Ringing Artifacts Removal System for Mobile Application Camera by Modified K-means Algorithm

Wonwoo Jang[#1], Junghwan Park[#2], Joohyun Kim[*3], Boodong Kwak[*4], Bongsoon Kang[#5]

[#]Multimedia Research Center, Dept. of Electronics Engineering, Dong-A University,
840 Hadan-dong, Saha-gu, Busan, South Korea
{[1]3per2kt, [2]drum0103}@didec.donga.ac.kr, [5]bongsoon@dau.ac.kr
[*]SAMSUNG Electro-Mechanics Co. Ltd
314 Maetan-3-dong, Young-tong-gu, Suwon, South Korea
{[3]joohyunkim, [4]boodong.kwak}@samsung.com

***Abstract*-This paper presents a new adaptive image quality enhancement algorithm and hardware implementation for reducing ringing artifacts. The proposed system operates with a block structure and 24-bit RGB data. Based on the modified K-means algorithm, two representative colors are generated in neighboring each block of the image, and ringing artifacts are reduced by moving all the pixels in each block to the closer of the two representative colors. The proposed system has been demonstrated experimentally by using a Xilinx VirtexII FPGA XCV6000, USB interface board, and a conventional CMOS sensor module for mobile application.**

I. INTRODUCTION

Compression and interpolation are used in a variety of video devices such as mobile camera phones, digital cameras, and TV sets [1]. These devices use compressed video data; therefore, in order to display the compressed data, they should use interpolation to generate the missing data. For a higher data compression ratio, more visual artifacts are generated due to quantization and truncation errors involved in interpolation. We focus on the ringing artifacts of some visual artifacts. Ringing artifacts are generated due to noise yielded by the boosted high-frequency components. We observe an International Organization for Standardization (ISO) resolution chart image with ringing artifacts; this image was taken by a mobile phone camera with a CMOS sensor module with the Bayer pattern. Since each pixel of the image through the Bayer pattern has only one of the three colors, we use color interpolation to generate the missing color data. This is when ringing artifacts are generated. Therefore, this paper presents an ringing artifacts removing algorithm and a system called Advanced Detail ENhancement (***ADEN***)

II. PROPOSED ALGORITHM

ADEN is based on a block structure and on 24 bits of RGB data. The block size is 16 × 16 pixels. For each block of the image, we set two ***representative colors*** by the K-means algorithm which is clustering objects based on attributes into K mutually exclusive partitions (K, positive integer) [2].

In order to handle the algorithm, we first decide the number of clusters, which is denoted by K. Assume a point in a cluster as the center. We calculate the distances between the assumed center and all objects. The objects that are the closest to a particularly assumed center are grouped into one cluster. In the next step, all of the distances in each cluster are iteratively summed to obtain a new center in each cluster. The optimization process involves the determination of a center such that the smallest sum is produced; this is the final point in each cluster. However, too many iterations are required in the abovementioned K-means process to determine each final point. Therefore, we propose a modified K-means algorithm that will be used in ***ADEN***. There are two modifications made to the K-means algorithm. One is that the number of clusters is two (K = 2). The other is that the number of iterations is limited to only one [3]. Figure 1 shows the modified K-means algorithm on three-dimensional RGB spaces.

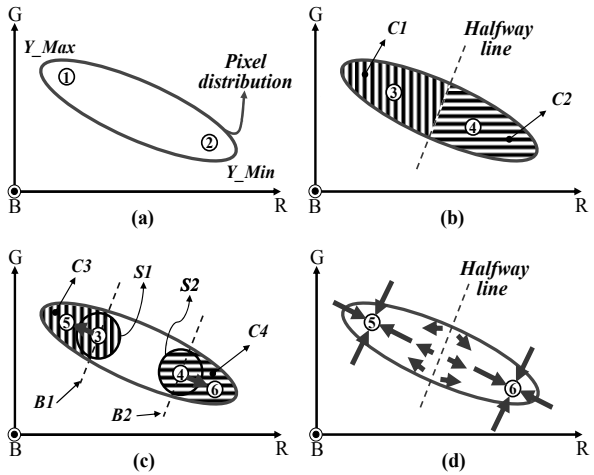


Figure 1. Diagram of the modified K-means algorithm

In Fig. 1(a), we select two pixels having maximum and minimum values of luminance among all of the pixels in the block. We know that ① is the pixel with the maximum value

of luminance and ② is the pixel with the minimum value of luminance in the pixel distribution. For the modified K-means algorithm, we consider the ① and ② as the initial points, ***Y_Max*** and ***Y_Min***.

In Fig. 1(b), we divide the pixel distribution into two parts far and close by the ***halfway line*** and define two clusters, ***C1*** and ***C2***. Eq. (1) groups all the pixels in each block into ***C1*** or ***C2*** by comparing the distance between each pixel and the initial points and summing up the value for each pixel.

$$\begin{aligned} &if\ (dist_{2\_1} <= dist_{2\_2}) \\ &\quad t_1rp_1 = t_1rp_1 + R_{in};\ \ t_1gp_1 = t_1gp_1 + G_{in}; \\ &\quad t_1bp_1 = t_1bp_1 + B_{in};\ N_{11}++; \\ &else \\ &\quad t_1rp_2 = t_1rp_2 + R_{in};\ t_1gp_2 = t_1gp_2 + G_{in}; \\ &\quad t_1bp_2 = t_1bp_2 + B_{in};\ N_{12}++; \end{aligned} \tag{1}$$

where $\boldsymbol{dist_{2\_1}}$ is the distance between ***Y_Max*** and all pixels, while $\boldsymbol{dist_{2\_2}}$ is the distance between ***Y_Min*** and all pixels. $\boldsymbol{R_{in}}$, $\boldsymbol{G_{in}}$ and $\boldsymbol{B_{in}}$ denote the component values of each pixel in the pixel distribution. $\boldsymbol{t_1rp_1}$, $\boldsymbol{t_1gp_1}$ and $\boldsymbol{t_1bp_1}$ are the accumulated values of the RGB pixels and $\boldsymbol{N_{11}}$ is the number of pixels in the ***C1*** area. $\boldsymbol{t_1rp_2}$, $\boldsymbol{t_1gp_2}$ and $\boldsymbol{t_1bp_2}$ are the accumulated values of the RGB pixels and $\boldsymbol{N_{12}}$ is the number of pixels in the ***C2*** area. We define two ***representative colors*** as follows:

$$Rep_{2\_1} = (t_1rp_1 / N_{11},\ t_1gp_1 / N_{11},\ t_1bp_1 / N_{11}) \tag{2}$$

$$Rep_{2\_2} = (t_1rp_2 / N_{12},\ t_1gp_2 / N_{12},\ t_1bp_2 / N_{12}) \tag{3}$$

where $\boldsymbol{Rep_{2\_1}}$ and $\boldsymbol{Rep_{2\_2}}$ denote the average values and are indicated by ③ and ④ in Fig. 1(b).

In Fig. 1(c), we update the ***representative colors*** by using the limited distance and dot-product to compensate for the weak points caused by the two modifications. We define two spheres ***S1*** and ***S2*** and two bars ***B1*** and ***B2*** that are centered at $\boldsymbol{Rep_{2\_1}}$ and $\boldsymbol{Rep_{2\_2}}$, respectively. We now assign the limited distance by considering a sphere whose radius is one-fourth of the distance between $\boldsymbol{Rep_{2\_1}}$ and $\boldsymbol{Rep_{2\_2}}$. The dot product is defined as follows:

$$\mathbf{a} \bullet \mathbf{b} = |a||b|\cos\theta \tag{4}$$

where *a* and *b* denote the magnitude of **a** and **b,** and θ is the angle between **a** and **b**. When the cosine values are less than zero (dot-product < 0), the angles range from π/2 to 3π/2. We define $\boldsymbol{dotproduct_{3\_1}}$ and $\boldsymbol{dotproduct_{3\_2}}$ as follows:

$$dotproduct_{3\_1} = \overrightarrow{((R_{in}, G_{in}, B_{in}) - Rep_{2\_1})} \bullet \overrightarrow{(Rep_{2\_2} - Rep_{2\_1})} \tag{5}$$

$$dotproduct_{3\_2} = \overrightarrow{((R_{in}, G_{in}, B_{in}) - Rep_{2\_2})} \bullet \overrightarrow{(Rep_{2\_1} - Rep_{2\_2})} \tag{6}$$

Eqs. (7) and (8) group all pixels in each block into two clusters on the basis of either limited distance or dot product. Using the dot product, we can obtain the pixel locations in the outer region between ***B1*** and ***B2***. Further, we can select pixels within the spheres ***S1*** and ***S2*** using limited distance. We define two clusters ***C3*** and ***C4*** in Fig. 1(c).

$$\begin{aligned} &if\ ((dist_{3\_1} <= limitdist)\,||\,(dotproduct_{3\_1} <= 0)) \\ &\quad t_2rp_1 = t_2rp_1 + R_{in};\ t_2gp_1 = t_2gp_1 + G_{in}; \\ &\quad t_2bp_1 = t_2bp_1 + B_{in}; N_{21}++; \\ &else \quad\quad "idle" \end{aligned} \tag{7}$$

$$\begin{aligned} &if\ ((dist_{3\_2} <= limitdist)\,||\,(dotproduct_{3\_2} <= 0)) \\ &\quad t_2rp_2 = t_2rp_2 + R_{in};\ t_2gp_2 = t_2gp_2 + G_{in}; \\ &\quad t_2bp_2 = t_2bp_2 + B_{in}; N_{22}++; \\ &else \quad\quad "idle" \end{aligned} \tag{8}$$

where $\boldsymbol{dist_{3\_1}}$ is the distance between $\boldsymbol{Rep_{2\_1}}$ and all pixels, while $\boldsymbol{dist_{3\_2}}$ is the distance between $\boldsymbol{Rep_{2\_2}}$ and all pixels.

limitdist indicates the radii of two spheres ***S1*** and ***S2***. $\boldsymbol{t_2rp_1}$, $\boldsymbol{t_2gp_1}$ and $\boldsymbol{t_2bp_1}$ are the accumulated values of the RGB pixels and $\boldsymbol{N_{21}}$ is the number of pixels in the ***C3*** area. $\boldsymbol{t_2rp_2}$, $\boldsymbol{t_2gp_2}$ and $\boldsymbol{t_2bp_2}$ are the accumulated values of the RGB pixels and $\boldsymbol{N_{22}}$ is the number of pixels in the ***C4*** area. We define two ***updated representative colors*** as follows:

$$Rep_{3\_1} = (t_2rp_1 / N_{21},\ t_2gp_1 / N_{21},\ t_2bp_1 / N_{21}) \tag{9}$$

$$Rep_{3\_2} = (t_2rp_2 / N_{22},\ t_2gp_2 / N_{22},\ t_2bp_2 / N_{22}) \tag{10}$$

where $\boldsymbol{Rep_{3\_1}}$ and $\boldsymbol{Rep_{3\_2}}$ denote the average values and are indicated by ⑤ and ⑥ in Fig. 1(c).

We can eliminate the ringing artifacts in the image by using ***Interval*** and ***Approach***. In Eq. (11), ***Interval*** is defined as the difference between the values of the ***representative colors*** and the values of all the pixels in the block. It accumulates all the difference values (***Interval***) of each RGB value. The accumulated value is denoted by ***Acc_interval*** for later use.

$$Interval = \left|R_{in} - R_{Rep2\_i}\right| + \left|G_{in} - G_{Rep2\_i}\right| + \left|B_{in} - B_{Rep2\_i}\right| \tag{11}$$

where R_{Rep2_i}, G_{Rep2_i}, and B_{Rep2_i} with $i = 1$ or 2 denote the component values of the ***representative colors*** of the pixels given by Eq. (1). We define ***Approach*** as follows:

$$Approach = \frac{(R_{in} - R_{Rep2\_1})^2 + (G_{in} - G_{Rep2\_1})^2 + (B_{in} - B_{Rep2\_1})^2}{(R_{in} - R_{Rep2\_2})^2 + (G_{in} - G_{Rep2\_2})^2 + (B_{in} - B_{Rep2\_2})^2} \tag{12}$$

In Fig. 1(d), we calculate new RGB values by using Eqs. (11) and (12). Eqs. (13)-(15) show the final RGB values obtained by using the proposed method.

$$R_{in\_mod} = R_{in} + k \times \frac{1 - Approach}{Acc\_interval} \times (R_{Rep3\_i} - R_{in}) \tag{13}$$

$$G_{in\_mod} = G_{in} + k \times \frac{1 - Approach}{Acc\_interval} \times (G_{Rep3\_i} - G_{in}) \quad (14)$$

$$B_{in\_mod} = B_{in} + k \times \frac{1 - Approach}{Acc\_interval} \times (B_{Rep3\_i} - B_{in}) \quad (15)$$

where k is a constant and R_{in_mod} indicates the final R value. Moreover, G_{in_mod} and B_{in_mod} are obtained in a similar manner. R_{Rep3_i}, G_{Rep3_i} and B_{Rep3_i} with $i = 1$ or 2 denote the component values of ***updated representative colors*** of the pixels as given in Eqs. (7) and (8). We then shift all the pixels of each block to the closer of the two ***updated representative colors*** inversely proportional to the distance between each ***updated representative color*** and each pixel. In this manner, we can eliminate the ringing artifacts.

III. ARCHITECTURE

We present a system based on ***ADEN***. Figure 2 shows the block diagram of ***ADEN***.

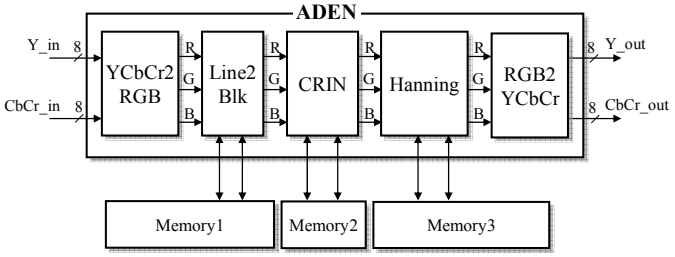


Figure 2. Block diagram of ADEN

The proposed system consists of five parts. Since the mobile application output is based on YCbCr samples for a 4:2:2 format, the bit width of the input and output signals used in the hardware is 16 bits. However, 24-bit RGB data is transferred over the inner blocks. Two blocks, YCbCr2RGB and RGB2YCbCr, are included to provided an environment that is compatible with the input and output signals. In the YCbCr2RGB block, 4:2:2 YCbCr data is first converted to 4:4:4 YCbCr by using interpolation to generate the missing Cb and Cr samples [4]. Secondly, the YCbCr color space is converted into an RGB color space. On the other hand, the RGB2YCbCr block follows a reverse order.

A general method to transfer image is as follows: we start at the top left corner of the image, move towards the right edge of the image, after which we move down one line. However, the proposed algorithm is based on block structure. The block line2blk stores the image data in the Memory1; this image data is for a 16 × 16 pixel image with 24 bits of RGB data per pixel. The data is then transferred to the block CRIN. The Hanning block restores the data of the 16 × 16 pixel image with 24 bits of RGB data per pixel in the form of lines by storing the block data in Memory3.

In the CRIN block, the modified K-means algorithm requires a set of four processing steps connected in series. This architecture requires that the output of the front step is the input of the next step. Since the amount of data processed in the modified K-means algorithm is four times that processed in the original algorithm, the pipeline architecture is used in this block to allow the parallel execution of four consecutive steps. Figure 3 shows the block diagram of CRIN.

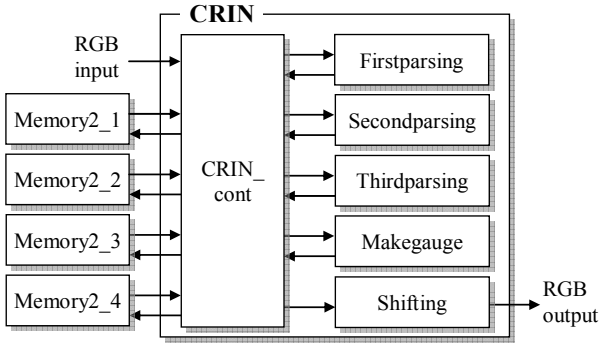


Figure 3. Block diagram of CRIN

The CRIN block consists of six parts. Since this block utilizes pipeline architecture, four Memory2 blocks are used. The CRIN_cont block generates the signals that decide the order in which the memories can be accessed and store initial points, ***representative colors***, ***updated representative colors*** and etc. for consecutive processing. The Firstparsing block selects two pixels with maximum and minimum luminance to define ***representative colors*** as shown in Fig. 1(a). The Secondparsing block finds the ***representative colors*** by modified K-means algorithm with K=2 and with one iteration as shown in Fig. 1(b). The Thirdparsing block updates ***representative colors*** by using dot-product and limited distance, as shown in Fig. 1(c). The Makegauge block calculates ***Interval*** and ***Approach***. The Shifting block calculates new RGB values to shift the closer of the two ***updated representative colors***, inversely proportional to the distance as Fig. 1(d) [5].

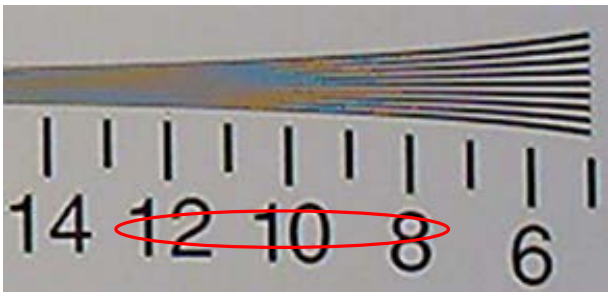
The proposed system is designed by using Verilog-HDL models and is verified by using the Synopsys simulator. The models are synthesized into gates to observe the hardware complexity by using the Synopsys synthesizer with the TSMC 0.25-μm ASIC library.

TABLE I
SYSTEM GATE COUNT

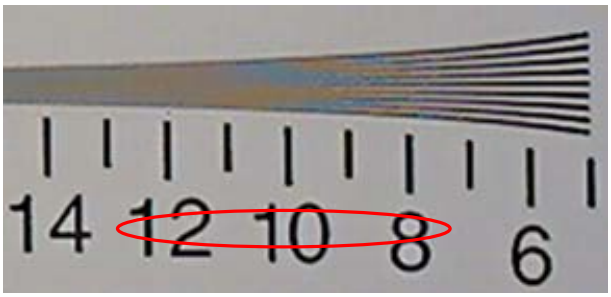
Block	Gate Count	Min Timing[MHz]
YCbCr2RGB	5,224	57.33
Line2Blk	5,298	85.98
CRIN	72,107	42.08
Hanning	12,672	43.78
RGB2YCbCr	4,785	55.55
Memory1	98,304 byte	
Memory2	3,200 byte	
Memory3	123,840 byte	
Logic Total	**100,086**	**43.1**
Used Memory	**225,344 byte**	

IV. EXPERIMENTAL RESULTS

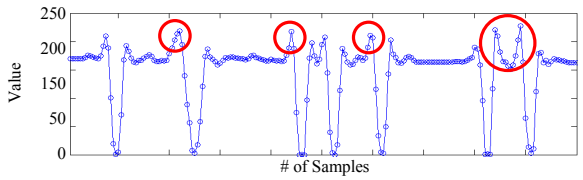
For performance of the proposed system, experiments were executed with the ISO resolution chart, as shown in Fig. 4. We took the original image inputted as Bayer pattern from a camera of mobile phone using the CMOS sensor module.



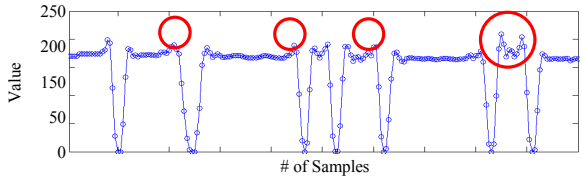
(a) Original image from a portion of the ISO resolution chart



(b) Image obtained in the proposed system



(c) One dimensional plot of (a) in Red Channel



(d) One dimensional plot of (b) in Red Channel

Figure 4. Experimental Result

Figures 4(a) and (c) were obtained without ADEN process. The red value shown in Fig. 4(c) is approximately between 0 and 230. However, we are able to discover positive overshoots and undershoots painted as a round around edges. Figure 4(b) shows the portion of resultant image by ADEN process. We are not able to perceive that circumference of the letters in Fig. 4(b) is more clear and explicit than the image in Fig. 4(a), but the over and undershoots are also removed as shown in Fig. 4(d). The over and undershoots are the principal factor of ringing artifacts, hence they ought to be eliminated for a vivid image.

Figure 5 shown below is PCB demonstration board for verification of this system [6]. The image data is processed by Xilinx FPGA VirtexII XCV6000 device after the CMOS sensor stores the real image data to the Frame Memory, and the processed data is then transferred to the display device by using the USB Output.

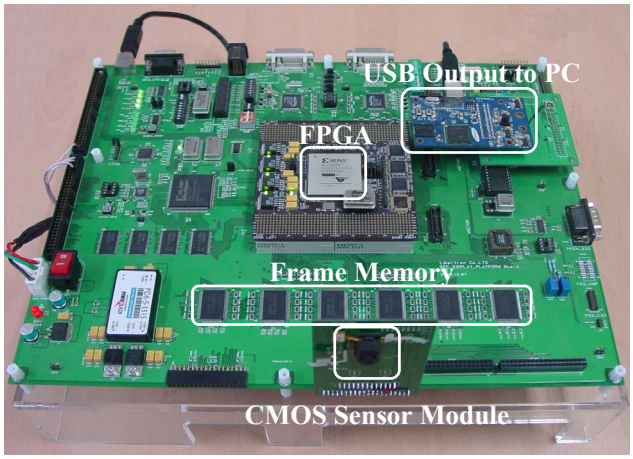


Figure 5. PCB demonstration board

V. CONCLUSIONS

In this paper, we have proposed an ***ADEN*** system that can be applied to mobile application camera. The ***ADEN*** system operates on a block structure with 24-bit RGB data. The block size is 16 × 16 pixels. First, two ***representative colors*** have been generated by the modified K-means algorithm on each block of the image. Secondly, the ***representative colors*** are updated by using dot-product and limited distance. Finally, this system eliminates ringing artifacts by moving all pixels of the block to the closer ***updated representative color***. The ability of the proposed system was shown in Fig. 4.

Since we have verified the feasibility of this system for a processed image with ringing artifacts, we can conclude that the proposed ***ADEN*** system can be used in any mobile application camera.

ACKNOWLEDGMENT

The authors wish to thank the IDEC for its software assistance.

REFERENCES

[1] H. S. Kong, V. A., Huifang Sun, "Edge map guided adaptive post-filter for blocking and ringing artifacts removal," Circuits and Systems, ISCAS2004, vol. 3, pp. 929-932, May, 2004.

[2] D. Charalampidis, "A modified k-means algorithm for circular invariant clustering," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 27, no. 12, pp. 1856-1865, Feb. 2005.

[3] J. Kim, W. Jang, B. Kwak, S. Kim, and B. Kang, "A New Image-scaling Algorithm Eradicating Blurring and Ringing to Apply to Camera Phones," 2007 International Conference on Consumer Electronics, pp. 5.4-1, Jan. 2007.

[4] K. Jack, *Video Demystified: A Handbook for the Digital Engineer*, HighText Pub., 1996.

[5] J. Kim, *Design of Smart Zoom System with Boost-up Filter and ADEN Algorithm*, Ph.D. Thesis, Dong-A University, Busan, Korea, Dec. 2006.

[6] J. Kim, J. Park, W. Choi, B. Kang, "Implementation of Advanced Image Scaler for Mobile Device by the effective of group delay," The Korea Institute of Signal Processing and Systems, vol. 8, no. 3, pp. 163-170, Jul. 2007.