Infrared Physics & Technology 85 (2017) 410–420

Contents lists available at ScienceDirect

Infrared Physics & Technology

journal homepage: www.elsevier.com/locate/infrared

Regular article

An improved non-uniformity correction algorithm and its hardware implementation on FPGA

Rong Shenghui [a], Zhou Huixin [a,*], Wen Zhigang [b,c], Qin Hanlin [a], Qian Kun [a], Cheng Kuanhong [a]

[a] School of Physics and Optoelectronic Engineering, Xidian University, Xi'an, China
[b] Xi'an Institute of Optics and Precision Mechanics of CAS, Xi'an, China
[c] Key Laboratory of Spectral Imaging Technology CAS, Xi'an, China

HIGHLIGHTS

- An improved neural network based non-uniformity correction algorithm is proposed.
- The Guided Image Filter is utilized to calculate the desired image to reduce the artificial ghosting.
- A projection-based motion detection is proposed to control the process of the parameter update.
- We designed and implemented a prototype in a field-programmable-gate-array (FPGA).

ARTICLE INFO

Article history:
Received 8 December 2016
Revised 19 May 2017
Accepted 4 July 2017
Available online 8 July 2017

Keywords:
Infrared focal plane array
Non-uniformity correction
Neural network
Guided filter
Motion detection
FPGA

ABSTRACT

The Non-uniformity of Infrared Focal Plane Arrays (IRFPA) severely degrades the infrared image quality. An effective non-uniformity correction (NUC) algorithm is necessary for an IRFPA imaging and application system. However traditional scene-based NUC algorithm suffers the image blurring and artificial ghosting. In addition, few effective hardware platforms have been proposed to implement corresponding NUC algorithms. Thus, this paper proposed an improved neural-network based NUC algorithm by the guided image filter and the projection-based motion detection algorithm. First, the guided image filter is utilized to achieve the accurate desired image to decrease the artificial ghosting. Then a projection-based moving detection algorithm is utilized to determine whether the correction coefficients should be updated or not. In this way the problem of image blurring can be overcome. At last, an FPGA-based hardware design is introduced to realize the proposed NUC algorithm. A real and a simulated infrared image sequences are utilized to verify the performance of the proposed algorithm. Experimental results indicated that the proposed NUC algorithm can effectively eliminate the fix pattern noise with less image blurring and artificial ghosting. The proposed hardware design takes less logic elements in FPGA and spends less clock cycles to process one frame of image.

1. Introduction

Infrared Focal Plane Array (IRFPA) is the most important part in an infrared imaging system [1]. However, due to the immature manufacturing processes and the dark-current noise, different detectors of an IRFPA are unable to output the same value under the same infrared irradiance, which is called the non-uniformity of IRFRA. This phenomenon will impose some fixed pattern noise (FPN) on the infrared image, which severely degrades the quality of images. Therefore, developing an effective non-uniformity correction (NUC) algorithm and implementing it on the embedded hardware platform are necessary tasks to improve the image quality for an infrared imaging system.

In recent years, different NUC algorithms have been proposed, which can be mainly divided into two categories, the reference-based NUC and the scene-based NUC algorithm. Two-point correction (TPC) is a typical reference-based algorithm. This method employs uniform blackbody as reference irradiance sources to calculate the correction parameter. The advantages of these methods lie in its simplicity and low computational complexity. Thus, it is very suitable to implement this kind of algorithms on the embedded hardware platform. For instance, Zhou [2] presented an improved TPC algorithm and implemented it on a FPGA + DSP based platform. Tomasz Sosnowski [3] implemented TPC as a part of image processing on an FPGA based platform. However, repeti-

* Corresponding author.
E-mail address: hxzhou@mail.xidian.edu.cn (H. Zhou).

http://dx.doi.org/10.1016/j.infrared.2017.07.007
1350-4495/

tive recalibration must be done when correction coefficients become invalid, which would interrupt the normal operation of the imaging system and increase maintenance costs.

The scene-based NUC algorithm can acquire and calibrate coefficients continually and automatically without interrupting the normal system operation. Typical scene-based NUC algorithms mainly contain constant statistics [4–7] based, temporal high-pass filter [8–10] based, Kalman filter based [11–14], registration based [15–17] and neural network [18] based algorithms.

The neural network based non-uniformity correction (NN-NUC) algorithms are widely used because of its better adaptively and noise immunity. However, the traditional NN-NUC would result in ghosting artifact to the moving object and image blurring to static scene, which restrict its application. In Scribner's algorithm, the local spatial neighborhood average is utilized to estimate the desired output of one pixel. The mean filter will lead the blurring of edge in estimated image and result in image blurring on the static scene and ghosting artifacts on the previous position of the moving objects.

Many researchers have proposed a number of improved algorithms. Hui Yu [18] set two different learning rate on gain and bias correction coefficients respectively, which will accelerate the convergence speed. Jufen zhao [19] utilized the ***L0*** gradient minimization method to achieve the estimated irradiance, which will make a good balance between non-uniformity and the details preservation. Fan FAN [20] utilized the a 1D-Guassian filter to achieve the desired image for the line scanning Infrared image. Though, Nicolas Celedon et al. [21] presented a FPGA-based hardware platform to implement the original Scribner's algorithm. Most improved algorithms don't consider the real-time ability to apply it on the hardware platform.

In order to solve the problems mentioned above, in this paper, an improved NN-NUC algorithm is proposed. First, the mean filter in Scribner's algorithm is substituted by the guided filter (GF) [22]. The guided image filter is an edge-preserving smoothing operator, which will severely decline the ghosting artifacts during the correction process. Then, a moving detection algorithm based on projection is utilized to prevent the blurring of the stationary scene caused by invalid updating of coefficients. Finally, An FPGA-based hardware design of the proposed improved algorithm is introduced completely.

The remainder is organized as follows. In Section 2, the correction model of proposed NUC algorithm is Introduced. In Section 3, the hardware design and the implementation based on FPGA are discussed. In Section 4, the proposed algorithm is applied to real and simulated infrared image sequences and the experimental results are analyzed in detail. The hardware resource analysis and the conclusion are given in Section 5 and in Section 6 respectively.

2. Improved non-uniformity correction algorithm

2.1. Neural network based NUC

Neural network based Non-uniformity correction (NN-NUC) algorithm was first proposed by D. A. Scribner [1]. In this algorithm, it is assumed that the response of each detection-element in an IRFPA can be corrected by a linear model

$$I^k_{nuc}(i,j) = G^k(i,j) \times I^k_{orig}(i,j) + O^k(i,j) \tag{1}$$

where (i,j) is the coordinate of a detector element in an IRFPA, k means the number of frame, $I^k_{orig}(i,j)$ is the original output of each detectors before NUC, $I^k_{nuc}(i,j)$ is the gray value of each pixel after correction, and $G^k(i,j)$ and $O^k(i,j)$ represent the NUC gain and offset correction coefficients respectively.

As the sense varies, the correction coefficients are updated by the steepest descent algorithm frame by frame in order to obtain the optimal correction result, which can be expressed as

$$G^{k+1}(i,j) = G^k(i,j) - 2 \times \alpha \times I^k_{orig}(i,j) \times (I^k_{nuc}(i,j) - I^k_f(i,j)) \tag{2}$$

and

$$O^{k+1}(i,j) = O^k(i,j) - 2 \times \alpha \times (I^k_{nuc}(i,j) - I^k_f(i,j)) \tag{3}$$

where α is a small constant to control learning rate, and $I^k_f(i,j)$ is utilized to estimate the desired output of pixel (i,j). The traditional NN-NUC algorithm adopts the mean value of the four neighbor pixels as the desired output image, which can be expressed as

$$I^k_f(i,j) = \left(I^k_{orig}(i-1,j) + I^k_{orig}(i+1,j) + I^k_{orig}(i,j-1) + I^k_{orig}(i,j+1)\right)/4 \tag{4}$$

By this way, gain and offset correction coefficients will become stability and convergence after several frames. The NN-NUC algorithm can also recalibrate coefficients automatically for any temporal drift in the correction coefficients. However, the mean filter is not an ideal operator to obtain the accurate desired image, because it will lead distortion at sharp areas. The traditional NN-NUC would result in ghosting artifact to the moving object and image blurring to static scene, which restricts its application.

2.2. Guided image filtering

As mentioned above, the traditional NN-NUC algorithm suffers the drawback of ghosting artifact and image blurring. This is because the mean filter will cause gradient distortion and lose many detail features where the wrong correction coefficient would be generated. As the scene varies or the object moves in the image, the improper correction coefficient cannot be repaired immediately. As a result, the inappropriate correction coefficient will cause the image degradation and distortion on the correction results. Hence, a more accurate desired image is the key to improve the correction performance, especially for the image with sharp edges [23].

However both mean filter and other linear filter will smooth the details in the image. The reason is that the filter weight is always the same in the whole image. Thus, a nonlinear filter, the guided image filter (GIF), which could reserve the image details and reduce the FPN, is adopted to solve this problem discussed above.

The GIF can transfer the structure of the guidance image to the filtering output. When the guidance image is as same as the input one, the guided image filter acts as a kind of edge-preserve filter which can adjust the smooth degree based on the local characteristics of the image. Thus GIF can distinguish the edge and the smooth region in the image. It assumes that the filter out I^k_f can be a liner transform of an input original image I^k_{orig}.

$$I^k_f = a_w \cdot I^k_{orig} + b_w \tag{5}$$

where (a_w,b_w) are some linear coefficients assumed to be constant in a local window w. In order to seek a solution that minimizes the difference between I^k_{orig} and I^k_f while maintaining the linear model, a cost function in the local window w is introduced.

$$E(a_w, b_w) = \sum_{(i,j)\in w} ((a_w \cdot I^k_{orig}(i,j) + b_w - I^k_{orig}(i,j)) + \varepsilon a_w^2) \tag{6}$$

where ε is a regularization parameter penalizing large a_w. By the linear ridge regression model, its solution is given by

$$a_w = \sigma_w^2/(\sigma_w^2 + \varepsilon) \tag{7}$$

$$b_w = (1 - a_w)\mu_w \tag{8}$$

where μ_w and σ_k^2 are the mean and variance of the I_{orig}^k in the local window w.

If the input original image I_{orig}^k changes a lot within w, there is $\sigma_k^2 \gg \varepsilon$, so $a_w \approx 1$ and $b_w \approx 0$. The filter can preserve most of details. If the input image is almost constant in w, there is $\sigma_k^2 \ll \varepsilon$, so $a_w \approx 0$ and $b_w \approx \mu_w$. The filter can remove the FPN as same as a mean filter.

The detail operation of GIF will be described as follows. First, the average $mean_I$ and variance var_I are calculated in a local window for every pixel of the input image I_{orig}^k, which can be described as

$$mean_I = f_{mean}(I_{orig}^k) \tag{9}$$

and

$$var_I = f_{mean}(I_{orig}^k .* I_{orig}^k) - mean_I .* mean_I \tag{10}$$

where f_{mean} is a mean filter in a local window, "$.*$" means the multiplication of corresponding elements between two matrixes. These operational symbols have the same meaning in this paper.

Next, the matrix a and b are calculated as

$$a = var_I ./ (var_I + \varepsilon) \tag{11}$$

and

$$b = mean_I - a .* mean_I \tag{12}$$

where "$./$" means the division of corresponding elements between two matrixes, and ε is a regularization parameter which can control the fuzzy degree. For every element of matrix a and b, the average are calculated in a local window, which can be described as

$$mean_a = f_{mean}(a) \tag{13}$$

and

$$mean_b = f_{mean}(b) \tag{14}$$

Finally, the output image I_f^k of GF can be obtained by

$$I_f^k = mean_a .* I_{orig}^k + mean_b \tag{15}$$

Through this accurate desire image, the correction coefficient can converge in the correct direction. And the drawback of ghosting artifact and image blurring can be avoided.

2.3. Projection-based motion detection

All the scene-based algorithm update correction coefficients based on the change of the scene. However, if the scene holds static, the correction coefficients will not updated appropriately which will lead image blurring. In order to solve this problem, a projection based moving detection algorithm is proposed in this paper. This algorithm can determine whether there is a significant motion between adjacent frames. Only if the motion between adjacent images is intense enough, will correction coefficients be updated. By this way, the image blurring in the static scene will be avoided.

For the current image I_{orig}^k, the projection vectors, vec_h^k and vec_v^k, are obtained by calculating the average of all pixels in the horizontal and the vertical direction respectively, which can be expressed as

$$vec_h^k(j) = \frac{1}{m}\sum_{i=1}^{m} I_{orig}^k(i,j) \quad j = 1,2,\ldots,n \tag{16}$$

$$vec_v^k(i) = \frac{1}{n}\sum_{j=1}^{n} I_{orig}^k(i,j) \quad i = 1,2,\ldots,m \tag{17}$$

where m and n are image height and width respectively. For previous image $I^{k-1}{}_{orig}$, the projection vector $vec^{k-1}{}_h$ and $vec^{k-1}{}_v$ in horizontal and vertical directions are calculated by the same algorithm.

Next, we calculate the difference value D_h^k between vec_h^k and $vec^{k-1}{}_h$ by

$$D_h^k = \sum_{j=1}^{n} \left| vec_h^k(j) - vec_h^{k-1}(j) \right| \tag{18}$$

and the difference value D_v^k between vec_v^k and $vec^{k-1}{}_v$ can be obtained as

$$D_v^k = \sum_{i=1}^{m} \left| vec_v^k(i) - vec_v^{k-1}(i) \right| \tag{19}$$

It is considered that there is a significant motion between adjacent images if $D_h^k > T$ or $D_v^k > T$, where T is a threshold. Thus, $D_h^k > T$ or $D_v^k > T$ means that the motion is intense enough and correction coefficients will be updated by the steepest descent algorithm, otherwise, correction coefficients will remain unchanged. This process can be expressed as follow (the pixel coordinates are omitted for simplicity). By this discriminant, over-correction can be avoided in the static scene.

$$G^{k+1} = \begin{cases} G^k - 2 \cdot \alpha \cdot I_{orig}^k \cdot (I_{nuc}^k - I_f^k), & D_v^k > T \text{ or } D_h^k > T \\ G^k, & D_v^k < T \text{ or } D_h^k < T \end{cases} \tag{20}$$

$$O^{k+1} = \begin{cases} O^k - 2 \cdot \alpha \cdot (I_{nuc}^k - I_f^k), & D_v^k > T \text{ or } D_h^k > T \\ O^k, & D_v^k < T \text{ or } D_h^k < T \end{cases} \tag{21}$$

In order to indicate the effectiveness of the proposed projection-based motion detection algorithm, a simulation experiment is introduced in this part. An artificial infrared sequence with static and slow motion scene is selected. The Metric RMSE illustrated in Section 4.3 is utilized to measure the performance. The RMSE results of the last frame are calculated with different threshold, which are shown in Fig. 1.

As can be seen form the figure, if the threshold is much larger, the correction parameters cannot be updated and the much FPN will be remained; if the threshold is much smaller, the motion detection algorithm will be invalidated and image blurring in the static scene will occur.

As mentioned above, different threshold will affect the correction result seriously. In this paper, the threshold with the minimum RMSE is selected to achieve the best performance, which equals 1000 in this sequence. However, the image size and camera character can also influent the selection of the threshold T. In practical applications, simulation experiment can be utilized to determine the appropriate threshold.

2.4. Improved NN-NUC algorithm

This paper presents an improved neural network NUC algorithm based on guided image filtering and projection motion detection. The guided image filter is utilized to overcome the drawback of mean filter in Scribner's algorithm. The moving detection algorithm is utilized to prevent the blurring of the static scene caused by the invalid update of coefficients. The proposed algorithm can be described as follows.

(1). Parameters initialization. For the first image, the correction parameters G^1 and O^1 are set as all-one matric and all-zero matric respectively.
(2). Non-uniformity correction. For the k-th image I_{orig}^k, the corrected image I_{nuc}^k is calculated by Eq. (1).
(3). Calculate the desire image. The desired image I_f^k is calculated by Guided Image Filter illustrated in Section 2.2.
(4). Calculate the difference between the adjacent frames. The D_v^k and D_h^k are calculated by the current image I_{orig}^k and the pre-

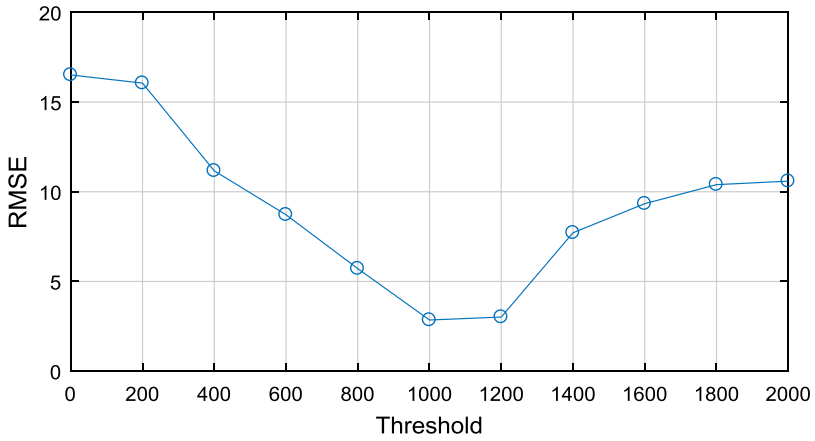


Fig. 1. RMSE results with different threshold.

vious image $I^{k-1}{}_{orig}$, which is performed as described in Section 2.3.

(5). Update the correction parameters. If the D_v^k or D_h^k are larger than the threshold T, The new values of G^{k+1} and O^{k+1} are calculated; otherwise, matrixes G^{k+1} and O^{k+1} should be kept consistent with the previous values.

(6). Repeat steps 2 to 5 until the entire sequence is processed.

3. FPGA based hardware design

3.1. The hardware platform

In order to correct the infrared image in real time, a suitable hardware platform is necessary. In this paper, a field-programmable gate array (FPGA) based hardware platform is proposed to implement this algorithm. Because of the capability of data parallelism, the FPGA can correct the NUC during the image capture, which meets the real-time requirement of the system. In this platform, the FPGA EP2S60F672I4 is the core which implements NUC algorithm and interfaces with the video decoder SAA7114, the video encoder SAA7121 and two pieces of Static Random Access Memory (SRAM). The diagram of our hardware platform is shown in Fig. 2.

3.2. Top level design

The hardware system design based on FPGA is usually followed a top-down design construction. The top level design diagram of FPGA in our system is shown in Fig. 3.

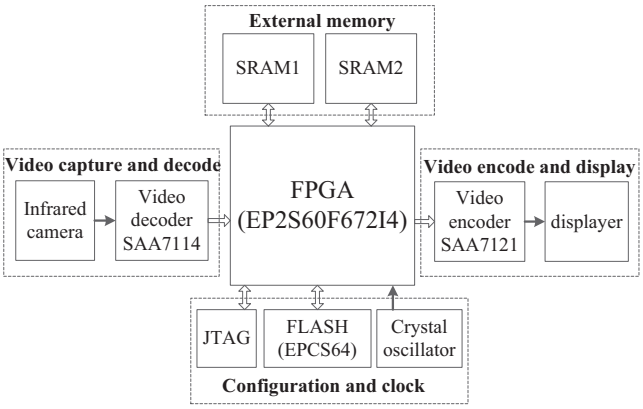


Fig. 2. Hardware platform diagram of system.

The I²C module controls the operating model of the video decoder and encoder. The video input module receives the digital video stream from the video decoder and write into the internal dual-port RAM. NN-NUC module receives the image data from the internal RAM, corrects the FPN and update correction coefficients. The corrected image is stored into the output dual-port RAM2 and then sent to the video encoder for display. The updated correction coefficients are stored in the external RAM by SARM controller module. Besides, a move detection module determines whether a significant movement between adjacent images occurs or not.

The pipeline architecture has a latency of some clock cycles, and the system processes one pixel data each clock cycle after the initial latency. These sub-modules will be discussed one by one in the following sections.

3.3. Correction module

The correction module implements the processing of Eq. (1). The architecture and the data flow of this module are shown in Fig. 4. Firstly, in order to meet the precision requirement, the initial 8-bit integer data format should be translated to a 32-bit floating-point format. Then, a floating-point multiplier and an adder are used to realize the function of non-uniformity correction. Finally, the format of the calculation result should be translated to 8-bit integer again by a Float-to-Fixed block. It takes a latency of 31 cycles to achieve the correction result of an input pixel, and we will get a corrected pixel per each clock cycle after the initial latency. All of blocks in this module are generated by Intellectual Property

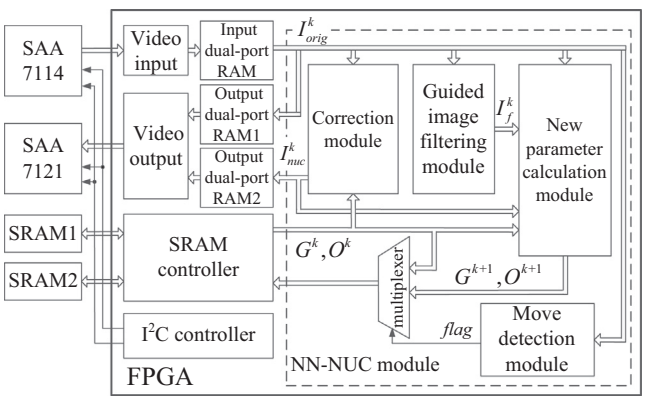


Fig. 3. Top level design diagram of FPGA.

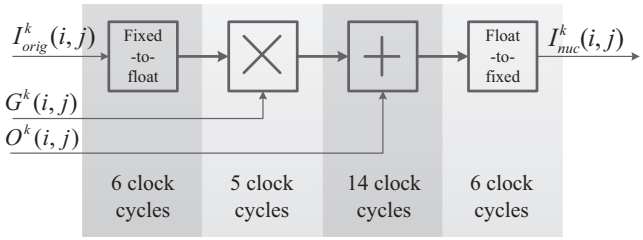


Fig. 4. Data flow diagram of the correction module.

cores (IP cores) in Quartus II provided by Altera and their latencies are also marked in Fig. 4.

3.4. Guided image filter module

The guided image filter module implements the process of Eqs. (9)–(15) in Section 2.2 which contains six stages. The data flow diagram with details latencies of each stage is shown in Fig. 5.

The first stage computes the average $mean_I$ and the variance var_I in a local window of $5 \times 5$with 8-bits integer format. This stage consists of two square operation blocks, two mean filters and a subtraction block. The square and the subtraction operation are made up of combinational logic with no latency.

The mean filter have a latency of $(3 \times n + 5)$ clock cycles, where n means the width of the image. Fig. 6 shows the structure of the mean filter in a window of 5×5, which consists of the line buffer register, the neighborhood buffer register, an adder with 25 inputs and a divider. The line buffer register is used to store pixel data of 5 consecutive lines. The data in the neighborhood buffer register are all pixels in a local window, and their average is calculated by an adder of 25 inputs and a divider. The data format of the adder and the divider is integer and they have no latency. All the latencies of mean filter in this stage are caused by the line buffer register and the neighborhood buffer register.

In the second stage, in order to ensure the accuracy, the data format is translated into 32-bit Float again, which takes a latency of 6 clock cycles. In the third stage, the parameter a is computed by a floating-point adder and divider, which have a latency of 20 clock cycles totally. In the fourth stage, the parameter b is computed by a floating-point multiplier and a subtractor, which have a latency of 19 clock cycles totally.

In the fifth stage, two blocks named para-mean are used to calculate the average of all elements in a window of 5×5 for the parameter matrix a and b. The kernel of these two blocks is a mean filter which is shown in Fig. 6.

The structure of para-mean block is shown in Fig. 7. The data format of matrix a and b is 32-bit float, however the data processed by the mean filter is an 8-bit integer. Thus it is necessary to translate the data format for matrix a and b. In order to keep the data accuracy, the data is multiplied by 10^6 firstly and divided by 10^6 after processing.

In the last stage, the consequence of guided image filtering is calculated by a floating-point multiplier and a floating-point adder. They are IP cores generated in Quartus II and have a latency of 19 clock cycles. In total, it takes a latency of $(6 \times n + 97)$ clock cycles for the whole guided image filtering module, where n means the width of the image. However after the initial latency, it will only take one clock cycle to get a filtered pixel.

3.5. New coefficients computation module

The new parameter computation module implements Eqs. (2) and (3)., The data format is translated to 32-bit Float firstly. The calculation of new gain and offset correction coefficients is implemented by some floating-point multipliers and subtractors which are IP cores generated. The architecture of this process is shown in Fig. 8, which takes 39clocks cycles latency totally.

3.6. Moving detection module

The moving detection module implements the algorithm described in Section 2.3. In this module, these register sets are 16-bit width with initial values 0. Register sets named vech[0:n-1] and vecv[0:m-1] are used to store horizontal and vertical projection vectors respectively.

When the (i,j)-th pixel is read into this module, its intensity is accumulated in both vech[j] and vecv[i]. After receiving all pixels of a complete image, the projection vector is obtained by intercepting the high 8 bits of each register in vech[0:n-1] and vecv[0:m-1]. Before these pixels of next image are coming, the value of vech[0:n-1] and vecv[0:m-1] are respectively copied into the register set named oldvech[0:n-1] and oldvecv[0:m-1] which are used to store these projection vectors of previous image. When the calculation of vech[0:n-1] and vecv[0:m-1] for next frame is completed, the

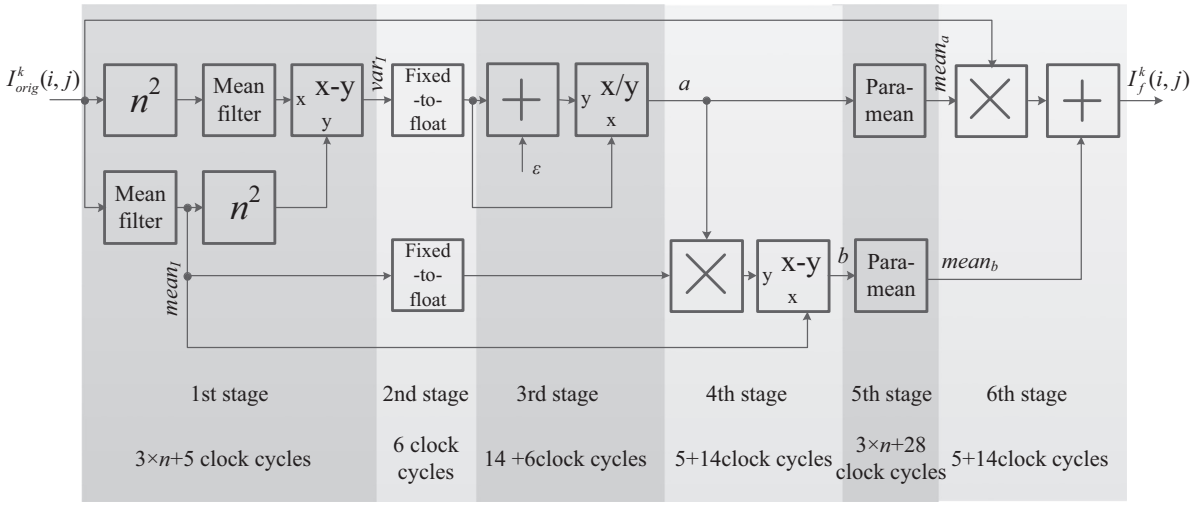


Fig. 5. Data flow diagram of the guided image filtering module.

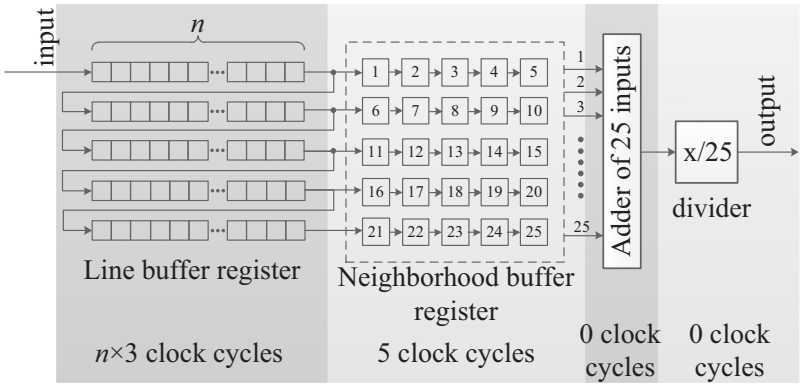


Fig. 6. Data flow diagram of the mean filter.

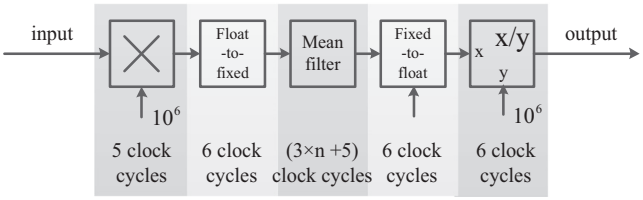


Fig. 7. Data flow diagram of the para-mean block.

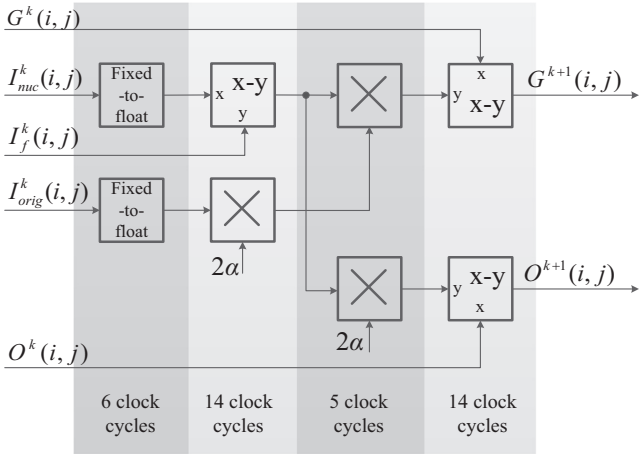


Fig. 8. Data flow diagram of the new parameter computation module.

horizontal difference value D_h^k can be acquired by summating the subtraction of the corresponding register between vech[0:n-1] and oldvech[0:n-1], and the vertical difference value D_v^k can be achieved by summating the subtraction of the corresponding register between vecv[0:m-1] and oldvecv[0:m-1]. It is considered that there is a significant movement between the adjacent image if $D_h^k > -T$ or $D_v^k > T$, and set the variable identifier *flag* as 1, otherwise the variable *flag* is 0 which means a significant movement don't occur.

Based on this identifier, a multiplexer behind the moving detection module controls the update of correction coefficients. If the variable *flag* is equal to 1, values of G^{k+1} and O^{k+1} are chosen to write into the external SRAMs. Otherwise, the values of G^k and O^k remain unchanged.

4. Experimental results and analysis

4.1. Experiment image sequences

Two infrared sequences are utilized to verify the performance of the proposed algorithm. The first test sequence is captured from an infrared camera without any processing and an example frame is shown in Fig. 9. The content of this sequence is a moving people in front of a static background which shows that the non-uniformity really exists and has greatly influence on the quality of the image. The second test sequence is achieved form VIVID data set which acquired by a corrected infrared camera. The experimental data is generated by adding an artificial non-uniformity into a group of clear images which are shown in Fig. 10. It is convenient to analyze the performance of NUC algorithm by comparing the corrected image with the clear one.

In the proposed algorithm, the guided filter window size r and fuzzy degree factor ε are set to 2 and 15 respectively. These parameters are determined by the non-uniformity degree and a larger value is need for a serious FPN. The learning rateα is set as 2×10^{-6}. This parameter is utilized to control the learning rate. A larger value can speed up the correction process but also will cause the algorithm to diverge. The threshold of motion detection T is equal to 1000, the gray range of image is set from 0 to 255.

4.2. Experimental results

Two groups of the correction experimental results are shown in Figs. 11and 12 respectively. It indicates the advantage of the proposed algorithm by comparing it with the neural network algorithm based on 4-neighbor and 5×5 mean filter respectively.

In Scribner's NN-NUC algorithm, the average of 4 neighbors is used to estimate the desired value of the pixel. It plays a role in the correction, however the residual non-uniformity is still visible as shown in Figs. 11(b) and 12(b). If the desired pixel is estimated by calculating the average in a window of 5×5, the quality of the image will be greatly improved, however it also causes the serious ghost artifacts as shown in Fig. 11(c). Figs. 11(d) and 12(d) are results by the proposed algorithm, and its advantage is obvious and less ghost artifacts can be seen from the results.

Fig. 9. An example frame from the 1st test sequence.

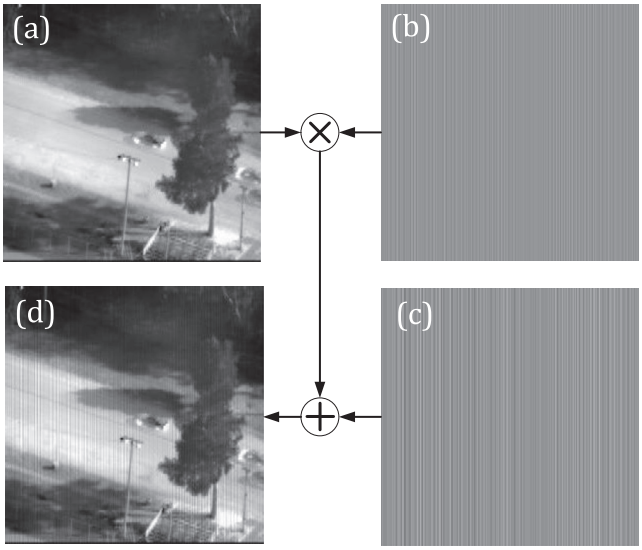


Fig. 10. Generation of the 2nd test sequence. (a) the clear image, (b) the gain component, (c) the offset component, (d) the image with artificial non-uniformity.

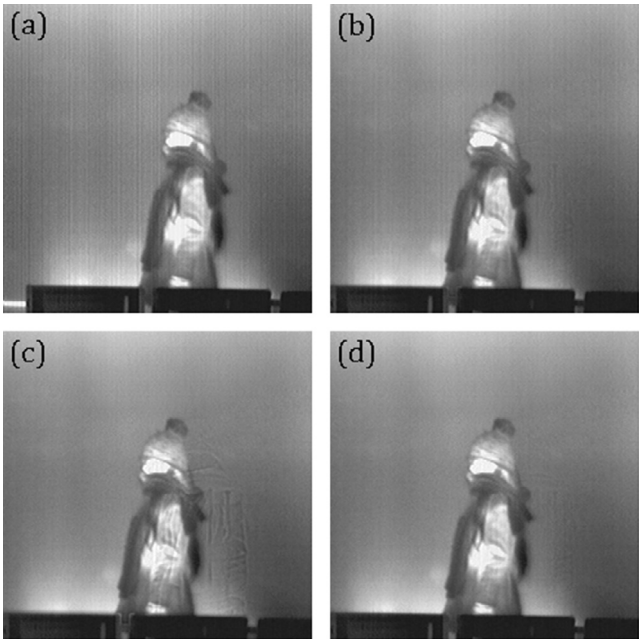


Fig. 11. Simulation results with the first sequence. (a) original image, (b) result of NN based on 4 neighbors, (c) result of NN based on 5×5 window, (d) result of proposed algorithm.

It not only improves the quality of the image greatly, but also overcomes the drawback of artificial ghosting.

4.3. Evaluation of NUC

In this section, two objective indicators, smoothness ρ and Root Mean Squared Error (RMSE), are used to evaluate the performance of proposed algorithm. The smoothness ρ is defined as [24]

$$\rho = \frac{\|h_1 * I_{nuc}\|_1 + \|h_2 * I_{nuc}\|_1}{\|I_{nuc}\|_1} \tag{22}$$

where I_{nuc} represents the corrected image, h_1 is an edge-detection mask which uses horizontal template $[1,-1]$, h_2 is the transposition of h_1, operator $*$ means the operation of convolution, and $\|\ \|_1$ represents L_1 norm. The result curves of smoothness of sequence 1 and 2 are shown in Fig. 13.

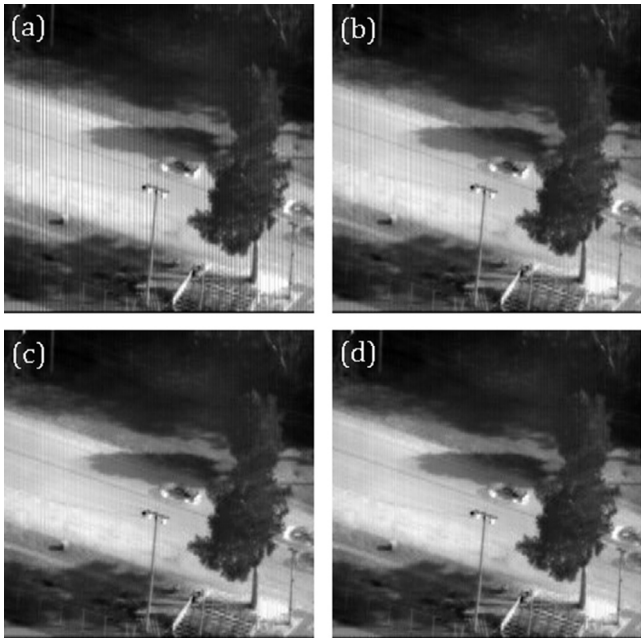


Fig. 12. Simulation results with the 2nd sequence. (a) original image, (b) result of NN based on 4 neighbors, (c) result of NN based on 5×5 window, (d) result of proposed algorithm.

The metric ρ is used to evaluate the ability of the proposed algorithm to remove FPN. A smaller smoothness indicates a better correction performance. As can be seen from the Fig. 13, the smoothness curves of proposed algorithm are much smaller than others, which indicates the proposed algorithm has better performance. A smaller ρ indicates that the correction result is smoother than before. Nevertheless, a smaller smoothness means that the corrected image loses more details. In order to evaluate the ability of the proposed algorithm to restore the original image, another metric RMSE is introduced which is defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(I_{nuc}(n) - I_{clear}(n))^2} \tag{23}$$

where I_{nuc} represents the corrected image, I_{clear} represents the real image without FPN, N is the total number of the pixel in an image. The RMSE can reflect the accuracy of the proposed algorithm, which means the original image without FPN is essential. Thus the second simulation sequence is evaluated by RMSE and the result is shown in Fig. 14.

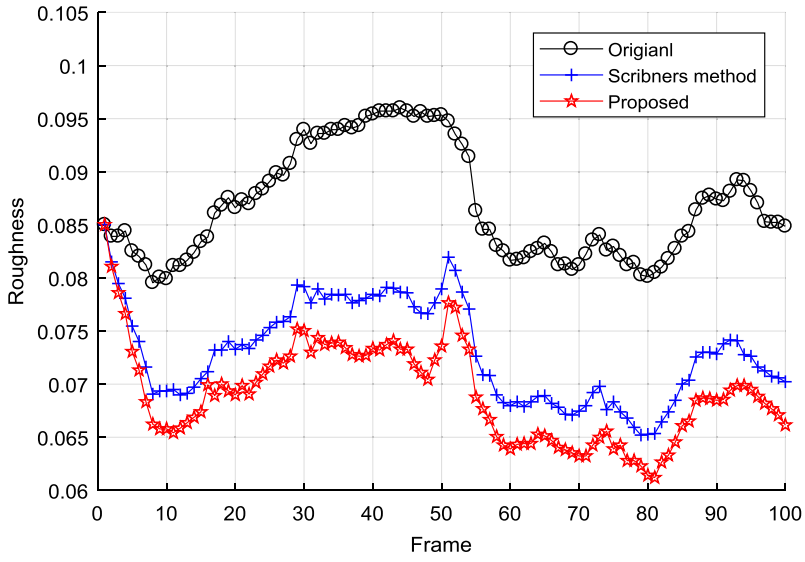
As can be seen from Fig. 14, the RMSE curve of proposed is smaller than the Scribner's algorithm after 10th frame. Besides, in terms of the proposed algorithm, both the smoothness and RMSE have the faster rate of decline in the first 10th frames, which means convergence speed is faster than the Scribner's one.

In summary, both the smoothness and RMSE show that the proposed algorithm has a better performance than original algorithm. The proposed algorithm has better performance both in the visual and the quantitative condition.

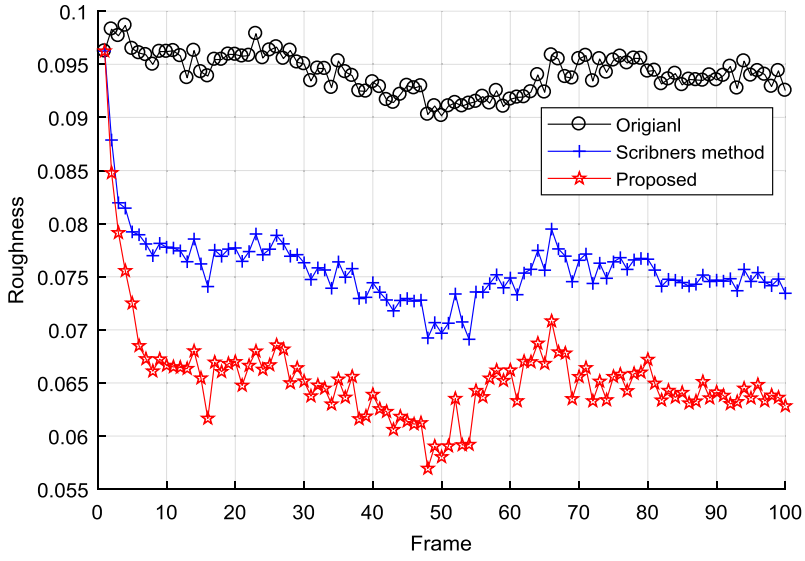
5. Hardware performance analyses

5.1. Correction results on hardware platform

In order to evaluate the performance of the proposed algorithm on hardware platform, a comparative experiment is designed in



(a) sequence 1 results



(b) sequence2 results

Fig. 13. Smoothness curves.

this section. The edited image is sent to our hardware platform form a DVD player with PAL format. After processing on the proposed hardware platform, the corrected image is collected by a video capture card and each image is spliced into two parts. The left image is the original image, and the right one is the corrected image.

In order to evaluate the effect of the motion detection module, it is blocked firstly. In this sequence, some video clips have static scenes which are utilized to evaluate the performance and the correction result is shown in Fig. 15. As can be seen from the correction result, through the FPN is removed, the image becomes fuzzy and most of details cannot be seen any more, which really reduces image quality.

Then, the motion detection module is added into the program again, the correction result is shown in Fig. 16. The motion detection module can accurately determine the update of coefficients. The corrected image will remain clear even if the scene is static. Thus, it is verified that the moving detection algorithm is useful to prevent the blurring of stationary scene. Through the proposed algorithm, the quality of the image has been greatly improved after processing in the hardware system.

5.2. Analysis of FPGA resource usage

The resource usage is one of the important factors for a FPGA based hardware design. The resource usage rate is summarized in Table 1. Adaptive look-up tables (ALUTs) and dedicated logic registers are used to implement the combinatorial logic and the temporal logic of NUC algorithm. I/O pins are used to connect with external storage chips and interface devices. The internal block memories of FPGA are used to generate the input and the output dual-port RAMs. DSP block 9-bit elements are mainly used in IP cores of floating-point calculation. PLLs are used to generate the 12.5 MHz system clock and the 27 MHz interface clock from a 50 MHz external crystal oscillator.

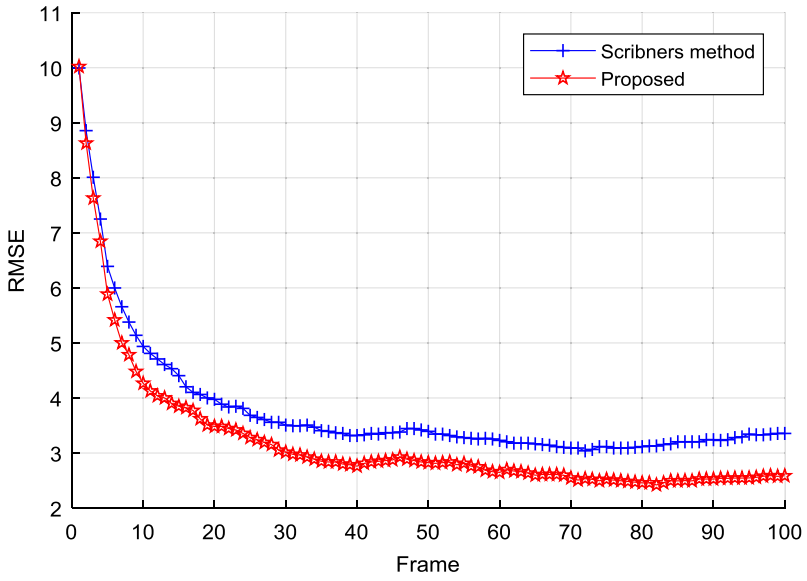


Fig. 14. RMSE curves of the second sequence.

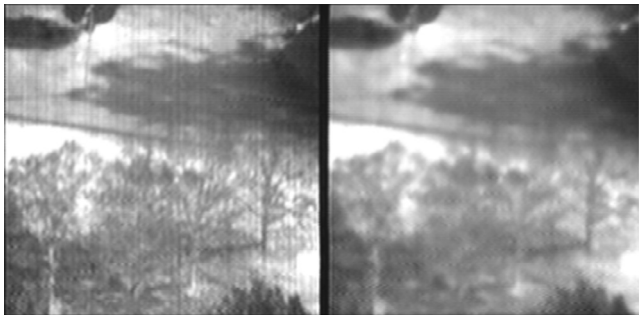


Fig. 15. Result without moving detection module.

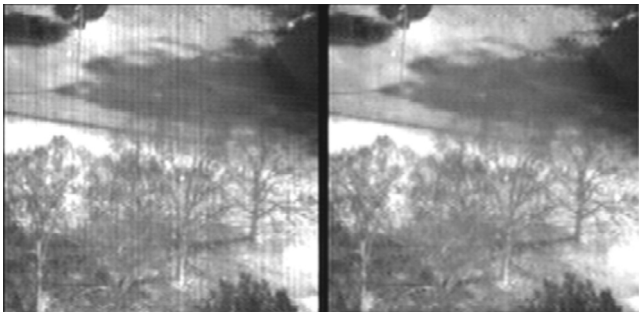


Fig. 16. Result with moving detection module.

Table 1
Resource usage of the overall design.

Resource type	Design usage	Total resources	Usage rate (%)
Combinational ALUTs	19596	48352	41
Dedicated logic registers	20118	48352	42
I/O pins	136	493	28
Block memory bits	1843518	2544192	72
DSP block 9-bit elements	148	288	51
PLLs	2	6	33

As shown in Table 1, it indicates that the proposed design has an effective utilization in EP2S60. The most used resource is Block memory bits, which is up to 72% and the usage of other resources does not exceed 50 percent. There are also some available resources to implement other image processing.

The usage rate of logic elements (LEs) for each module is shown in Fig. 17, which is calculated by the design usage of ALUTs. It indicated that the most LEs are consumed by the guided image filtering module, because this module contains the most fixed-point and float-point arithmetic operations. These projection vectors in moving detection module are implemented by using dedicated logic register, which leads this module also consumes a large number of LEs. Other modules with a lot of arithmetic operations also have a large usage of LEs.

5.3. Computation time

For the real time system, the computation time is another important performance metric. In our system, the correction module and the guided image filtering module are collateral. Thus, it is only needed to consider the slower one when the latency is calculated. For a 256×256 image, it will take a latency of 1672 (1633 + 39) clock cycles totally in our system. After the initial latency, it will only take one clock cycle to achieve a corrected pixel or a correction coefficient. There are 65536 (256×256) pixels in a complete image, and it will take 67208 (1672 + 65536) clock cycles for a complete image to accomplish all calculations. The frequency of system clock is 12.5 MHz, and the clock period is 80 ns. It indicates that it will take less than 5.4 ms (80 ns $\times$ 67208) for our system to process one frame.

The non-uniformity correction is usually acted as a pre-processing operation in a real-time IRFPA system, thus it is required to spend as little time as possible. For a video at 25fps, the available time of each frame is 40 ms. In our system, much time remained which could be implemented other processing. In addition, the performance of our system is currently limited by the frame rate of the input video, however can be readily improved to a frame rate of more than 180 fps .

Compared with other typical embedded systems, such as ARM, DSP and ASIC, the proposed platform is more efficient and economical. The architecture of ARM chip limits its use in the field of a large number of mathematical operations. Though, increasing the number of cores is a viable solution. This will result in large power consumption. For a DSP based platform, it is totally a serial computing system, which means that less than one frame latency is

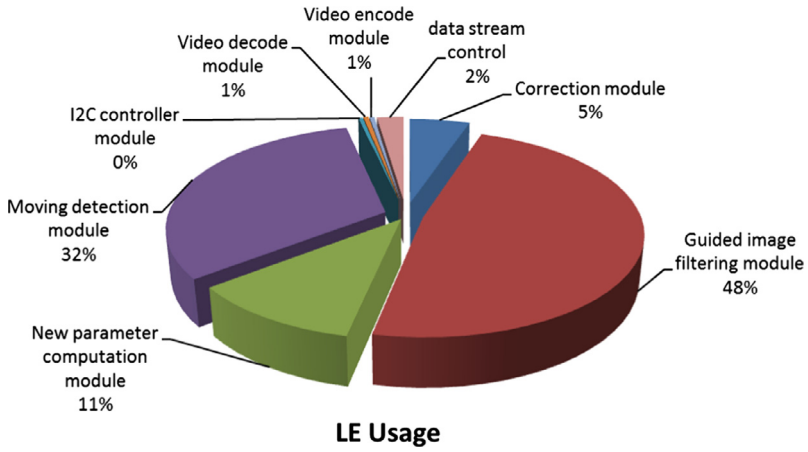


Fig. 17. LE usage percentage of modules.

unavoidable. ASIC is a kind of integrated circuit customized for a particular use, which means the design and production process is extremely time-consuming and expensive.

6. Conclusions

This paper proposed an improved NUC algorithm with less ghosting and better performance. Besides, an FPGA-based hardware design is also proposed, which is very efficient and economical. In this algorithm, the guided image filter is utilized to estimate the desired image which will overcome the drawback of ghosting appears in the original algorithm, and the projection based moving detection algorithm is utilized to determine the updating of coefficients which will prevent the image blurring during static scene. In this paper, the hardware platform and the design module in FPGA are also described in detail according to the top-down design rule. All of algorithm design modules are designed with a pipelined architecture, and the structure and the latency of them are fully discussed as well. The advantage of the proposed algorithm is analyzed, and two different image sequences are used to verify the performance of algorithm. Experimental results indicated that the proposed algorithm has better performance than the previous NUC algorithm. However, different threshold T will affect the correction result seriously. To select threshold adaptively is the focus of research in the future.

The hardware design for the proposed algorithm is implemented on the Altera Stratix II FPGA EP2S60F672I4 chip. The proposed algorithm consumes only less than 45% of LEs and 72% of available block memories in the chip. In addition, the processing rate can be raised to 180fps (256×256 pixels) by increasing the frame rate of input infrared video. In summary, the proposed hardware design is an effective system for non-uniformity correction.

Conflict of interest statement

We declared that they have no conflicts of interest to this work. we do not have any commercial or associative interest that represents a conflict of interest in connection with the manuscript submitted.

Acknowledgement

We would like to express our sincere appreciation to the anonymous reviewers for their insightful and valuable comments, which have greatly helped us in improving the quality of the paper. This work is partially supported by the National Natural Science Foundation of China (Nos. 61675160, 61265006 and 61401343), and the 863 Program of China (2014AA8098089C).

References

[1] D.A. Scribner, K.A. Sarkady, J.T. Caulfield, M.R. Kruer, G. Katz, C. Gridley, C. Herman, Nonuniformity correction for staring IR focal plane arrays using scene-based techniques, Int. Soc. Opt. Photon. (1990) 224–233.
[2] H.X. Zhou, S.Q. Liu, R. Lai, D.B. Wang, Y.B. Cheng, Solution for the nonuniformity correction of infrared focal plane arrays, Appl. Optics 44 (2005) 2928–2932.
[3] T. Sosnowski, G. Bieszczad, M. Kastek, H. Madura, Processing of the image from infrared focal plane array using FPGA-based system, in: Mixed Design of Integrated Circuits and Systems (MIXDES), 2010 Proceedings of the 17th International Conference, IEEE, 2010, pp. 581–586.
[4] J.G. Harris, Y.M. Chiang, Nonuniformity correction of infrared image sequences using the constant-statistics constraint, IEEE Trans. Image Process. 8 (1999) 1148–1151.
[5] S. Torres, R. Reeves, M. Hayat, Scene-based nonuniformity correction method using constant-range: Performance and analysis, in: Proceedings of 6th World Multiconference on Systemics, Cybernetics and Informatics, 2002, pp. 224–229.
[6] R. Redlich, M. Figueroa, S.N. Torres, J.E. Pezoa, Embedded nonuniformity correction in infrared focal plane arrays using the Constant Range algorithm, Infrared Phys. Technol. 69 (2015) 164–173.
[7] S.E. Godoy, J.E. Pezoa, S.N. Torres, Noise-cancellation-based nonuniformity correction algorithm for infrared focal-plane arrays, Appl. Optics 47 (2008) 5394–5399.
[8] W.X. Qian, Q. Chen, G.H. Gu, Space low-pass and temporal high-pass nonuniformity correction algorithm, Opt. Rev. 17 (2010) 24–29.
[9] C. Zuo, Q.A. Chen, G.H. Gu, W.X. Qian, New temporal high-pass filter nonuniformity correction based on bilateral filter, Opt. Rev. 18 (2011) 197–202.
[10] J.Q. Bai, Q.A. Chen, W.X. Qian, X.Y. Wang, Ghosting reduction in scene-based nonuniformity correction of infrared image sequences, China Opt. Lett. 8 (2010) 1113–1116.
[11] S.N. Torres, M.M. Hayat, Kalman filtering for adaptive nonuniformity correction in infrared focal-plane arrays, J. Opt. Soc. Am. A 20 (2003) 470–480.
[12] H.X. Zhou, H.L. Qing, L.P. Bai, Q.C. Liu, X. Geng, B.J. Wang, Nonuniformity correction algorithm with nonlinear model for infrared focal plane arrays, Infrared Phys. Technol. 53 (2010) 10–16.
[13] C. San Martin, J.E. Pezoa, S. Torres, P. Meza, D. Gutierrez, Block-recursive filtering for offset nonuniformity estimation in infrared imaging systems: Theory and implementation, Pattern Recogn. Lett. 31 (2010) 478–483.
[14] H.X. Zhou, H.L. Qin, Y.B. Jian, B.J. Wang, S.Q. Liu, Improved Kalman-filter nonuniformity correction algorithm for infrared focal plane arrays, Infrared Phys. Technol. 51 (2008) 528–531.
[15] R. Sheng-Hui, Z. Hui-Xin, Q. Han-Lin, L. Rui, Q. Kun, Nonuniformity correction for an infrared focal plane array based on diamond search block matching, J. Opt. Soc. Am. A 33 (2016) 938–946.
[16] C. Zuo, Q. Chen, G.H. Gu, X.B. Sui, J.L. Ren, Improved interframe registration based nonuniformity correction for focal plane arrays, Infrared Phys. Technol. 55 (2012) 263–269.
[17] Y.J. Liu, H. Zhu, Y.G. Zhao, Scene-based nonuniformity correction technique for infrared focal-plane arrays, Appl. Optics 48 (2009) 2364–2372.
[18] H. Yu, Z.-J. Zhang, C.-S. Wang, An improved retina-like nonuniformity correction for infrared focal-plane array, Infrared Phys. Technol. 73 (2015) 62–72.

[19] J.F. Zhao, X.M. Gao, Y.T. Chen, H.J. Feng, Z.H. Xu, Q. Li, Fast iterative adaptive nonuniformity correction with gradient minimization for infrared focal plane arrays, Infrared Phys. Technol. 65 (2014) 87–93.
[20] F. Fan, Y. Ma, B. Zhou, Y. Fang, J.H. Han, Z. Liu, A Scene Based Nonuniformity Correction Algorithm for Line Scanning Infrared Image, Opt. Rev. 21 (2014) 778–786.
[21] N. Celedon, R. Redlich, M. Figueroa, FPGA-based neural network for nonuniformity correction on infrared focal plane arrays, in: Euromicro Conference on Digital System Design, 2012, pp. 193–200.
[22] K.M. He, J. Sun, X.O. Tang, Guided Image Filtering, IEEE Trans. Pattern Anal. 35 (2013) 1397–1409.
[23] S.H. Rong, H.X. Zhou, H.L. Qin, R. Lai, K. Qian, Guided filter and adaptive learning rate based non-uniformity correction algorithm for infrared focal plane array, Infrared Phys. Technol. 76 (2016) 691–697.
[24] M.M. Hayat, S.N. Torres, E. Armstrong, S.C. Cain, B. Yasuda, Statistical algorithm for nonuniformity correction in focal-plane arrays, Appl. Optics 38 (1999) 772–780.