Efficient Hardware Calculation of Running Statistics

Donald G. Bailey*, Michael J. Klaiber†
*School of Engineering and Advanced Technology, Massey University, Palmerston North
Email:d.g.bailey@massey.ac.nz
†Institute for Parallel and Distributed Systems, University of Stuttgart
Email: michael.klaiber@ipvs.uni-stuttgart.de

***Abstract*—Calculation of mean, variance and standard deviation are often required for segmentation or feature extraction. In image processing, often an integer approximation is adequate. Conventional methods require division and square root operations, which are expensive to realize in hardware in terms of both the amount of required resources and latency. A new class of iterative algorithms is developed based on integer arithmetic. An implementation of the algorithms as a hardware architecture for a Field-Programmable Gate Array (FPGA) is compared with architectures using the conventional approach, which shows a significantly reduced latency while using less hardware resources.**

I. Introduction

In many applications in image processing such as segmentation and feature extraction statistics like the mean, the variance and the standard deviation are required. Especially in segmentation, the decision process is often based on the statistics collected on a certain image region and therefore the calculation of these statistics has a major influence on the performance of the whole algorithm. In hardware feature extraction architectures, the calculation of object features often has to be finished within one clock cycle [1], [2]. In these cases where statistics of the image objects are extracted, a high operation frequency for calculation is necessary not to decrease the throughput of the whole feature extraction architecture.

Several previously proposed methods use floating point arithmetic to calculate standard deviation [3], [4]. The method in [4] extends the method in [3] but still relies on floating point arithmetic and is therefore expensive to realize as an FPGA architecture. The method proposed in [5] describes a parallelization approach for calculating sample variance, which is of advantage for calculating statistics of big sets of data, which are available in memory, but less relevant for running statistics, where the elements are received serially with the goal of calculating the statistics with low latency. In [6] a hardware architecture is described to calculate the local variance of a fixed-size window. The the method proposed in this paper is able to calculate running statistics over an arbitrary number of pixels. The architecture in [6] can be realized by division by a constant, which reduces hardware cost for division at the cost of flexibility and functionality compared to an integer division in the architecture in this paper. The proposed approach enables the use of integer arithmetic, which is inexpensive in terms of resources and achievable maximum operation frequency on FPGAs.

978-1-4799-0883-7/13/$31.00 ©2013 IEEE

The definitions of the mean and variance of a set X consisting of N elements are

$$\mu_i = \frac{1}{N_i} \times \sum^{N_i} x_i \tag{1}$$

$$\sigma_i^2 = \frac{1}{N_i - 1} \times \sum^{N_i} (x_i - \mu_i)^2, \tag{2}$$

which require several iterative additions followed by a division for mean, an additional multiplication for variance calculations, and an additional square root for the standard deviation. To achieve the goal of calculating the running statistics, which allows the statistics to be accessed after each new element is added to the set, this naive approach is impractical due to the fact that the variance can only be calculated in a second pass after all elements of the set X have been passed once and the mean value μ has been calculated.

The running variance and standard deviation can be calculated according to

$$\sigma_i^2 = \frac{1}{N_i - 1} \times \sum^{N_i} (x_i^2) - \mu_i^2. \tag{3}$$

Using this equation, σ^2 is accessible immediately after the most recent element x has been added. Even though these solutions work well on general purpose processors, divisions and square root calculations are expensive in terms of resources and latency when realizing a dedicated hardware architecture. Furthermore in many image processing applications integer accuracy is acceptable. Therefore, no floating point operations with high precision are required, which can be reflected in hardware architectures using fewer resources or working with a lower latency.

In this paper we derive a method for efficient calculation of running statistics adapted to the needs of image processing and the realization of a hardware architecture, which carries out this method using fewer hardware resources than conventional methods while maintaining a high operation frequency. This is achieved by replacing operations which are expensive to realize in hardware with combinational circuits and iterative approaches, which achieve an acceptable accuracy at higher speed.

The methods for calculation of running mean, variance and standard deviation of image data are presented in Section II. Section III shows how efficient hardware architectures

can be built from these methods and Section IV gives the implementation results for those architectures on an FPGA and discusses the results.

II. Running Statistics Calculation

A. Mean and Variance calculation

The method proposed for calculating the mean is inspired by Welford's method [3] and is based on the idea of scaling the values to an origin close to the mean. This process is referred to as a renormalization step. In this way it is possible to shift the origin to an approximation of the mean value, which provides an approximation accessible after every iteration without additional calculation. It uses integer arithmetic, contrary to the solution in [3] which requires floating point operations, making the new approach better suited for realization as a dedicated digital circuit.

The calculation of the running mean works as follows. The partial sum P stores the sum of the already processed elements relative to the origin m.

$$P'_i = P_{i-1} + (x - m_{i-1}) \tag{4}$$

To keep the origin close to the mean of the past elements, the origin must be shifted by

$$\triangle_m = \frac{P'_i}{N_i} \tag{5}$$

$$m_i = m_{i-1} + \triangle_m \tag{6}$$

to include the most recent element. Since this changes the origin, the partial sum P has to be adapted as well.

$$P_i = P'_i - N_i \times \triangle_m \tag{7}$$

By making $\triangle_m$ an integer, i.e. using an integer division, the mean μ is represented by m as an integer approximation and a fractional part $\frac{P}{N}$.

$$\mu_i = m_i + \frac{P_i}{N_i} \tag{8}$$

The fraction part can always be made smaller than ± 0.5, allowing m_i to be used as an approximation of the running mean value μ_i.

A similar method can be used to calculate the running variance using only integer arithmetic. First, we define the variance σ^2 to consist of an integer part V and a fractional part W,

$$\sigma_i^2 \triangleq V_i + \frac{W_i}{N_i - 1} - (\frac{N_i}{N_i - 1} \times (\mu_i - m_i)^2) \tag{9}$$

where the last term is neglected (since $\mu_i \approx m_i$) and

$$W_i = V_i + \sum^{N_i} ((x_i - m_i)^2 - V_i) \tag{10}$$

The integer and fractional parts of the variance are updated as follows. In the first step the most recent element x_i has to be included in W.

$$W'_i = W_{i-1} + (x_i - m_{i-1})^2 - V_{i-1} \tag{11}$$

This is then renormalized to the new origin

$$W''_i = W'_i - 2\Delta_m \times P'_i + N_i \times \Delta_m^2 \tag{12}$$

The excess of the fractional part

$$\Delta_v = \frac{W''_i}{N_i - 1} \tag{13}$$

is finally used to adjust V to within ± 0.5 of σ^2.

$$\begin{aligned} V_i &= V_{i-1} + \Delta_v \\ W_i &= W''_i - (N_i - 1) \times \Delta_v \end{aligned} \tag{14}$$

Although the described methods still need division operations, they are more robust against less accurate division. The divisions in Equations 5 and 13 can be approximated to a power of two with resulting multiplications in Equations 7, 12 and 14 being shift operations, which is preferable in terms of hardware cost. The proposed method therefore performs the divisions over several iterations, instead of performing a full division for each new element added to the set. There is no loss in accuracy when calculating the mean and variance for the whole set of elements using this method. Even though the difference of the approximations m and V to the actual mean μ and variance σ^2 is greater than one at some points, Equations 8 and 9 still hold with the approximations rapidly converging to their true values for natural images, as we will show in Section IV.

B. Standard Deviation

For calculation of the standard deviation σ, the square root of the variance has to be extracted. Since the square root operation, like a division, requires a lot of hardware resources, an iterative approach is proposed to calculate it. Let S be an integer approximation of the standard deviation σ, with remainder R.

$$V_i \triangleq S_i^2 + R_i \tag{15}$$

On update,

$$R'_i = R_{i-1} + \triangle_v. \tag{16}$$

Since R is the remainder its absolute value has to be smaller than S. When R exceeds the approximation S, S has to be adapted. For this adaption the Newton-Raphson iteration is used.

$$\triangle_S = [\frac{R}{2S}] \tag{17}$$

$$\begin{aligned} S_i &= S_{i-1} + \triangle_S \\ R_i &= R'_i - 2 \times S_{i-1} \times \triangle_S - \triangle_S^2 \end{aligned} \tag{18}$$

This iteration step brings S closer to σ with every iteration.

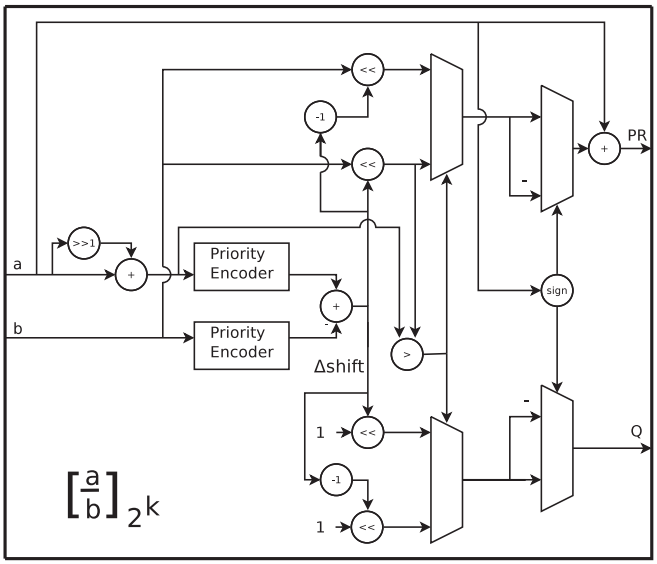

Fig. 1. Hardware architecture for division to the closest power of two.

III. Hardware Architectures for Calculation of Running Statistics

A. Division by next power of two operator

Realizing conventional algorithms for division, such as restoring division, non-restoring division or the SRT algorithm requires a significant of hardware resources. The latency of the division architecture depends on the width of the dividend and divisor and can become significantly long [7], [8]. In many image processing applications, integer accuracy is sufficient. A radix-N division can be replaced by an approximation, which can be built more efficiently using digital hardware resources. Such an approximation is the closest power of two of the quotient of two numbers. For this operation the $[\frac{a}{b}]_{2^k}$ operator is used. The hardware architecture to realize this is depicted in Figure 1. In order to achieve rounding, the dividend is multiplied by 1.5, realized by a shift and add operation. After this two priority encoders examine the dividend a and the divisor b and search for the first 1 in case the number is positive or the first 0 if the number is negative. The use of priority encoders enables a gain in the combinational path compared to conventional division algorithms, because there is no need for cascaded additions and subtractions of partial remainders. The difference of the results of the priority encoders $\triangle_{shift}$ indicates the power of the quotient of a and b, but can be off by one since it is possible that the dividend $|a| < (b \times 2^{\triangle_{shift}})$. In order to prevent this, another comparison is required, which decrements $\triangle_{shift}$ according to the sign of the dividend and the divisor. The quotient bit Q is acquired by shifting a '1' by $\triangle_{shift}$ to the left and taking the complement of it depending of the sign of dividend and divisor. The partial remainder PR of the division can be determined by shifting the divisor b in an analogous manner and be reused for the dividend of the following division.

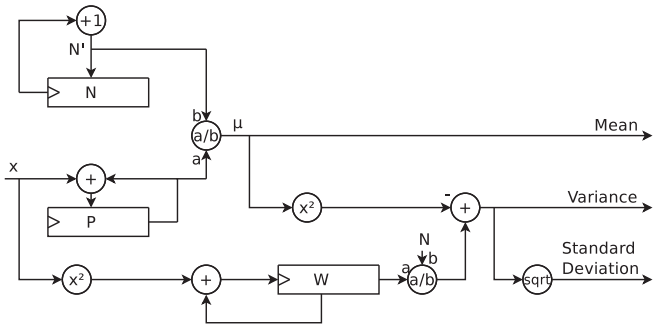

Fig. 2. Circuit calculating the mean, variance and standard deviation according to the conventional method.

B. Hardware architectures for running statistics

An architecture for calculating running statistics with the conventional approach is depicted in Figure 2. In this architecture, the division uses combinational logic with a non-restoring division. Among conventional division architectures the non-restoring approach has been shown to create a short combinational path and least resource utilization [8]. The square root calculation is carried out also by a combinational circuit performing a non-restoring calculation of the square root [9].

The hardware architecture in Figure 3 shows the data path of the method for calculation of running mean and running variance proposed in Section II at the register transfer level. This architecture uses the proposed division to the next power of two instead of a radix N-division. For the proposed method this means that Equations 6 and 13 have to be replaced by

$$\begin{aligned} \Delta_m &= [\frac{P_i'}{N_i}]_{2^k} \\ \Delta_v &= [\frac{W_i''}{N_i - 1}]_{2^k}. \end{aligned} \tag{19}$$

This change reduces the accuracy of the calculation and therefore the fractional part of the mean $\triangle_m$ and the fractional part of the variance $\triangle_v$ can exceed one, which results in a less accurate integer approximation. Nevertheless the integer part converges towards actual value within just a few pixels, in which case they do not differ too much and hence compensates the decreased accuracy.

In order to get the running standard deviation the square root of the variance has to be extracted. The architecture in Figure 4 depicts the data path for the iterative method of calculating the square root proposed in Section II. In this architecture the next power of two of the quotient is again used due to hardware efficiency. This approximation sacrifices the quadratic convergence of the Newton-Raphson method for a linear convergence of regular division.

IV. Experimental Results and Discussion

In this section the methods for calculating running statistics and the hardware architectures built using these methods are evaluated and analyzed.

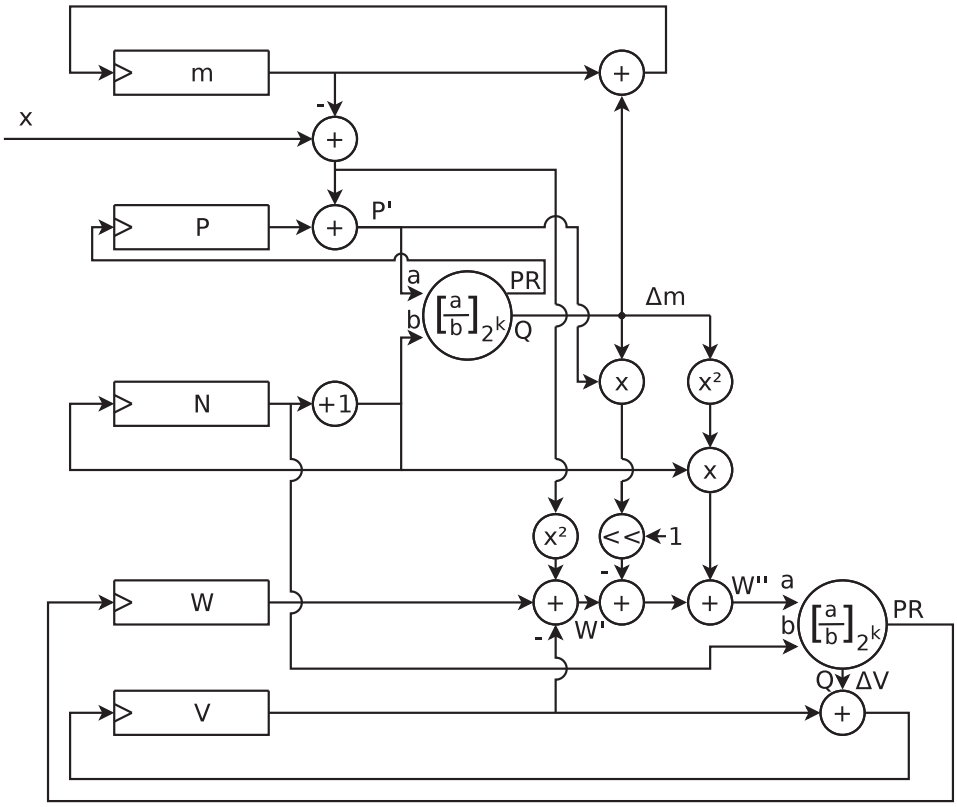

Fig. 3. Architecture for calculation of mean and variance with the proposed method.

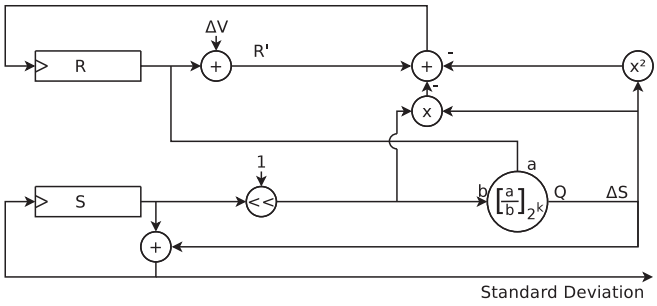

Fig. 4. Architecture for calculating approximation of standard deviation with the proposed method.

A. *Evaluation of accuracy of the hardware architecture*

The proposed method is evaluated on different test images [10], where each one represents a different class of images. For analysis of the accuracy of the proposed method to calculate of running statistics, the pixel values of the middle row of the test image are used traversing the image from left to right. Only one iteration is performed per pixel. After each pixel, results are compared to the results of the conventional calculation. Figure 5 shows the test images together with the plots of the difference of the running statistics values between the conventional and proposed methods.

For the first couple of pixels, the difference is expectably high. This results from the lost accuracy due to the division to the next power of two and the small number of samples. Nevertheless this converges to the value of the conventional method within just a few pixels. The same can be observed at rapid changes of the pixels' intensity in the image. The difference to the conventional method first increases, but converges to the actual value within just less than 5 pixels for the observed cases. For the observed images the proposed method shows a deviation for calculating the running mean which are in most cases below 2% difference to the actual mean.

As the diagrams in Figure 5 indicate, single intermediate results of the calculation differ from the actual statistic values, but recover within just a few pixels and converge towards the desired value. This fast convergence of the values enables the method to provide good approximations of the statistic values.

B. *Comparison of hardware resource requirements*

The hardware architectures introduced in Section III were implemented on a on Xilinx Virtex 6 XC6VLX240T-2FF1759. The required FPGA resources are given in Table II for a hardware architecture using the conventional approach and in Table I for the proposed approach. When comparing the resource utilization for both architectures one can observe that the conventional approach requires significantly more look-up tables (LUTs) than the proposed architecture. This results mainly from the fact that the conventional architecture has dedicated circuits for division and square root operations,

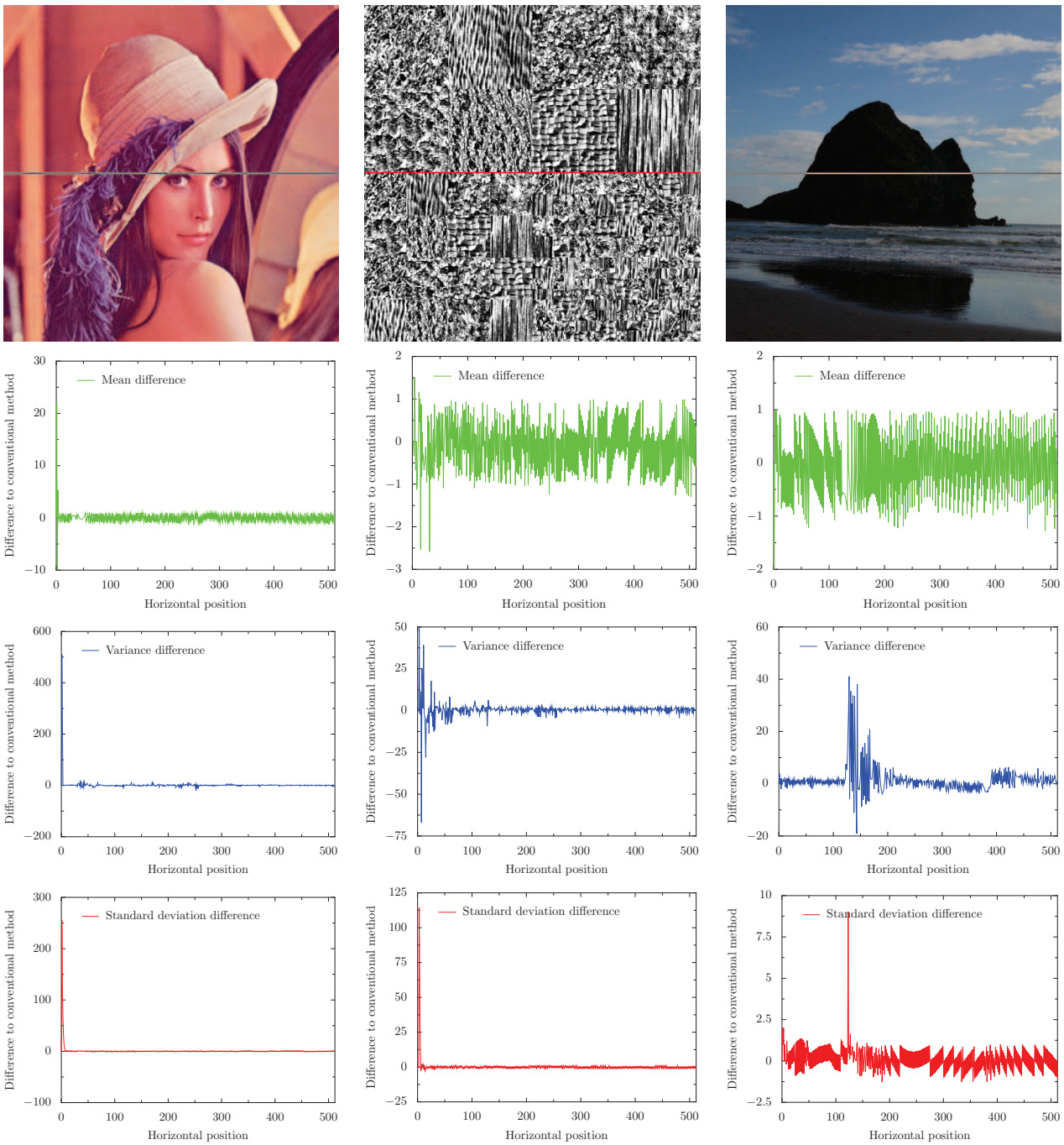

Fig. 5. Difference of the absolute values of running mean, variance and standard deviation of conventional and the proposed method.

TABLE I
RESOURCE REQUIREMENTS OF FPGA HARDWARE ARCHITECTURE CALCULATING RUNNING STATISTICS WITH THE PROPOSED APPROACH ON A XILINX VIRTEX 6 XC6VLX240T-2FF1759.

	Mean			
Image size	0.3 MP		1 MP	
Shifter	Comb	HW	Comb	HW
Registers	75	75	78	78
LUTs	495	314	515	323
DSP48E1s	0	4	0	4
f_{max}[MHz]	109.9	75.5	109.4	75.3
Framerate [FPS]	327	226	109	75

	Mean + Variance			
Image size	0.3 MP		1 MP	
Shifter	Comb	HW	Comb	HW
Registers	134	131	139	136
LUTs	979	597	1005	616
DSP48E1s	12	20	12	20
f_{max}[MHz]	38.9	29.4	38.7	29.4
Framerate [FPS]	116	88	38	29

	Mean + Variance + Standard Deviation			
Image size	0.3 MP		1 MP	
Shifter	Comb	HW	Comb	HW
Registers	165	159	170	165
LUTs	1430	842	1485	869
DSP48E1s	15	25	15	25
f_{max}[MHz]	29.4	21.4	29.4	21.4
Framerate [FPS]	88	64	29	21

while the proposed architecture does not requires these dedicated divsion and square root operations, but carries out these operations in an iterative fashion and uses the FPGA's DSP elements as hardware multipliers. The use of the changed method for calculating the statistic values without division and the use of hardware multipliers allows the architecture to run at a higher frequency due to the reduced latency. An increase of the operation frequency of more than 4 for calculating the running mean and an increase of 3 for calculating the running variance and an increase of more than 2 for calculation of the standard deviation can be achieved. By introducing a pipeline register, the architecture for calculating of the standard deviation can achieve the same f_{max}, as for calculating the variance at the cost of one clock cycle latency. The proposed architectures allow to quadruple or triple respectively the frame rate compared to the conventional approach.

Furthermore, the use of hardware multipliers embedded into Virtex 6 FPGA fabric for shifting operations was examined (similar to the proposal in [11]) and compared to realization of shifters with LUTs. The results in Table I indicate that LUTs are preferable for realizing shifters when a high f_{max} is the design goal.

V. CONCLUSION

The proposed method for calculating the mean, variance and standard deviation in an iterative manner can be realized efficiently as a hardware architecture. Contrary to carrying out combinational divisions, the proposed approach distributes the divisions over several iteration steps. This enables the calculation of running statistics using simple hardware elements. The gain of the proposed method for hardware architectures was shown by realizing it on an Field-Programmable Gate Array (FPGA), which tripled the frame rate that the architecture could process at a slightly decreased accuracy.

TABLE II
RESOURCE REQUIREMENTS OF FPGA HARDWARE ARCHITECTURE CALCULATING RUNNING STATISTICS WITH THE CONVENTIONAL APPROACH ON ON A XILINX VIRTEX 6 XC6VLX240T-2FF1759.

	Mean		+Variance		+Standard Deviation	
Image size	0.3MP	1MP	0.3MP	1MP	0.3MP	1MP
Registers	76	79	111	114	155	159
LUTs	1264	1358	2934	3036	3087	3193
DSP48E1s	0	0	2	2	2	2
f_{max} [MHz]	20.9	20.1	17.01	17.0	13.9	13.4
Framerate [FPS]	69	20	56	17	46	13

VI. ACKNOWLEDGEMENTS

Michael Klaiber would like to thank the German Academic Exchange Service (DAAD) for financial support. This work has been carried out within a research scholarship.

Michael Klaiber would like to thank the German Research Foundation (DFG) for the financial support. This work has been carried out within the research project Si 586 7/1 which belongs to the priority program DFG-SPP 1423 "Prozess-Spray".

REFERENCES

[1] D. Bailey, C. Johnston, and N. Ma, "Connected components analysis of streamed images," in *International Conference on Field Programmable Logic and Applications (FPL 2008)*, Sept. 2008, pp. 679–682.

[2] M. Klaiber, L. Rockstroh, Z. Wang, Y. Baroud, and S. Simon, "A memory-efficient parallel single pass architecture for connected component labeling of streamed images," in *2012 International Conference on Field-Programmable Technology (FPT)*, Dec. 2012, pp. 159–165.

[3] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[4] J. Bennett, R. Grout, P. Pebay, D. Roe, and D. Thompson, "Numerically stable, single-pass, parallel statistics algorithms," in *IEEE International Conference on Cluster Computing and Workshops (CLUSTER '09).*, Sept. 2009, pp. 1–8.

[5] T. F. Chan, G. H. Golub, and R. J. LeVeque, "Updating formulae and a pairwise algorithm for computing sample variances," Stanford, CA, USA, Tech. Rep., STAN-CS-79-773, Nov. 1979.

[6] E. Granger, S. Catudal, R. Grou, M. M. Mbaye, and Y. Savaria, "On current strategies for hardware acceleration of digital image restoration filters," *WSEAS Transaction on Electronics*, vol. 1, no. 3, pp. 551–557, 2004.

[7] J. Deschamps and G. Sutter, "Decimal division: Algorithms and FPGA implementations," in *2010 VI Southern Programmable Logic Conference (SPL)*, Mar. 2010, pp. 67–72.

[8] D. G. Bailey, "Space efficient division on FPGAs," in *13th Electronics New Zealand Conference*, Nov. 2006, pp. 206–211.

[9] Y. Li and W. Chu, "A new non-restoring square root algorithm and its vlsi implementations," in *IEEE International Conference on Computer Design: VLSI in Computers and Processors, (ICCD '96)*, Oct. 1996, pp. 538–544.

[10] "The usc-sipi image database," 2013. [Online]. Available: http://sipi.usc.edu/database/

[11] P. Gigliotti, "Implementing barrel shifters using multipliers," *Xilinx Application Note XAPP195*, 2004.