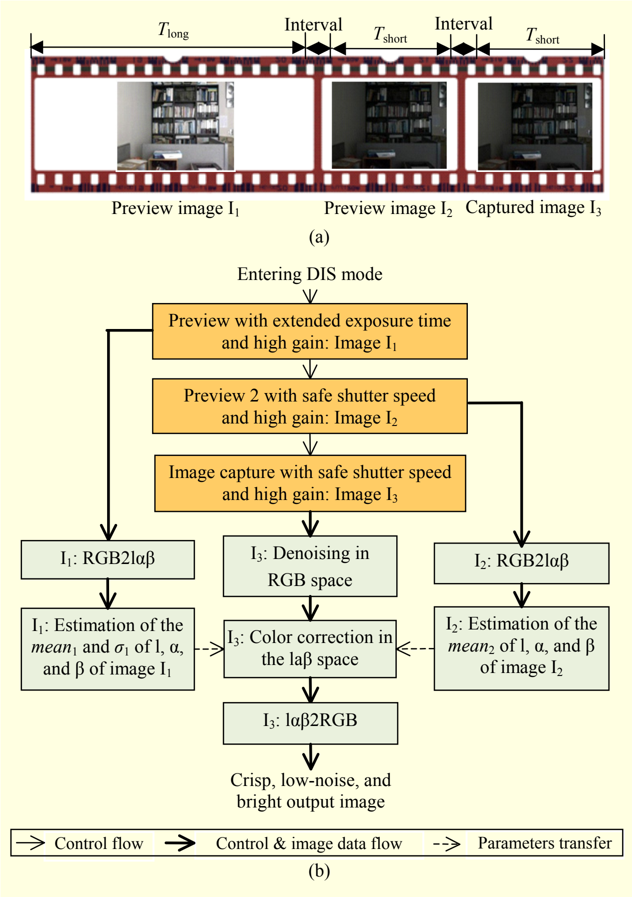

Fig. 1. Illustration of proposed DIS: (a) three images used in this method and (b) system diagram.

The first preview image (I_1) is acquired under a long exposure time (T_{long}). Although I_1 is motion-blurred because of hand motion, the brightness and color of the image are still acceptable. The second preview image (I_2) is acquired using a safe shutter speed (T_{short}) and a high ISO, producing an under-exposed image where the motion blur is largely reduced, but it is too dark and has serious noise. The third image (I_3) is captured using the identical settings of the sensor with I_2. Thus, the characteristics of I_3 are the same as I_2.

As shown in Fig. 1(b), we first denoised the captured image I_3 and then transferred the brightness of I_1 to I_3 using the parameters of I_1 and I_2. Finally, we obtain the output image, which is crisp and bright, while retaining an acceptable level of noise.

This design is based on two observations: (i) the intervals between the three images are very short and so have nearly identical scenes and (ii) the same sensor setting is used to acquire I_2 and I_3. Therefore, the statistic parameters of I_2 can be treated as the approximation of I_3.

The innovation of this algorithm is to use the parameters of I_2, instead of I_3, to perform the color correction. Through this adjustment, we can pre-calculate all the parameters needed for color correction before I_3 and hence do not need a frame buffer to store the image data. Otherwise, if not using I_2 but using the parameters of I_3 in the process, frame memory has to be used. Frame memory has to be used to buffer the I_3 data while we are calculating the parameters because the calculations of the parameters, such as mean and standard deviation, have to use the data of I_3, and we have to first obtain the parameters and then use them to process I_3.

1. Modified Adaptive Spatial-Tonal Denoising

Because I_3 is taken under low-light conditions, a relatively higher ISO is used, resulting in more noise. Most of this noise consists of readout electronics and quantization noise, which can be approximated to be independent of signal. To suppress this noise, an adaptive spatial-tonal normalized convolution can be used. The denoising filter can be mathematically expressed as

$$F(I_p)=\frac{\sum_{P\in Ns} G_s(|q-p|)G_t(|I_q-I_p|)I_p}{\sum_{P\in Ns} G_s(|q-p|)G_t(|I_q-I_p|)}, \qquad (1)$$

where I_p and $F(I_p)$ are the input and output pixel value, respectively, G_s is the spatial Gaussian kernel, G_t is the tonal Gaussian kernel, and N_s is the spatial neighborhood of pixel p.

Equation (1) is the general form of the bilateral filter (BF), which uses a Gaussian function for both G_s and G_t. However, a BF is difficult to implement with hardware. For easy hardware implementation, we propose a modified adaptive spatial-tonal normalized convolution (MASTNC) based on [6]. The MASTNC is performed by two 3×3 masks, instead of the two Gaussian functions, as follows:

$$g_s=\frac{1}{\sum_{i=1}^{9} g_i}\begin{bmatrix} g_1 & g_2 & g_3 \\ g_4 & g_5 & g_6 \\ g_7 & g_8 & g_9 \end{bmatrix} \text{ and } g_t=\frac{1}{\sum_{i=1}^{9} f_i}\begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix}, \qquad (2)$$

where g_i and f_i are the spatial weight and the tonal weight, respectively. They can be calculated by

$$g_i = 2^{(2-d_i^s)},\ d_i^s = |x_i - x_5| + |y_i - y_5|,\ x_i, y_i = -1,0,1, \qquad (3)$$

$$\text{and } f_i = (255 - d_i^t)^q,\ d_i^t = |p_i - p_5|, \qquad (4)$$

where d_i^s and d_i^t are the spatial and tonal distance, respectively, x_i and y_i are the coordinates of the 3×3 window, p_i is the pixel value in the 3×3 window, and q controls the amount of edge smoothing.

After removing the exponential operations of the Gaussian functions, the proposed algorithm has three main 8-bit multiplications that can be easily implemented by hardware. Because this study focuses on the real-time hardware framework which does not use frame memory, q is set to a fixed value of 8 for simplicity.

2. Lαβ Color Space-Based Global Color Correction

The color correction module transfers the bright tonal information of I_1 to I_3, according to the statistics of I_1 and I_2. Because R, G, and B have a strong correlation, which forces all color channels to be modified in tandem, an orthogonal lαβ color space-based color correction [7] was adopted as

$$\begin{cases} l_{\text{out}} = \dfrac{\sigma_1^l}{\sigma_3^l}(l_3 - mean_3^l) + mean_1^l, \\ \alpha_{\text{out}} = \dfrac{\sigma_1^\alpha}{\sigma_3^\alpha}(\alpha_3 - mean_3^\alpha) + mean_1^\alpha, \\ \beta_{\text{out}} = \dfrac{\sigma_1^\beta}{\sigma_3^\beta}(\beta_3 - mean_3^\beta) + mean_1^\beta, \end{cases} \tag{5}$$

where l, α, and β are the l, α, and β components of the images, respectively, $mean$ and σ are the mean and standard deviation of the l, α, and β channels, subscript 1 and 3 denote the parameters of image I_1 and I_3, respectively, and the subscript 'out' denotes the output image data.

To save memory usage, we substitute the $mean_2$ and σ_2 of the image I_2 for the $mean_3$ and σ_3 when we use (5) to perform the color correction. Thus, there is no image data that must be stored because we can pre-calculate $mean_2$ before starting to capture and process I_3.

3. Pipelining Timing Design

As shown in Fig. 2, the pipelining framework mainly consists of three stages: the estimation of $mean_1$ and the standard deviation σ_1 of I_1; the estimation of $mean_2$ and the standard deviation σ_2 of I_2; and the denoising and color correction of I_3 using the parameters calculated in stages I and II.

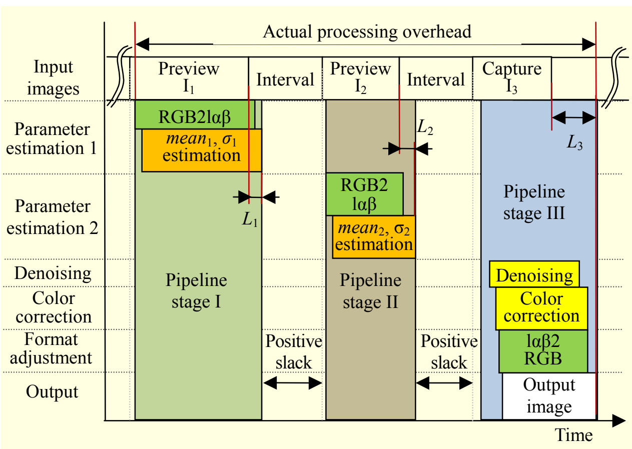

Fig. 2. Timing design of proposed system.

In Fig. 2, we performed RGB2lαβs and lαβ2RGB by fixed-point calculation using a left/right-shifting operation, then calculated $mean_1$, σ_1, $mean_2$, and σ_2 by an online algorithm. Thus, stages I and II are finished with latencies of L_1 and L_2 respectively, both of which are smaller than the interval between two images.

In the third stage, we used a circular buffer that consisted of 2 line buffers and 9 registers to realize the 3×3 convolutions of (2). As a result, there was a relatively long latency (L_3) in this stage. However, L_3 was also smaller than the interval between two images and did not affect real-time implementation.

Because we use the parameters of I_1 and I_2 to process I_3, this method does not require a frame buffer to store I_3. All three stages can be performed in real-time; they receive image data and output the required parameters or data after an acceptable latency.

III. Simulation Results

First, to verify the feasibility of using the mean and standard deviation of I_2 to substitute I_3, we presented a pilot study that took 11 consecutive frames using a hand-held camera and calculated the difference between every two consecutive frames. As shown in Table 1, the average is very small and almost all values are smaller than 1 except for one outlier. Therefore, the error due to using the $mean_2$ and σ_2 of I_2 to

Table 1. Difference in values of mean and standard deviation between two consecutive frames.

Para.	$\Delta_{1\text{-}2}$	$\Delta_{2\text{-}3}$	$\Delta_{3\text{-}4}$	$\Delta_{4\text{-}5}$	$\Delta_{5\text{-}6}$	$\Delta_{6\text{-}7}$	$\Delta_{7\text{-}8}$	$\Delta_{8\text{-}9}$	$\Delta_{9\text{-}10}$	$\Delta_{10\text{-}11}$	Average
$mean_l$	0.51	0.56	0.47	0.79	0.23	1.09	0.20	0.14	0.40	0.22	0.34
σ_l	0.18	0.15	0.00	0.43	0.52	1.10	0.00	0.08	0.22	0.21	0.21
$mean_\alpha$	0.24	0.58	0.39	0.71	0.36	1.07	0.21	0.13	0.47	0.24	0.32
σ_α	0.18	0.18	0.01	0.45	0.55	1.22	0.02	0.11	0.27	0.21	0.23
$mean_\beta$	0.41	0.57	0.29	0.70	0.20	0.97	0.02	0.19	0.19	0.38	0.28
σ_β	0.15	0.18	0.06	0.48	0.48	1.23	0.07	0.10	0.20	0.24	0.23

Table 2. Denoising result (PSNR) of proposed MASTNC compared to BF.

Test image	σ =10			σ =20			σ =30		
	Noisy	BF	Prop.	Noisy	BF	Prop.	Noisy	BF	Prop.
Lenna	28.15	33.41	33.07	22.11	27.57	28.82	18.72	22.17	25.84
Boat	28.14	31.50	30.91	22.18	26.90	27.89	18.74	21.94	25.33
House	28.16	33.56	32.66	22.11	27.56	28.64	18.71	22.13	25.64
Mandrill	28.17	28.90	27.71	22.26	25.67	25.99	18.85	21.59	24.18
Fingerprint	28.14	28.72	30.93	22.15	24.56	27.90	18.77	20.81	25.33

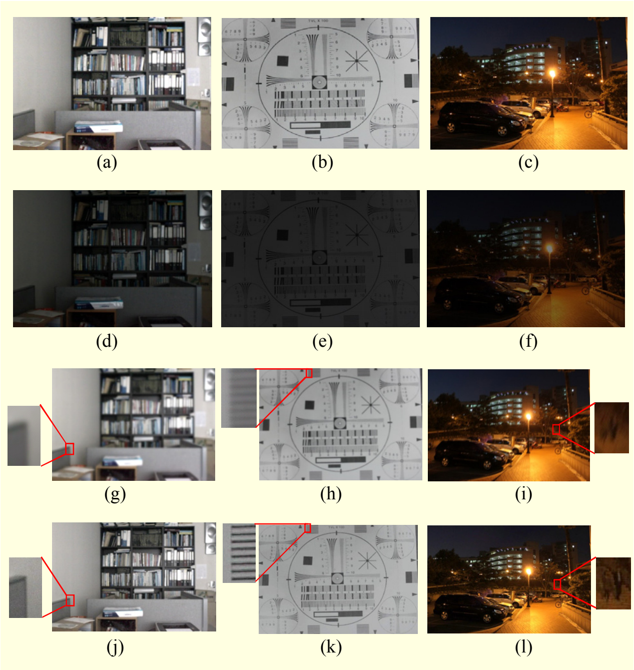

Fig. 3. Results of proposed method: (a) to (c) ground truth images, (d) to (f) images under short exposure, (g) to (i) images under long exposure, and (j) to (l) output images after color correction.

substitute for that of I_3 is negligible.

Second, we compared the proposed denoising algorithm with the BF. Although we optimized the BF to make it easy to implement with hardware, Table 2 shows that the decrease in PSNRs is not obvious when the standard deviation of the noise is 10. Our algorithm even outperforms the BF when the standard deviation of the noise is 20 or 30.

Then, as shown in Fig. 3, we used three scenes (indoor, test chart, and dusk) to illustrate the results of color correction for subject evaluation. Figures 3(j) to 3(l) show that the final output images are bright and crisp. For objective evaluation, the CIELAB color distortion metric is used as

$$\Delta E = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\sqrt{\Delta L_{ij}^{*2} + \Delta a_{ij}^{*2} + \Delta b_{ij}^{*2}}, \qquad (6)$$

where ΔL^*, Δa^*, and Δb^* are the differences in the L*a*b* color space, and M and N are the width and height of the test image, respectively. According to the standard, the distortion is barely perceptible when ΔE is smaller than 3, while it is perceptible but acceptable when ΔE ranges from 3 to 6. In our simulation, ΔE between the output images (Figs. 3(j) to 3(l)) and the reference images (Figs. 3(a) to 3(c)) are 2.39, 1.19, and 2.81, respectively.

Finally, the memory and system requirements are listed in Table 3. In our experiment, the shutter speed of previewing I_1, I_3, and capturing I_3 are 1/3 s, 1/60 s, and 1/60 s, respectively.

Table 3. Comparison of cost and system requirements.

Algorithm	[3]	[4]	[5]	Method A [8]	Proposed
Implementation	S/W	S/W	S/W	S/W&H/W	H/W
System requirement	PC	PC	PC	OMAP 3 processor	FPGA or ASIC
Time consumption	Not real-time	14.7 s	Not real-time	1.4 s/mpix	Real-time
Memory usage	1 frame memories	1frame memory	≥ 2 frame memories	1 frame memory	No frame memory

The latency of L_1 and L_2 are only a couple of clocks, and L_3 is longer than 2 rows but shorter than 3 rows. Because the time of each row is only 33.5 μs and the interval between two images is longer than 1 ms, the proposed algorithm can be implemented by hardware in real-time and does not need any frame memory.

IV. Conclusion

We proposed a pipelining DIS framework that does not require a frame buffer. Simulation results show that an acceptable final output image can be obtained in real-time. The proposed method is very suitable for applications that have constraints of size and cost, such as camera phones, robot vision, automobile cameras, and related fields.

References

[1] J.-H. Moon and S.-Y. Jung, "Implementation of an Image Stabilization System for a Small Digital Camera," *IEEE Trans. Consum. Electron.*, vol. 54, no. 2, May 2008, pp. 206-212.

[2] H. Hong and I.-K. Park, "Single-Image Motion Deblurring Using Adaptive Anisotropic Regularization," *Optical Eng.*, vol. 49, no. 9, 2010, pp. 097008-1-097008-13.

[3] J. Jia et al., "Bayesian Correction of Image Intensity with Spatial Consideration," *Proc. ECCV, LNCS*, vol. 3024, 2004, pp. 342-354.

[4] J.-H. Lee et al., "Anti-shaking Algorithm for the Mobile Phone Camera in Dim Light Conditions," *Proc. 10th Pacific Rim Conf. Multimedia: Adv. Multimedia Inf.*, 2009, pp. 968-973.

[5] M. Tico and K. Pulli, "Low-Light Imaging Solutions for Mobile Devices," *Proc. 43rd Asilomar Conf. Signals, Syst. Comput.*, 2009, pp. 851-855.

[6] T.-Q. Vinh and Y.-C. Kim, "Edge-Preserving Algorithm for Block Artifact Reduction and its Pipelined Architecture," *ETRI J.*, vol. 32, no. 3, June 2010, pp. 380-389.

[7] E. Reinhard and T. Pouli, "Colour Spaces for Colour Transfer," *LNCS*, vol. 6626, 2011, pp. 1-15.

[8] White Paper, Almalence Inc. www.almalence.com