EURASIP Journal on Applied Signal Processing 2005:13, 2026–2042
© 2005 Viktor Fischer et al.

Cost-Effective Video Filtering Solution for Real-Time Vision Systems

Viktor Fischer
Laboratoire Traitement du Signal et Instrumentation, Unité Mixte de Recherche CNRS 5516, Université Jean Monnet, 10 rue Barrouin, 42000 Saint-Etienne, France
Email: fischer@univ-st-etienne.fr

Rastislav Lukac
Multimedia Laboratory, The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road Toronto, ON, Canada M5S 3G4
Email: lukacr@dsp.utoronto.ca

Karl Martin
Multimedia Laboratory, The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road Toronto, ON, Canada M5S 3G4
Email: kmartin@dsp.utoronto.ca

Received 23 December 2003; Revised 12 November 2004

This paper presents an efficient video filtering scheme and its implementation in a field-programmable logic device (FPLD). Since the proposed nonlinear, spatiotemporal filtering scheme is based on order statistics, its efficient implementation benefits from a bit-serial realization. The utilization of both the spatial and temporal correlation characteristics of the processed video significantly increases the computational demands on this solution, and thus, implementation becomes a significant challenge. Simulation studies reported in this paper indicate that the proposed pipelined bit-serial FPLD filtering solution can achieve speeds of up to 97.6 Mpixels/s and consumes 1700 to 2700 logic cells for the speed-optimized and area-optimized versions, respectively. Thus, the filter area represents only 6.6 to 10.5% of the Altera STRATIX EP1S25 device available on the Altera Stratix DSP evaluation board, which has been used to implement a prototype of the entire real-time vision system. As such, the proposed adaptive video filtering scheme is both practical and attractive for real-time machine vision and surveillance systems as well as conventional video and multimedia applications.

Keywords and phrases: VHDL implementation, FPLD, bit-serial approach, pipelined solution, video filtering and enhancement, nonlinear adaptive filter design.

1. INTRODUCTION

Computer vision methods are becoming increasingly important for the development of novel commercial devices such as wireless phones, vision-based pocket devices, sensor networks, and surveillance and automotive apparatus [1, 2, 3, 4]. This increases the demand for hardware-based implementations of new, relatively complex video processing algorithms [5]. It is not difficult to see that incorporating recent advances in the fields of computer and machine vision, hardware, software, digital signal/image processing, graphics, and telecommunications into integrated intelligent computer vision systems extends the possibilities of the traditional vision-based communication between end-users or between machine and human. At the same time, it increases the proliferation of the vision systems.

Due to imperfections in image sensors, digital images are often contaminated with noise [6, 7]. Although many sources of noise, including fixed pattern noise and on-chip/off-chip amplifier noise, can be significantly reduced [8], images are still affected by the corruption caused by photon shot noise and dark-current shot noise resulting from the photoelectric process with Poisson statistics [8]. Due to the complex nature of the noise process, the overall acquisition noise is usually modeled as a zero mean white Gaussian noise [9, 10]. Aside from this type of noise, image imperfections resulting from impulsive noise are generated during transmission through a communication channel [11, 12, 13], with sources ranging from human-made sources (switching and interference) to signal representation (bit errors) and natural (atmospheric lightning) ones. The resulting noisy samples, so-called outliers, differ significantly in magnitude from the noise-free signal elements. It has been widely observed that outliers, as well as noise in general, affect the perceptual quality of the image, decreasing not only the appreciation of the image but also the performance of the task for which the image was intended [14]. Therefore, image filtering is of paramount importance [6, 7, 14].

Nonlinear filters have replaced linear filters in many image processing applications [15, 16, 17, 18, 19], since (i) they can operate effectively in various noisy conditions and potentially preserve the structural information of the image, (ii) image signals are nonlinear in nature, and (iii) images are perceived via the human visual system which has strong nonlinear characteristics [20]. Among numerous nonlinear filters, the most popular filtering schemes are based on robust order statistics [6] due to excellent robust properties and simplicity of design. The order-statistics-based filters utilize algebraic ordering of a windowed set of data to compute the output signal [7]. They constitute a rich class of nonlinear filtering operators whose comprehensive overview can be found in [15, 16]. The most interesting examples include the well-known median filter [21], rank-order filters [22], combination filters [23, 24], multistage filters [25], weighted median filters [26, 27], weighted order-statistic filters [28, 29], lower-upper-middle (LUM) filters [30, 31], morphological filters [6, 32], and stack filters [33, 34, 35, 36].

In the last decade, technological advances in hardware and software have allowed the extension of nonlinear filtering algorithms to multidimensional image processing. Therefore it is not a surprise that nonlinear filters are successfully used today in color image [7, 14, 37, 38, 39, 40] and/or video processing [9, 41, 42, 43, 44, 45] applications. Since many applications require the processing of signals under real-time constraints, significant research efforts have been dedicated to the hardware implementation and embedding of nonlinear filtering techniques in various circuits [46, 47, 48, 49, 50].

This paper introduces the field-programmable logic device (FPLD) implementation of the simplified variant [51, 52] of the nonlinear adaptive video filtering (NAVF) technique [53]. Utilizing an adaptive design based on order-statistics and incorporating both temporal correlation existing among the frames and spatial correlation of the samples within the frames, the three-dimensional (3D), so-called spatiotemporal scheme under consideration is capable of tracking video nonlinearities. Due to the extra degrees of freedom achieved through the use of three different smoothing levels (as opposed to the conventional fixed smoothing operator), the proposed scheme is capable of tracking varying image and noise statistics. At the same time, the scheme produces an excellent tradeoff between noise attenuation and signal-detail preservation resulting in reconstructed video with impressive visual quality. Based on the simplified structure of [51, 52], the NAVF complexity has been reduced, without significant loss in performance, to a level useful for practical implementation in FPLDs. The use of the FPLD-based reconfigurable technology makes the proposed filtering system sufficiently adaptable and flexible for different types of camera and video processing applications; modification of the architecture may be required due to changes in the nature/representation of the signal, varying image and noise statistics, and various end-user needs.

The rest of this paper is organized as follows. The video filtering scheme under consideration is briefly described and analyzed in Section 2. Performance comparisons with relevant filtering schemes are provided in terms of commonly used image quality measures. An overview of the prior art in filter implementation is provided. The proposed implementation approach is outlined in Section 3, with motivation and design characteristics discussed in detail. In Section 4, experimental results corresponding to the implementation of the proposed method using the selected Altera FPLD are analyzed in terms of maximum usable clock frequency, hardware resources, and power consumption for both the stand-alone filter and complete 3D filtering solution. Finally, this paper concludes in Section 5.

2. MOTIVATION AND BACKGROUND

Consider a $K_1 \times K_2 \times K_3$ image sequence $x : \mathbb{Z}^3 \rightarrow \mathbb{Z}$ representing a 3D image signal or a time sequence of $K_1 \times K_2$ two-dimensional (2D) images with samples $x(p, q, t) \in \mathbb{Z}$, where K_3 denotes the number of image frames. Each image pixel $x(p, q, t)$, for $p = 1, 2, \ldots, K_1$, $q = 1, 2, \ldots, K_2$, and $t = 1, 2, \ldots, K_3$, is a function of the spatial coordinates (p, q) and time t.

The most common approach to the problem of noise reduction is the utilization of some kind of smoothing operation which filters out random fluctuations due to noise. This approach is based on a sliding window $W(p, q, t) = \{x(p-m, q-n, t-o) \in \mathbb{Z};\ m, n, o \in S\}$ centered in $x(p, q, t)$, with S denoting the 3D support of the window [9] of finite size N. Assuming, for simplicity, that the index i, for $i = 1, 2, \ldots, N$, denotes the position of the samples within the filtering window (Figure 1), the data population of $W(p, q, t)$ can be equivalently rewritten to $W = \{x_1, x_2, \ldots, x_N\}$. The filtering procedure replaces the center $x^* = x(p, q, t)$ of the window by some function $f(\cdot)$ of the local neighborhood samples $\{x_1, x_2, \ldots, x_N\}$. Thus, the value of the estimated sample $y(p, q, t) = y = f(W)$ depends on the values of the image samples in its neighborhood W. The window operator slides over the image to affect individually all the image pixels. This is based on the assumption that the processes generating the image are stationary within the window and the probability of a particular behavior does not depend on the image coordinates.

2.1. *Algorithm description*

Following the robust theory of order statistics [6], most popular filtering algorithms developed to suppress impulsive noise in images operate on the ordered values within the observation window. Using sample ordering, both correlation

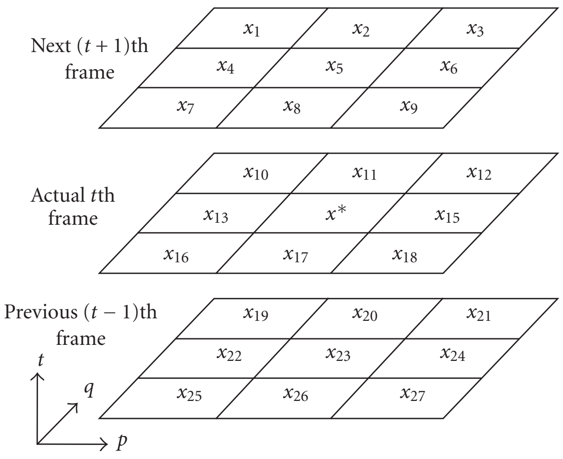

FIGURE 1: A $3 \times 3 \times 3$ (cubic) spatiotemporal filtering window centered in $x^* = x_{(N+1)/2}$.

and time information is ignored and the estimate is purely constituted based on magnitude information [9]. The fact that the noisy samples usually correspond to the extreme ranks in the ordered sequence makes the samples occupying the middle ranks favorable to complete the filtering task.

Let $W = \{x_1, x_2, \ldots, x_N\}$ be the set of the input samples within the observation window (Figure 1). Based on magnitude information, the ordering of $x_1, x_2, \ldots, x_N$ results in the ordered set commonly defined as follows [6, 15, 17, 19]:

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)}, \tag{1}$$

where $x_{(i)} \in W$, for $i = 1, 2, \ldots, N$, represents the ith order statistic.

Using the smoothing parameter $k = 1, 2, \ldots, (N+1)/2$, the comparison of the lower and upper order statistics $x_{(k)}$ and $x_{(N-k+1)}$ of (1), respectively, along with the middle sample $x^* = x_{(N+1)/2}$ in W forms a lower-upper-middle (LUM) smoothing function [30, 31], defined as follows:

$$y_k = \operatorname{med}\{x_{(k)}, x^*, x_{(N-k+1)}\}, \tag{2}$$

where y_k denotes the LUM smoother output and med is a median operator.

If x^* lies in the range formed by $x_{(k)}$ and $x_{(N-k+1)}$, it is not modified. If x^* lies outside this range, it is replaced with one of the two extremes that lies closer to the sample median ($x_{(N+1)/2}$). By varying the filter parameter k, the amount of smoothing done by the LUM smoother can range from no smoothing equivalent to the identity operation ($k = 1$) to the maximum amount of smoothing provided by the median ($k = (N+1)/2$). It is evident that the LUM smoothing capability increases with k. However, with large k, the smoothing operation often results in image blurring [30]. Therefore, depending on varying image and noise statistics, the adaptive choice of k is of paramount importance.

In order to track the changes in local image statistics and provide the best balance between the smoothing and detail-preserving LUM characteristics, the NAVF scheme has been introduced [53]. Its adaptive behavior is achieved by the comparison of the absolute differences $|x^* - y_k|$ with associated thresholds $\xi_k \geq 0$, for $k = 1, 2, \ldots, (N+1)/2$. Since $k = 1$ denotes the identity filter $y_1 = x^*$ whose filtering operation does not affect the input, and the smoothing capability of the LUM smoother increases with k, it is reasonable to say that $|x^* - y_k|$ increases in magnitude as follows:

$$|y_1 - x^*| \leq |y_2 - x^*| \leq \cdots \leq |y_{(N+1)/2} - x^*|, \tag{3}$$

where $|y_1 - x^*| = 0$. Following the terms of (3), the associated thresholds should be set according to

$$\xi_1 \leq \xi_2 \leq \cdots \leq \xi_{(N+1)/2} \tag{4}$$

with $\xi_1 = 0$. The zero value of ξ_1 corresponds to the use of the identity operator y_1 which keeps the central sample x^* unchanged.

Based on (3) and (4), the NAVF output y is equivalent to y_η, with $\eta = \sum_{k=1}^{(N+1)/2} \lambda_k$ defined via the parameters

$$\lambda_k = \begin{cases} 1 & \text{if } |y_k - x^*| \geq \xi_k, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $\xi_1, \xi_2, \ldots, \xi_{(N+1)/2}$ are the thresholds used to control the accuracy of the NAVF estimates. In the case of the $3 \times 3 \times 3$ spatiotemporal filter window ($N = 27$) shown in Figure 1, the NAVF scheme requires the calculation of $(N+1)/2 = 14$ different y_k's. For this spatiotemporal processing commonly used in video filtering applications and a conventional 8-bit-per-pixel image representation, the recommended setting of $\xi_1, \xi_2, \ldots, \xi_{(N+1)/2}$ found through a genetic algorithm optimization is defined as follows [53]:

$$\begin{aligned} &\{\xi_1, \xi_2, \ldots, \xi_{14}\} = \\ &\quad \{0, 4, 5, 7, 9, 12, 15, 16, 22, 23, 38, 43, 48, 52\}. \end{aligned} \tag{6}$$

These values are sufficiently robust for a wide range of test videos with various image statistics and/or motion complexity [53]. During optimization of the filter parameters $\xi_1, \xi_2, \ldots, \xi_{14}$, it has been observed that larger threshold values spoil the noise attenuation characteristics of the NAVF scheme and result in unremoved outliers in the filter output. While smaller thresholds improve this situation, the detail-preserving characteristics are, in turn, negatively impacted.

It is clear that the NAVF scheme requiring the determination of 14 different y_k's is computationally demanding and cannot be used as a cost-effective solution for real-time video and multimedia applications. Therefore, the NAVF structure controlled by (6) has been reduced as follows [51]:

$$\{\xi_1, \xi_7, \xi_{14}\} = \{0, 15, 52\}, \tag{7}$$

where ξ_1, ξ_7, ξ_{14} are associated with the identity operation y_1,

```
Inputs:   $K_3$ noisy frames of $K_1 \times K_2$ image pixels
          Moving window spawning the input set $\{x_1, x_2, \ldots, x_n\}$
          Thresholds $\xi_7 = 15$, $\xi_{14} = 52$
Output:   $K_3$ frames of $K_1 \times K_3$ image pixels
For $t = 1$ to $K_3$
  For $p = 1$ to $K_1$
    For $q = 1$ to $K_2$
      Determine the input set $W(p, q, t) = \{x_1, x_2, \ldots, x_N\}$, for $N = 27$
      Let $x^* = x_{(N+1)/2}$ denote the central sample
      Sort $\{x_1, x_2, \ldots, x_N\}$ to the ordered set $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)}$
      Let filter output $y = x^*$
      Let the LUM smoother $y_7 = \mathrm{med}\{x_{(7)}, x^*, x_{(N-7+1)}\}$
      Let the LUM smoother $y_{14} = \mathrm{med}\{x_{(14)}, x^*, x_{(N-14+1)}\}$
      If $|y_7 - x^*| \geq \xi_7$ OR $|y_{14} - x^*| \geq \xi_{14}$
        Update filter output to $y = y_7$
      End
      If $|y_7 - x^*| \geq \xi_7$ AND $|y_{14} - x^*| \geq \xi_{14}$
        Update filter output to $y = y_{14}$
      End
    End
  End
End
```

Algorithm 1: Algorithm of the reduced NAVF scheme.

the balanced LUM smoothing y_7, and the median (the maximum smoothing) operation y_{14}, respectively. The robustness of the setting defined in (7) has been verified, through the use of the linear and evolutionary optimization tools, in [52]. The results indicate that the filter is sufficiently robust to relatively large deviations from the assumed during-training conditions. In the case of substantial qualitative difference in terms of noise characteristics, reoptimization of the filter parameters may be recommended. It should be emphasized that both the full (original) and reduced NAVF solutions are primarily geared to address the problem of impulsive noise removal. For such a task, the proposed solution results in excellent performance.

The algorithmic steps performed by the reduced NAVF scheme are summarized, in pseudocode format, in Algorithm 1. The scheme requires in each processing location (p, q, t): (i) to determine the window center x^* and the input set $W = \{x_1, x_2, \ldots, x_N\}$, (ii) to order the input samples according to their magnitude, (iii) to determine the outputs of the two LUM smoothers y_7 and y_{14}, (iv) to compare the absolute differences $|y_7 - x^*|$ and $|y_{14} - x^*|$ with the corresponding thresholds ξ_7 and ξ_{14}, respectively, and (v) based on these comparisons, to set one of x^*, y_7, and y_{14} as the filter output y.

2.2. *Filtering efficiency*

Experimentation with a number of test videos corrupted by impulsive noise [15] showed that the reduced NAVF scheme of (7) is sufficiently robust and operates without significant loss in performance [52]. This is demonstrated here using test image sequences consisting of 30 frames with an 8-bit-per-sample representation and 256×256 spatial resolution. For better illustration, Figures 2a, 2b, and 2c depict the 5th frame of the test videos. The example of the noisy frame is shown in Figure 2d. This image corresponds to a video frame contaminated by 10% random-valued impulsive noise [15, 53], with the rate denoting the amount of the corrupted pixels and the noise magnitude independent from pixel to pixel.

The method is applied to the test videos degraded by 5% and 10% noise and performance is measured via the mean absolute error (MAE) and mean square error (MSE) measures commonly used in the image processing community. Using these error criteria, the reduced NAVF scheme [51, 52] is compared to other video filtering techniques, including the previously mentioned full NAVF scheme [53], median filter (MF) [6], standard LUM smoothers [30], and multistage median filters (MMFs) [41] as well as some spatiotemporal switching median filters with the switching control based on the averaging operations defined over the middle-ranked samples (ICM) [54], local contrast probability (LCP) [55], center-weighted median switching filter (CWMSF) [56], variance of the input set (VSMF) [57], and advanced multilevel processing (ASM) [58].

Tables 1 and 2 summarize the objective, numerical results corresponding to the test videos shown in Figures 2a, 2b, and 2c. The results indicate that the full NAVF scheme achieves the best performance in terms of MAE and MSE among all the tested filtering schemes. Moreover, it can be seen that the reduced NAVF scheme also produces excellent results, although its filtering structure has been simplified from 14 to 3 smoothing levels y_k compared to the full NAVF scheme. Therefore it can be concluded that the reduced NAVF scheme is useful for video filtering purposes and, because of its simplicity, it is suitable also for cost-effective applications.

Figures 2c, 2d, 2e, and 2f show the visual comparison of the original frame, the contaminated frame, and the filtered outputs produced using the MF technique and the reduced

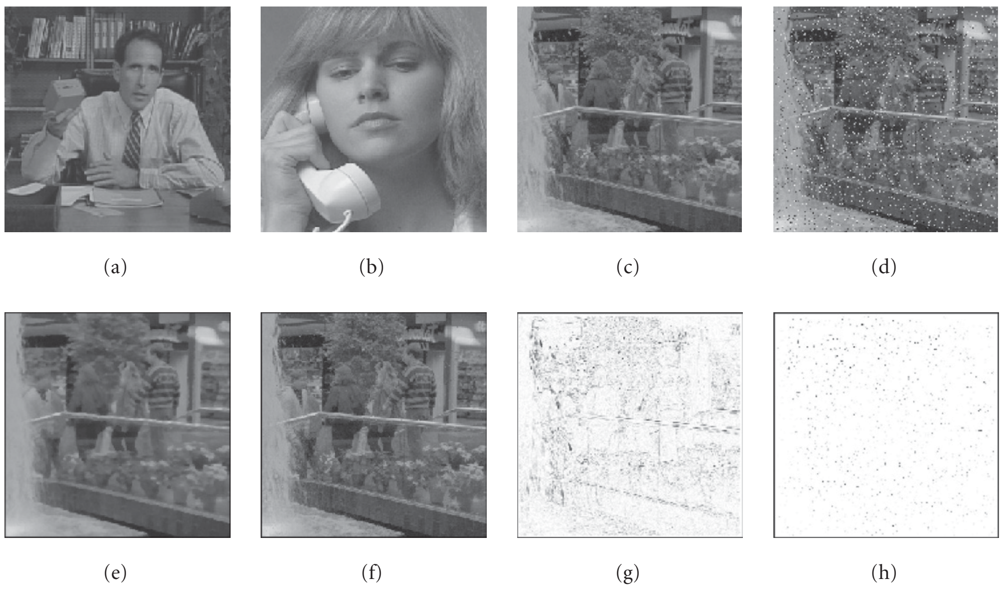

Figure 2: The 5th frame of the test videos: (a) Salesman, (b) Woman, (c) People, (d) image contaminated with 10% impulsive noise, (e) MF output, (f) reduced NAVF output, (g) MF estimation error emphasized by a factor of 3, and (h) reduced NAVF estimation error emphasized by a factor of 3.

Table 1: Comparison of the presented algorithms using test videos corrupted with 5% impulsive noise.

Test video	Salesman		Woman		People	
Method	MAE	MSE	MAE	MSE	MAE	MSE
Noisy	3.709	419.2	3.461	353.3	3.645	399.7
MF	4.107	57.3	3.190	27.7	6.852	117.9
LUM $k = 6$	0.970	18.0	0.757	8.5	1.624	29.6
LUM $k = 10$	2.070	28.8	1.530	11.9	3.452	52.4
MMF	1.664	19.9	1.720	16.0	3.720	58.3
ICM	0.706	22.8	0.487	11.9	0.792	23.2
LCP	1.880	39.7	1.327	17.5	2.494	47.6
CWMSF	0.545	13.9	0.412	7.6	0.783	21.3
VSMF	1.288	31.5	0.846	12.9	1.672	36.7
ASM	0.459	11.9	0.308	5.4	0.717	19.7
Full NAVF	0.386	8.2	0.265	3.8	0.863	22.9
Reduced NAVF	0.436	9.7	0.284	4.4	0.859	24.1

NAVF scheme. Figure 2e illustrates that the MF scheme blurs image edges, structural content, and fine details. On the other hand, the reduced NAVF scheme preserves the image details and removes outliers (Figure 2f). Due to this impressive performance, the reduced NAVF produces a denoised image similar to the original depicted in Figure 2c. Figures 2g and 2h show the estimation errors of the standard MF scheme compared to the reduced NAVF scheme. It can be seen that the MF approach is characterized by large estimation error, which corresponds to edge blurring and destruction of fine details (Figure 2g). The reduced NAVF scheme tends to avoid the blurring of structural content and excellently preserves the desired signal features. This results in very small estimation error, as depicted in Figure 2h.

2.3. *Filter implementation: prior art*

Apart from the numerical behavior (actual performance) of any proposed algorithm, its computational complexity is a realistic measure of its practicality and usefulness. The (full and reduced) NAVF filtering schemes belong to the class of filters based on order statistics. To determine the output based on their rank within a group of inputs, various techniques have been proposed for implementing these kinds of filters [46, 47, 48, 49, 50].

Based on the amount of information processed concurrently, implementation approaches can be classified into two main groups [47]: word-based and bit-based techniques. Word-based architectures (or bit-parallel architectures) process the bits of the input samples in parallel, but the samples

Table 2: Comparison of the presented algorithms using test videos corrupted with 10% impulsive noise.

Test video	Salesman		Woman		People	
Method	MAE	MSE	MAE	MSE	MAE	MSE
Noisy	7.287	825.1	6.738	688.4	7.069	772.8
MF	4.237	59.6	3.265	29.2	6.991	121.1
LUM $k = 6$	1.527	38.5	1.117	18.4	2.254	54.2
LUM $k = 10$	2.288	34.8	1.674	14.9	3.706	59.9
MMF	2.286	50.6	2.142	33.3	4.420	92.1
ICM	1.211	35.4	0.938	22.7	1.356	37.6
LCP	1.833	42.8	1.235	19.2	2.444	51.0
CWMSF	0.923	24.3	0.761	12.8	1.234	33.9
VSMF	1.397	38.5	0.927	18.3	1.720	42.8
ASM	0.860	21.6	0.609	10.9	1.165	30.2
Full NAVF	0.736	16.2	0.523	8.1	1.283	32.6
Reduced NAVF	0.776	17.1	0.553	8.8	1.299	34.1

are usually processed sequentially. On the contrary, bit-based filters (or bit-serial architectures) process input samples bitwise, but the samples included in the window are processed in parallel. In contrast to bit-parallel algorithms, the bit-serial algorithms often enable the creation of efficient pipelined structures.

Another kind of classification recognizes nonrecursive and recursive algorithms [50]. Since recursive algorithms use the same piece of hardware in an iterative manner, they are usually more area-efficient, but slower. Because of the existing loop, the pipelining cannot be applied. On the other hand, nonrecursive algorithms enable speeding up the filtering process via pipelining techniques and block processing [59].

The architectures of the rank-order-based filters can be divided into three main categories [50]: array architectures, sorting network architectures, and stack filter-based architectures. In array architectures [50], each element in the window is associated with a rank, and with each window shift, the ranks of the elements are updated. The array architectures with window size N consist of a semisystolic linear array of N processors. These architectures are suited for both bit-parallel and bit-serial implementations. Furthermore, they can be easily pipelined, thereby supporting high throughput applications. However, this kind of architecture is not suitable for large processing windows such as those used in spatiotemporal (3D) video filters considered in this paper.

The sorting network architectures implement the rank-order filter by first sorting the samples and then selecting the sample of corresponding rank [46, 50, 59, 60, 61, 62]. The filtering can be faster, if the sorting of samples from the previous position of the sliding window is maintained and only the incoming sample is positioned to the correct rank. Sorting network architectures with presorted values can be relatively efficient for one dimensional (1D) filters, where only one sample has to be classified in each sample period. However, these architectures are not suitable for 3D filters, since multiple samples (in our case, 9 samples for a cubic $3 \times 3 \times 3$ window) arrive at each new sample period.

Probably the most efficient implementation approach related to the use of rank-order-based filters for image processing applications is based on stack filters. A stack filter translates the filtering operation to the binary domain through the use of threshold decomposition [49, 63]. The bit-parallel realization of the stack filter decomposes the input sample to $2^B - 1$ bit levels [49, 50], where B is the sample wordlength. Each level is processed separately. It is clear that if B is high, the bit-parallel architecture of the stack filter is not suitable for a cost effective application, since the number of processing levels depends on B exponentially. In the bit-serial version of the stack filter [50, 64, 65, 66], the input samples are processed bitwise in only B bit levels using (i) a majority function [67, 68, 69], (ii) a positive Boolean function (PBF) [64, 70], or (iii) a polarizing function [71, 72]. Since the area of the bit-serial stack filters depends linearly on the number of bit levels, these stack filters usually permit the most area-efficient implementations [50, 65].

While there are several implementations of rank-order-based filters in field-programmable gate arrays (FPGAs) published in the literature [62, 73, 74], we did not find any FPGA implementation of 3D rank-order-based filters suitable for video processing applications. In fact, due to the significant growth in time delays and hardware requirements in spatiotemporal video filtering such as the considered reduced NAVF scheme, very few algorithms are suitable for hardware implementation. Based on the aforementioned facts, we have selected the nonrecursive, bit-serial stack architecture based on a PBF function as the best candidate. Besides its area-efficient implementation, it enables the use of a pipelined structure and thus an increase in the speed of the filtering process. The proposed hardware structure is presented in the next section.

3. PROPOSED HARDWARE IMPLEMENTATION

The FPLD target technology is selected in this paper to improve adaptability and flexibility of the filtering system for different types of cameras and video processing applications.

```
Inputs:  Set of noisy samples (input set) ${x_1, x_2, ..., x_N}$
         Bit word length $B$
Output: Image sample $y$
For $i = 1$ to $N$ do in parallel
  Let $g_i^{B-1} = 1$
End
For $j = B - 1$ to 0 do in serial
  For $i = 1$ to $N$ do in parallel
    If $g_i^{j+1} = 0$
      Let $x_i^j = x_i^{j+1}$
    Let $y^j = f_{k,N}(x_1^j, x_2^j, ..., x_N^j)$
    If $g_i^{j+1} = 1$ and $y^j \oplus x_i^j = 1$
      Let $g_i^j = 0$
    Else
      Let $g_i^j = g_i^{j+1}$
  End
End
Let output $y = y^{B-1}2^{B-1} + y^{B-2}2^{B-2} + \cdots + y^1 2 + y^0$
```

Algorithm 2: Algorithm of the pipelined bit-serial realization of the PBF-based LUM smoother.

The scalability of the filter together with the reconfigurable technology used for filter implementation should enable easy modification of the proposed architecture for video signals differing in parameters such as sample frequency and frame size, as well as the number of bits used for sample representation. Since not all filter implementations are directly exploitable in FPLD technology, the utilization of FPLD devices in video signal filtering sometimes needs a special approach. Our goal is to (i) propose a cost-effective and flexible solution using FPLD devices, and (ii) ensure its suitability for real-time spatiotemporal (3D) video filtering.

3.1. Bit-serial structure implementation

The bit-serial approach of [64] reduces the filtering procedure to binary calculations and simplifies the smoothing operators to become PBFs. In this way, the designer avoids implementing time-consuming ordering operations, which make the filtering algorithm significantly slower and difficult for realization especially for large window shapes such as the employed 3×3×3 spatiotemporal window shown in Figure 1.

Algorithm 2 summarizes the steps performed using the bit-serial realization of the PBF-based LUM smoother. Note that the LUM smoother is required to process the set of N image samples coded with B bits per sample. Each input sample $x_i \in W = \{x_1, x_2, \ldots, x_N\}$ is expressed in binary form as $x_i = (x_i^{B-1}, x_i^{B-2}, \ldots, x_i^0)$, where x_i^j, for $j = B-1, B-2, \ldots, 0$, represents jth bit of x_i. The algorithm produces the output sample $y = (y_i^{B-1}, y_i^{B-2}, \ldots, y_i^0)$. Note that propagation (binary) flags $g_i^{B-1}, g_i^{B-2}, \ldots, g_i^0$ are associated with each input sample x_i. When the flag g_i^{j+1} is high ($g_i^{j+1} = 1$), the binary inputs $x_i^j, x_i^{j-1}, \ldots, x_i^0$ remain unchanged. When $x_i^j \neq y^j$, all the bits $x_i^{j-1}, x_i^{j-2}, \ldots, x_i^0$ change to x_i^j for the remaining algorithmic steps. This is also indicated by flags $g_i^{j-1}, g_i^{j-2}, \ldots, g_i^0$, which are updated to low ($g_i^{j-1} = g_i^{j-2} = \cdots = g_i^0 = 0$) for the remaining steps.

The most important part of the procedure summarized in Algorithm 2 corresponds to the LUM-PBF expression in [75]. This simplifies the LUM smoother defined by the smoothing parameter k and the window size N into the PBF defined as follows:

$$y^j = f_{k,N}(\cdot) = \begin{cases} 1 & \text{if } x^{*j} = 1, \sum W^{*j} \geq k-1, \\ 1 & \text{if } \sum W^{*j} \geq N-k+1, \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

It has been proven [75] that the output bit $y^j = f_{k,N}(x_1^j, x_2^j, \ldots, x_N^j)$ of the LUM smoother can be simply determined using the jth bit of the central input sample x^* and 1's in the set W^{*j} of neighboring binary samples associated with the jth bit. This results in the fast, binary LUM smoothing defined in (8).

For illustrative purposes, Table 3 summarizes the computational steps of the bit-serial LUM-PBF approach. We consider the window size $N = 9$, the smoothing parameter $k = 4$, the word length $B = 8$, and the input set $W = \{140, 135, 31, 152, 145, 141, 138, 141, 142\}$ with the central sample $x^* = 145$. The procedure starts by evaluating the most significant bit ($j = 7$) of the output. At this bit level, $x^{*7} = 1$ and the number of 1's in W^{*7} is equal to 7. Thus, applying (8), $y^7 = 1$. This binary output is compared with the most significant bits (MSBs) of the other window elements. Sample x_3, whose MSB is different from the filter output y^7, propagates its actual bit value x_3^7 to all less significant bits in all the following steps. This replaces the original, lower bits with the jth bit resulting in $x_3^6 = x_3^7, x_3^5 = x_3^7, \ldots, x_3^0 = x_3^7$. The procedure continues by evaluating x^{*6}, W^{*6}, and (8) down to the least significant bit related to $j = 0$. The output bits $y^{B-1}, y^{B-2}, \ldots, y^0$ constitute the output pixel $y = x_6 = x_8 = 141$. Applying the LUM smoother, defined by $k = 3$, to the identical input set W, the output sample is changed to $y = x_9 = 142$ as shown in Table 4.

The hardware implementation of the conventional LUM smoothing algorithm consists of two basic types of blocks [76]: (i) $N \times B$ LUM propagation cells, and (ii) B combinatorial blocks implementing the LUM-PBF defined by (8).

Figure 3 presents the propagation cell structure. The cell output x_i^j, which is obtained from the output of the multiplexer, is the same as the output of the flip-flop, if the propagation flag from the upper level $g_i^{j+1} = 1$; otherwise the cell output is replaced with x_i^{j+1} propagated from upper levels. This new bit value is then passed to the $f_{k,N}(\cdot)$ PBF function and to the lower-level cells. To determine a new value of the propagation flag g_i^j, the binary output $y^j = f_{k,N}(\cdot)$ of the PBF function is compared with the data bit x_i^j using a XNOR gate. If $y^j \neq x_i^j$, the propagation flags (AND gate outputs) for all lower levels will be 0.

Table 3: LUM-PBF smoothing ($N = 9$ and $k = 4$) realized via the bit-serial approach.

Sample	x_1	x_2	x_3	x_4	x^*	x_6	x_7	x_8	x_9	—	y
Value	140	135	31	152	145	141	138	141	142	—	141
Bit j	x_1^j	x_2^j	x_3^j	x_4^j	x^{*j}	x_6^j	x_7^j	x_8^j	x_9^j	W^{*j}	y^j
7	1	1	0	1	1	1	1	1	1	7	1
6	0	0	0(0)	0	0	0	0	0	0	0	0
5	0	0	0(0)	0	0	0	0	0	0	0	0
4	0	0	1(0)	1	1	0	0	0	0	1	0
3	1	0	1(0)	1(1)	0(1)	1	1	1	1	6	1
2	1	1(0)	1(0)	0(1)	0(1)	1	0	1	1	5	1
1	0	1(0)	1(0)	0(1)	0(1)	0	1(0)	0	1	2	0
0	0	1(0)	1(0)	0(1)	1(1)	1(0)	0(0)	1	0(1)	3	1

Table 4: LUM-PBF smoothing ($N = 9$ and $k = 3$) realized via the bit-serial approach.

Sample	x_1	x_2	x_3	x_4	x^*	x_6	x_7	x_8	x_9	—	y
Value	140	135	31	152	145	141	138	141	142	—	142
Bit j	x_1^j	x_2^j	x_3^j	x_4^j	x^{*j}	x_6^j	x_7^j	x_8^j	x_9^j	W^{*j}	y^j
7	1	1	0	1	1	1	1	1	1	7	1
6	0	0	0(0)	0	0	0	0	0	0	0	0
5	0	0	0(0)	0	0	0	0	0	0	0	0
4	0	0	1(0)	1	1	0	0	0	0	1	0
3	1	0	1(0)	1(1)	0(1)	1	1	1	1	6	1
2	1	1(0)	1(0)	0(1)	0(1)	1	0	1	1	5	1
1	0	1(0)	1(0)	0(1)	0(1)	0	1(0)	0	1	2	1
0	0(0)	1(0)	1(0)	0(1)	1(1)	1(0)	0(0)	1(0)	0	1	0

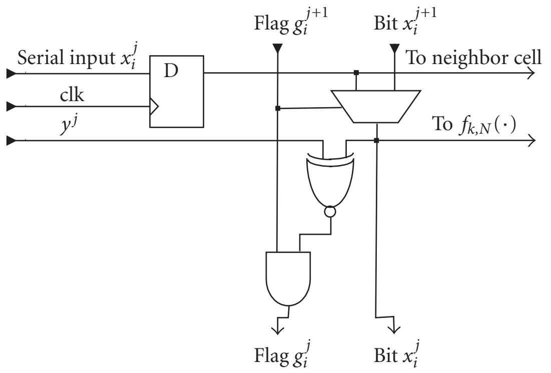

Figure 3: The ith propagation cell at the jth bit plane of the LUM smoother.

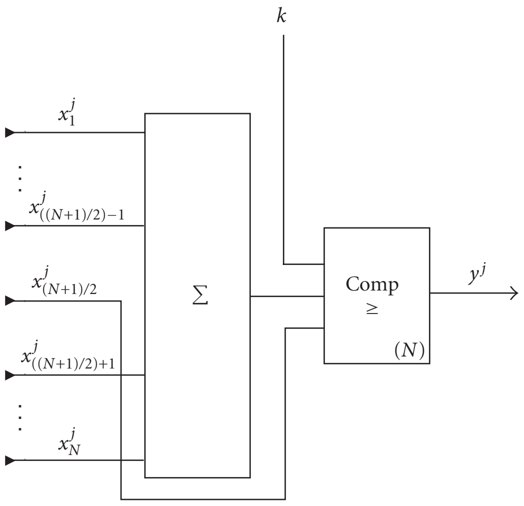

Figure 4: Internal structure of the N-input PBF function $f_{k,N}(\cdot)$.

The combinatorial block representing the PBF implementation according to (8) is composed of the adder and the comparator (see Figure 4). The adder counts the number of 1's (high bits) present at 26 binary inputs (included in W^{*j}) of the block. The comparator produces the output bit of one binary LUM smoother by comparing the result of the addition with the value $k-1$ for $x^{*j} = 1$ or $N-k+1$ for $x^{*j} = 0$ (8).

Figure 5 shows one-bit level of the proposed 3D adaptive filter with $N = 27$. Nine groups of 3 propagation cells—because of the space limitation, only two extremes and one central group are depicted—are connected to 9 horizontal

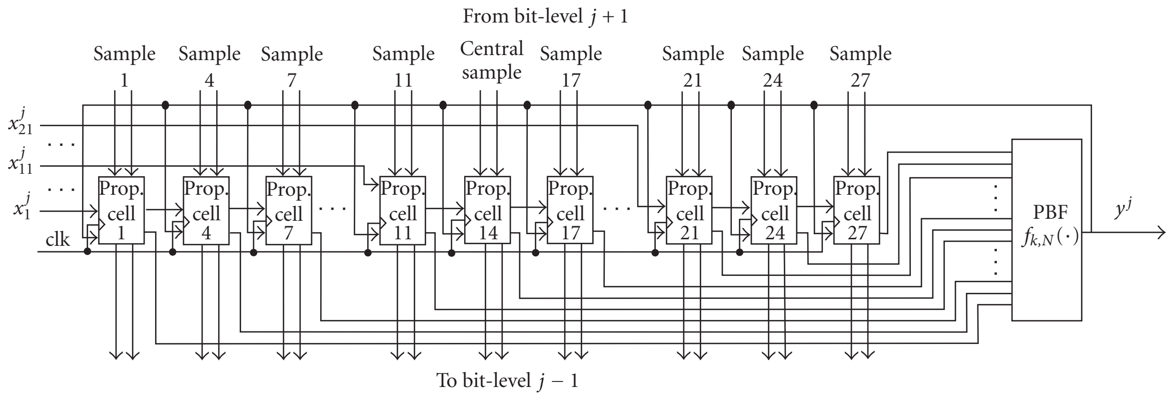

FIGURE 5: The jth bit plane of the 3D LUM smoother with $N = 27$.

data inputs (samples x_1, x_2, x_3, x_{10}, x_{11}, x_{12}, x_{19}, x_{20}, and x_{21} from Figure 1). Vertical propagation flags and propagated data bits coming from upper levels are updated and transmitted to lower levels. The PBF has 26 equivalent inputs and one special input for the central sample. The output of the PBF represents the bit-level filter output.

3.2. Parallel and pipelined filter structure

In the parallel version of the LUM filter, all the bit levels of the input set are processed in parallel. However, one of the main advantages of the bit-serial structure is that the bit levels can be processed independently and thus faster. This feature can be successfully used in the pipelined version of the filter, in which each bit level of the filter processes corresponding bit of one of B subsequent samples. The differences between these two implementations of the filter will now be discussed.

The complete LUM filter with parallel structure contains B identical levels. The critical data path (the longest data path between any two registers determining the maximum clock frequency of the filter) starts at the highest bit-level cells and passes horizontally through the PBF at the same level, coming back to the input of the XNOR gate of the bit cell. It continues vertically to the next lower level and so on until the lowest level of the filter. Thus, the parallel version of the filter has B propagation cells and B PBFs in the data path.

Figure 6 shows the pipelined version of the LUM filter propagation cell. Comparing Figures 3 and 6, it can be observed that the standard structure shown in Figure 3 has been extended with two pipeline registers. Due to the bit-serial nature of the algorithm, the bit-level pipelining allows concurrent processing of B subsequent bits corresponding to different bit levels using a set of B identical bit-slices from Figure 5. Since the bits of the input sample are not processed concurrently, they have to be delayed in input/output delay lines composed of triangles of shift registers (Figure 7). The critical data path of the pipelined version includes only one propagation cell and one PBF at the same level. It is therefore up to B times faster than the standard parallel version. However, the pipelined version is larger, because each propagation cell has two additional registers and $B + 1$ clock periods latency.

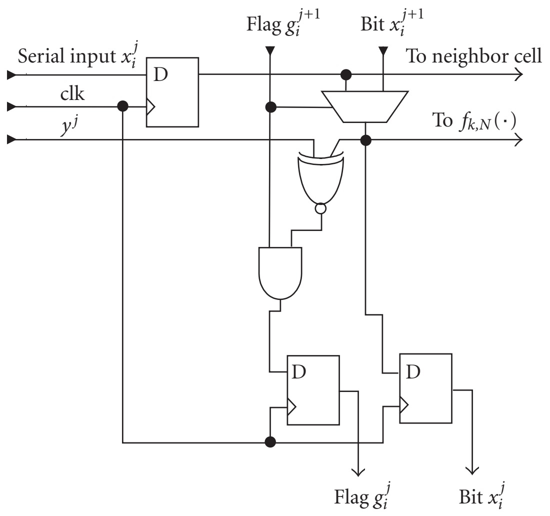

FIGURE 6: Pipelined version of the LUM smoother propagation cell.

Because of its higher speed, the pipelined scheme is much more appropriate for the real-time video applications. Therefore, the pipelined version is used throughout this paper.

Figure 8 presents the complete structure of the pipelined reduced NAVF scheme. Two pipelined B-level LUM smoothers (for $k = 7$ and $k = 14$) process B levels of input samples concurrently. Since nine new samples appear at the input of the proposed filter for each updated location (p, q, t), nine input shift register blocks are necessary. The use of two LUM filters necessitates the implementation of two output shift register blocks. Because the central sample is used for computing the output, it has to be delayed by $B + 1$ clock periods in a B-bit shift register. This delay corresponds to the sum of delays of input and output shift registers. The complete reduced NAVF scheme has a $B+3$ latency, because both subtraction block and comparators contain one pipeline register, too.

3.3. Area/cost reduction

Since the LUM filter contains $N \times B$ propagation cells, it should be designed very carefully. One FPLD logic cell (or

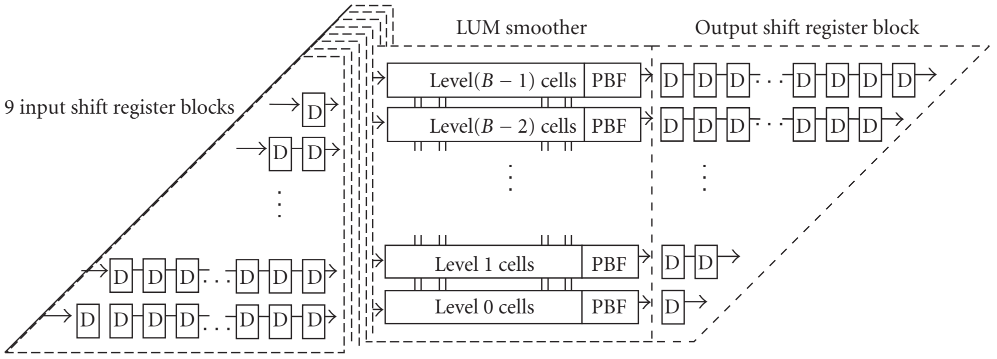

FIGURE 7: Pipelined scheme with input and output shift registers.

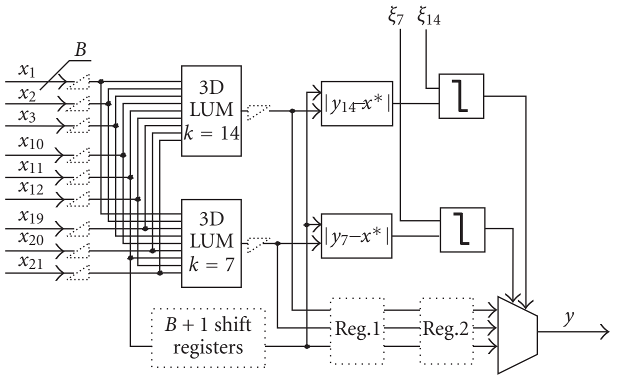

FIGURE 8: Pipelined structure of the reduced NAVF scheme.

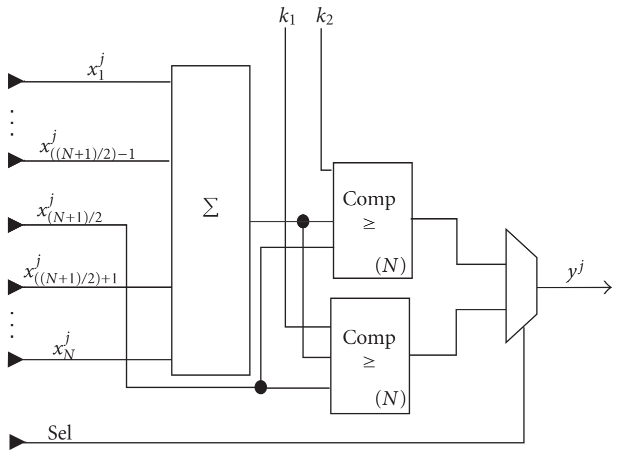

FIGURE 9: N-input double PBF function $f_{k,N}(\cdot)$ internal structure.

bit slice) usually contains one 16-bit lookup table (LUT) followed by a configurable register. We could expect that the propagation cell from Figure 6 will be implemented in three such logic cells. However, the combinatorial function at the output of the multiplexer represents the pipeline register input and at the same time represents the propagation cell output. Therefore, the propagation cell will occupy four logic cells instead of three: one cell where only a register is employed (data bit register), two cells where both LUTs and registers (pipeline registers) are utilized, and one cell with only a LUT employed (output to PBF). Some FPLD technologies enable the output of both the combinatorial function and its registered version from the same logic cell (register packing). This option would lead to a filter-area reduction of up to 20%.

The fact that the reduced NAVF filter from Figure 8 contains two similar LUM smoother structures can be used to further reduce the filter area through efficient resource sharing. This possibility is based on the observation that the most important part of the PBF function area is occupied by the adder from Figure 4. This has the consequence that the PBF function area changes only slightly with the parameter k (about 53 ± 1 logic cells per PBF). For the same reason, the double PBF function block for the two parameters k_1 and k_2, based on the sharing of the adder by two comparators from Figure 9, will increase the size of the block by an insignificant amount (about 55 ± 1 logic cells per double PBF). The price to be payed for this efficient resource sharing is a reduction in speed by a factor of 2, as the binary outputs corresponding to k_1 and k_2 must be multiplexed in time.

Unfortunately, the sharing of the propagation cell is not as successful as it is in the case of the PBF function. Since propagated values of the shared smoothers are not the same, two pairs of the pipeline registers will be necessary (see Figure 10) to propagate the bit and flag for both of them. Two new multiplexers on the vertical outputs of the propagation cell are also added. However, the data register on the input of the propagation cell, the multiplexer, the XNOR, and AND gates remains sharable. Therefore, we can expect that the size of the double propagation cell will grow from three to at least five logic cells. The clock-enable signal employed in the double propagation cell does not influence the overall cell size, because it is a standard signal available in all logic cells in the FPLD device.

3.4. Implementation issues of a complete video processing system

We have used the Stratix EP1S25 DSP board from Altera [77] to verify the overall area and performance of a complete video system based on the proposed filter. Besides the Stratix

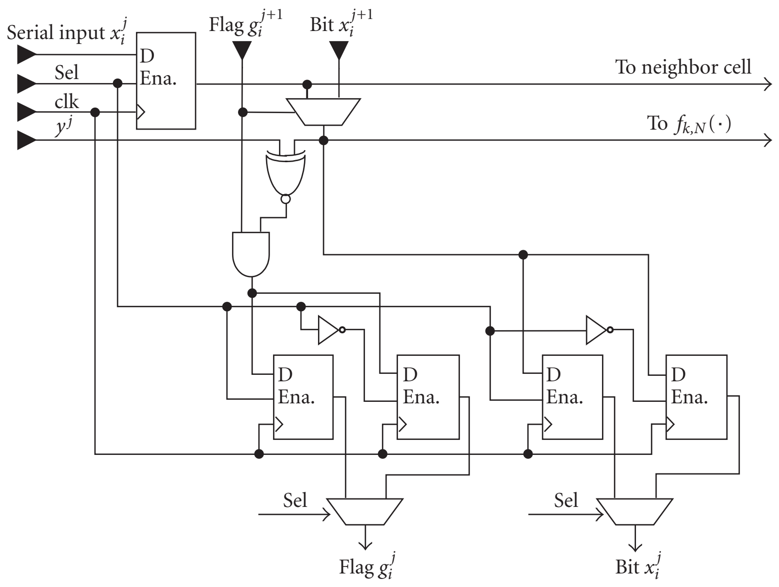

Figure 10: Double-propagation-cell internal structure.

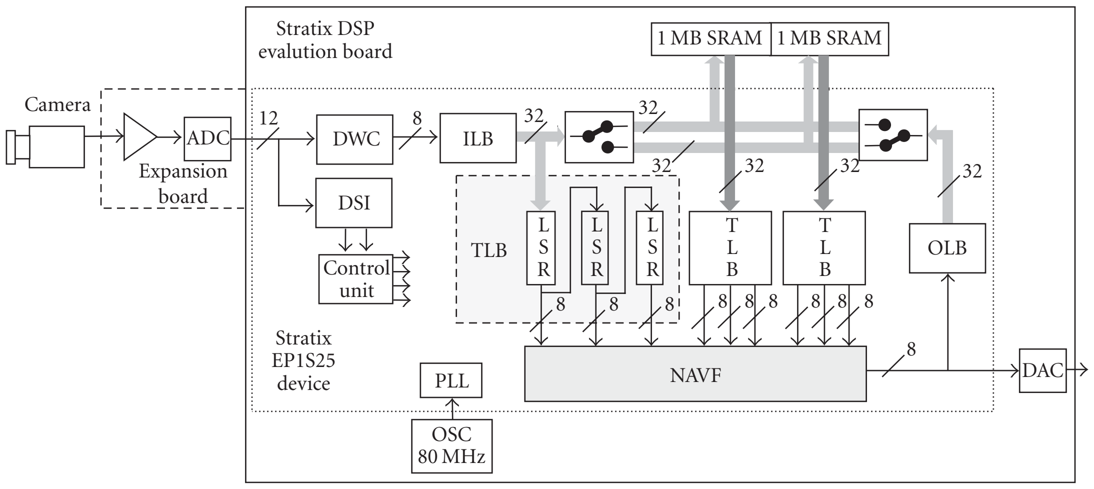

Figure 11: Block diagram of the complete video filtering system based on the 3D reduced NAVF filter.

EP1S25 FPLD being in the fastest speed grade, the board features other components, which has been used in our design: one of two 14-bit 165 MHz D/A converters, two synchronous 1 MB SRAM blocks with independent data and control buses and common address bus, an expansion prototype connector (EPC), oscillator, and so forth. We have placed a video amplifier and a double 12-bit A/D converter with a maximum rate of 50 mega-samples per second to a small expansion board connected to the EPC, because two A/D converters (ADCs) available in the DSP board were not suitable for the video signal conversion. All other hardware blocks were implemented in the STRATIX device (see Figure 11): the detector of synchronization impulses (DSI), 12-bit-to-8-bit data width converter (DWC), input line buffer (ILB), three triple line buffers (TLBs) containing three line shift registers (LSRs) each, two data bus demultiplexers, output line buffer (OLB), control unit, and PLL-based clock generator.

Buffers ILB, OLB and the first LSR of each TLB block use the dual-port feature of the embedded memory blocks. They can therefore have independent read and write frequency. The control frequency of the external memory blocks can thus be much higher than the input/output filter speed. Since the memory data bus is four times wider than input/output data stream, all the memory accesses can be realized during inactive portion of the line. In fact, this interval is divided in two halves: in the first part input data from the camera and output data from the reduced NAVF filter are written simultaneously to two external memory blocks; in the second half of the inactive line portion two lines of two previous images are read to TLB buffers. Since each line buffer can store up to 1024 pixels, any camera having up to 60 images of 1024×1024 pixels can be connected to the system. Please note that the detailed description of the system exceeds the scope of this paper. The reader can find more information in [78].

Table 5: Mapping of $f_{k,N}(\cdot)$ for $k = 7$ and $N = 27$ into Altera STRATIX EP1S25-5 device.

Compiler	Parameter	
	LCs	t_{pd} (ns)
Quartus II v. 4.1	54	12.9

4. RESULTS

The basic structures and blocks utilized in the reduced NAVF have been described in very high-speed integrated circuit hardware description language (VHDL). Filter components have been synthesized using Altera Quartus II v. 4.1 VHDL compiler. We have chosen the Quartus II development system to implement the complete filtering scheme, because it enables good control over compilation, placement, and routing parameters, namely, register packing and shift-register implementation in embedded memory. VHDL output generated by the fitter was used as simulation input for ModelSim v. 5.8c VHDL simulator. Output values have been compared with Matlab-generated test values in an automatic test bench procedure.

The reduced NAVF scheme has been mapped into the Altera STRATIX EP1S25 device. This device is also used in the Altera STRATIX EP1S25 DSP development board, which has been used to implement the whole video system including the filter. We have used power calculator for Stratix devices, version 3.0 from Altera, to estimate typical power consumption of various filter versions for the toggle rate of 12.5%. We have selected a typical power estimation instead of a worst-case one, because the filter structure occupies only a small part of the device (up to 13%) and the standby power consumption is higher (typical standby power consumption of the device is about 135 mW) or much higher (worst-case standby power consumption is about 450 mW) than that of the filter. The standby power consumption estimation precision has thus a dominant influence on the overall precision of the method.

Most of the blocks are realized as parameterized modules. Top-level parameters include the window size corresponding to the spatiotemporal filtering window (default value $N = 27$), the word-length B (8 bits by default), smoothing parameters ($k = 7$ and $k = 14$ by default), and the associated thresholds ($\xi_7 = 15$ and $\xi_{14} = 52$ by default).

4.1. LUM-PBF function implementation results

Since the LUM-PBF function represents a relatively complex N-input combinatorial function and it is included in the critical path of the smoother, it was important to analyze the impact of the smoothing parameter k on the overall complexity and speed of the PBF function realization. Fortunately, the function size and speed change insignificantly with k and both of them are dominated by the adder block present in the function entry, as explained in the previous section. As it can be seen in Table 5, the LUM-PBF function $f_{7,27}(\cdot)$ occupies 54 logic cells (note that carry-chain must be enabled). Table 5 also illustrates the input/output point-to-point delay (t_{pd}) corresponding to the LUM-PBF implementation using the selected STRATIX [79] device. We can conclude that a low number of logic cells and small point-to-point delays of the LUM-PBF function demonstrate the cost efficiency of (8). This is a very important fact, because these parameters determine, to a great extent, the size and especially the clock frequency of the reduced NAVF scheme.

4.2. LUM smoother implementation results

Given $N = 27$, $B = 8$, and $k = 7$, the first two lines of Table 6 allow the quantitative comparison of the standard and pipelined LUM, in terms of logic cell count, static timing analysis frequency, and estimation of the power dissipation. In both of these mappings, register packing was not employed. The pipelined realization achieves a sevenfold increase in processing speed with the tradeoff of a 50% increase in area compared to the parallel version. Higher count of logic cells in the pipelined version is caused by the use of the shift registers, namely, ten input and one output register blocks of $[B \cdot (B+1)/2]$ registers.

The LUM cells matrix area remains very similar in both standard and pipelined filter versions (for 8 bit levels and 27 input elements, the LUM area is realized by less than $27 \times 8 \times 4 = 864$ logic cells). Since 8 PBFs occupy approximately 400 LCs, the total area estimation of the LUM smoother without input/output shift registers (1048) is close to the values obtained for both implementations (subtracting the 360 logic cells necessary for the implementation of the input/output shift registers in the pipelined version).

However, the size of the pipelined version can be significantly reduced using register packing, as shown in the third line of the table. The small reduction in speed is insignificant. Additional area reduction can be obtained by implementing the shift registers in the embedded memory. Although Xilinx devices allow the implementation of up to 16-element shift registers in one LUT, this is not available in Altera devices. Some limited functionality (concerning the minimal length of the chain) exists for the implementation of small shift registers in the M4K or M512 embedded memory blocks available in the STRATIX family [77, 79]. The fourth line of Table 6 presents results obtained using this method of shift register implementation. Using the aforementioned techniques, the pipelined LUM smoother's size becomes comparable to the size of its parallel version, with the pipelined smoother being about six times faster.

It can be observed in Table 6 that the parallel structure (PRS) has the smallest power consumption (note that the typical standby power consumption is 135 mW). However, if we reduce the clock frequency for the pipelined version of the smoother with register packing and embedded shift registers (PPS + RP + ESR) to 14.3 MHz, we will obtain a very similar result (145 mW instead of 142 mW).

4.3. Reduced NAVF scheme implementation results

Table 7 allows for the comparison of the proposed method to the efficient adaptive switching ASM video filtering solution. It can be seen that the proposed architecture consumes significantly less hardware resources compared to the ASM scheme. This advantage is obtained using unique binary

Table 6: Mapping of the LUM smoother for $k = 7$ and $N = 27$ into Altera STRATIX EP 1S25-5 device using the Quartus II v. 4.1 compiler and fitter. The filter has been implemented using the parallel structure (PRS), pipelined structure (PPS), PPS with register packing (RP), and embedded shift registers (ESRs).

Structure	Parameter			
	Area		Speed	Power
	LCs No.	RAM (bits)	f (MHz)	Estimation (mW)
PRS	1090	0	14.3	142
PPS	1605	0	103.5	221
PPS + RP	1260	0	96.4	212
PPS + RP + ESR	899	263	97.2	199

Table 7: Mapping of the ASM scheme and the proposed reduced NAVF scheme into Altera STRATIX EP 1S25-5 device using Quartus II v. 4.1 compiler and fitter. The reduced NAVF scheme has been implemented using the pipelined structure without and with register packing (RP) and embedded shift registers (ESRs).

Structure	Parameter			
	Area		Speed	Power
	LCs no.	RAM (bits)	f (MHz)	Estimation (mW)
ASM	4655	0	91.0	280
Reduced NAVF	2670	0	97.6	243
Reduced NAVF + RP	2108	0	93.2	234
Reduced NAVF + RP + ESR	1736	323	93.2	220
Reduced NAVF + shared LUMs + RP	1715	0	81.7(/2)	222
Reduced NAVF + shared LUMs + RP + ESR	1410	202	82.0(/2)	219

operations required in the LUM smoothers which are effectively utilized in the proposed reduced NAVF structure from Figure 8. The second line of the table presents the results obtained for the pipelined version of the reduced NAVF scheme without register packing and without implementation of the shift registers in the embedded memory. The overall LC count is, in this case, slightly lower than that for the two pipelined LUM smoothers from the second line of Table 6. This is because nine input shift register fields can be shared by the LUM smoothers. Absolute value computation and comparison blocks are realized using the standard Library of Parameterized Modules (functions *lpm_abs* and *lpm_compare*). Since the outputs of these modules are registered, they do not influence the final filter speed. The speed is therefore limited mostly by the PBF implementation and can be as high as 97.6 Mpixels per second. Because the obtained speeds are much faster than required in common video applications, we have concentrated our effort on the reduction of the filter area. As can be observed in the third line of Table 7, register packing remains an efficient method for LUM smoother size reduction (about 20%) while preserving the speed of the filter. The use of embedded shift registers in the STRATIX family can further reduce the logic area (see the fourth line of Table 7). Another significant reduction in the area proposed in this paper (about 16%) can be obtained using the LUM function sharing in a double LUM smoother structure described in the previous subsection. However, the LUM smoother sharing slightly increases the complexity of the PBF function and it thus decreases the clock speed. An even more important fact is that the use of double structures necessitates time multiplexing. The overall speed of the reduced NAVF structure based on the shared LUM smoother is thus two times slower (as indicated by the parenthesis in the fifth and sixth lines of the Table 7). While this reduced filtering speed is still higher than the speed of most conventional video cameras, the proposed reduced NAVF scheme with LUM smoother sharing is the most area- and energy-efficient.

4.4. Complete video filtering system implementation results

The proposed reduced NAVF scheme needs 9 pixels to be available at the filter input at each video sample period. The set of input/output line buffers and data bus-width converter together with a control logic described in Section 3.4 were implemented in the same reconfigurable device. As it can be seen in Table 8, this additional logic occupies few logic cells (about 2% of the cells available in the selected device) and a small amount of RAM bits (about 4% of all available bits). The frequency 123.1 MHz given in the table specifies the maximum clock frequency of the 32-bit data bus. Thanks to the use of the true dual-port embedded memory blocks, this frequency can be independent of the video signal sampling rate. This memory access frequency is high enough to

Table 8: Mapping of the complete video filtering system including control logic, line buffers, and the proposed reduced NAVF scheme into Altera STRATIX EP 1S25-5 device using Quartus II v. 4.1 compiler and fitter. The fastest and the most economic reduced NAVF schemes are considered.

Structure	Parameter					
	Area				Speed	Power
	LCs		RAM		f	estimation
	No.	(%)	(bits)	(%)	(MHz)	(mW)
Filter control + buffers	521	2	90112	4	123.1	—
Fast video filtering system	3191	12.4	90112	4	97.4	283
Economic video filtering system	1918	7.5	90368	4	41.2	249

store incoming lines and to read two lines of the past images from the external memories during inactive portion of the video line. The speed of the complete filtering solution is thus limited only by the used reduced NAVF scheme. Table 8 presents also the area, speed, and power dissipation estimation of the complete systems using the fastest (the reduced NAVF filter) and the most economic version of the filter (the reduced NAVF filter with the LUM sharing, register packing, and embedded shift registers). As it can be seen, the unused part of the device (about 90%) is still big enough to constitute the necessary resources for implementing additional image processing functions, such as compression, analysis, and others utilized in computer vision.

When considering the speed of the system, two parameters have to be taken into account [47]: the delay T_d (sometimes called the latency) is the time from the presentation of a set of input until the output of the results and the period T_p is the time between successive presentations of problem instances. The period T_p of the proposed solution corresponds to the maximum usable sampling period and it is limited by the speed of the reduced NAVF scheme (10.2 nanoseconds and 24.3 nanoseconds for the fast and economic solutions). Since both the fast and economic solutions are faster than the output pixel rate of a common video camera, the data can be filtered in real time. The latency of the proposed solution is defined by the principle of the sliding window and its dimension ($3 \times 3 \times 3$ pixels). The output of the system is therefore delayed by two frames, two lines, and two pixel periods because of the window size, plus $B+3$ pixel periods imposed by the pipelining principle applied in the reduced NAVF filter.

We did not specify power consumption estimates for filter control and buffers (first line of Table 8), because it would be dominated by the standby power and the control unit, and buffers do not represent a stand-alone part of the filtering scheme. However, the consumption estimation of this block is included in the next two lines of the table.

5. CONCLUSIONS

In this paper, an efficient video filtering technique useful for real-time computer vision applications was introduced. The behavior of the filtering scheme under consideration was analyzed in detail with respect to the parameters used. Experimentation with a wide range of test videos and noise intensities showed that the reduced NAVF structure produces excellent results. Moreover, its simple structure suggests the possibility of implementation as a cost-effective FPLD solution, keeping the majority of available resources unused for the implementation of a compact, modern, integrated computer vision system. Recent FPLD devices have the capacity and performance comparable to application-specific integrated circuits (ASICs), while maintaining flexibility and low development costs. The main disadvantages of FPLD devices—higher unit price in high-volume applications and higher power consumption—can be successfully resolved using their mask-programmed equivalents (e.g., HardCopy version of FPLD devices for Altera). Although the choice of the Altera Stratix EP1S25 device was motivated by the use of Altera STRATIX DSP development board, the filter area is so small that it can be mapped into almost any low cost FPLD device (e.g., the smallest Cyclone device [80]). The flexibility of the complete video filtering system structure is only limited by the architecture of the development board and size of the memory blocks (embedded and external memory used for line and frame buffers). The proposed system structure enables modification of the pixel and frame frequency, and is able to process videos with different frame spatial dimensions. Thus, the proposed solution allows for easy adaptation to the camera chosen by the end-user. However, the reduced NAVF filtering structure is more flexible itself. It can be easily adapted to the window size (by the use of the parameter N), to the number of bits per pixel (parameter B), and to the statistics of the processed video (smoothing parameters k_1 and k_2), and thus reused in a large variety of image processing applications. It can be therefore concluded that the efficiency and versatility of the proposed solutions make our video filtering system ideal for a new generation of advanced and intelligent vision systems.

ACKNOWLEDGMENT

The work of R. Lukac is partially supported by a NATO/NSERC Science Award.

REFERENCES

[1] R. Chapuis, R. Aufrere, and F. Chausse, "Accurate road following and reconstruction by computer vision," *IEEE Trans. Intell. Transport. Syst.*, vol. 3, no. 4, pp. 261–270, 2002.

[2] D. A. Forsyth and J. Ponce, *Computer Vision: a Modern Approach*, Prentice-Hall, Englewood Cliffs, NJ, USA, 2002.

[3] R. Kumar, H. Sawhney, S. Samarasekera, et al., "Aerial video surveillance and exploitation," *Proc. IEEE*, vol. 89, no. 10, pp. 1518–1539, 2001.

[4] A. Schweikard, M. Bodduluri, and J. R. Adler, "Planning for camera-guided robotic radiosurgery," *IEEE Trans. Robot. Automat.*, vol. 14, no. 6, pp. 951–962, 1998.

[5] N. K. Ratha and A. K. Jain, "Computer vision algorithms on reconfigurable logic arrays," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 1, pp. 29–43, 1999.

[6] I. Pitas and A. N. Venetsanopoulos, "Order statistics in digital image processing," *Proc. IEEE*, vol. 80, no. 12, pp. 1893–1919, 1992.

[7] R. Lukac, B. Smolka, K. Martin, K. N. Plataniotis, and A. N. Venetsanopulos, "Vector filtering for color imaging," *IEEE Signal Processing Mag.*, vol. 22, no. 1, pp. 74–86, 2005, Special Issue on Color Image Processing.

[8] G. C. Holst, *CCD Arrays, Cameras, and Displays*, JCD Publishing and SPIE Optical Engineering Press, Bellingham, Wash, USA, 2nd edition, 1998.

[9] J. C. Brailean, R. P. Kleihorst, S. Estratiadis, A. K. Katsaggelos, and R. L. Lagendijk, "Noise reduction filters for dynamic image sequences: a review," *Proc. IEEE*, vol. 83, no. 9, pp. 1272–1292, 1995.

[10] C. Kotropoulos and I. Pitas, "Adaptive LMS L-filters for noise suppression in images," *IEEE Trans. Image Processing*, vol. 5, no. 12, pp. 1596–1608, 1996.

[11] Y. Neuvo and W. Ku, "Analysis and digital realization of a pseudorandom Gaussian and impulsive noise source," *IEEE Trans. Commun.*, vol. 23, no. 9, pp. 849–858, 1975.

[12] V. R. Jain and S. N. Gupta, "Impulsive noise in digital communication," *International Journal of Electronics and Communications*, vol. 47, no. 4, pp. 255–259, 1993.

[13] W. Henkel, T. Kessler, and H. Y. Chung, "Coded 64-CAP ADSL in an impulse-noise environment-modeling of impulse noise and first simulation results," *IEEE J. Select. Areas Commun.*, vol. 13, no. 9, pp. 1611–1621, 1995.

[14] K. N. Plataniotis and A. N. Venetsanopoulos, *Color Image Processing and Applications*, Springer-Verlag, Berlin, Germany, 2000.

[15] J. Astola and P. Kuosmanen, *Fundamentals of Nonlinear Digital Filtering*, CRC Press, Boca Raton, Fla, USA, 1997.

[16] I. Pitas and A. N. Venetsanopoulos, *Nonlinear Digital Filters, Principles and Applications*, Kluwer Academic, Boston, Mass, USA, 1990.

[17] S. K. Mitra and G. L. Sicuranza, Eds., *Nonlinear Image Processing*, Academic Press, San Diego, Calif, USA, 2001.

[18] C. Kotropoulos and I. Pitas, *Nonlinear Model-Based Image/Video Processing and Analysis*, John Wiley & Sons, New York, NY, USA, 2001.

[19] K. E. Barner and G. R. Arce, *Nonlinear Signal and Image Processing: Theory, Methods and Applications*, CRC Press, Boca Raton, Fla, USA, 2004.

[20] O. Faugeras, "Digital color image processing within the framework of a human visual model," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 27, no. 4, pp. 380–393, 1979.

[21] J. W. Tukey, "Nonlinear (nonsuperposable) methods for smoothing data," in *The Collected Works of J.W. Tukey, Volume II: Time Series*, pp. 837–856, Wadsworth Advanced Books & Software, Monterey, Calif, USA, 1965–1984.

[22] E. J. Coyle, "Rank order operators and the mean absolute error criterion," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, no. 1, pp. 63–76, 1988.

[23] A. Bovik, T. Huang, and D. Munson Jr., "A generalization of median filtering using linear combinations of order statistics," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 31, no. 6, pp. 1342–1350, 1983.

[24] I. Pitas and A. N. Venetsanopoulos, "Adaptive filters based on order statistics," *IEEE Trans. Signal Processing*, vol. 39, no. 2, pp. 518–522, 1991.

[25] G. R. Arce and R. E. Foster, "Detail-preserving ranked-order based filters for image processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 1, pp. 83–98, 1989.

[26] L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo, "Weighted median filters: a tutorial," *IEEE Trans. Circuits Syst. II*, vol. 43, no. 3, pp. 157–192, 1996.

[27] G. R. Arce, "A general weighted median filter structure admitting negative weights," *IEEE Trans. Signal Processing*, vol. 46, no. 12, pp. 3195–3205, 1998.

[28] P.-T. Yu and W.-H. Liao, "Weighted order statistics filters—their classification, some properties, and conversion algorithm," *IEEE Trans. Signal Processing*, vol. 42, no. 10, pp. 2678–2691, 1994.

[29] S. Marshall, "New direct design method for weighted order statistic filters," *IEE Proceedings: Vision, Image and Signal Processing*, vol. 151, no. 1, pp. 1–8, 2004.

[30] R. C. Hardie and C. G. Boncelet, "LUM filters: a class of rank-order-based filters for smoothing and sharpening," *IEEE Trans. Signal Processing*, vol. 41, no. 3, pp. 1061–1076, 1993.

[31] R. Lukac and S. Marchevsky, "Boolean expression of LUM smoothers," *IEEE Signal Processing Lett.*, vol. 8, no. 11, pp. 292–294, 2001.

[32] P. Maragos and R. Schafer, "Morphological filters—Part II: Their relations to median, order-statistic, and stack filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 35, no. 8, pp. 1170–1184, 1987.

[33] J. P. Fitch, E. J. Coyle, and N. C. Gallagher, "Median filtering by threshold decomposition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 32, no. 6, pp. 1183–1188, 1984.

[34] M. Gabbouj, E. J. Coyle, and N. C. Gallagher, "An overview of median and stack filtering," *Circuit Systems Signal Processing*, vol. 11, no. 1, pp. 7–45, 1992.

[35] M. K. Prasad and Y. H. Lee, "Stack filters and selection Probabilities," *IEEE Trans. Signal Processing*, vol. 42, no. 10, pp. 2628–2643, 1994.

[36] J. L. Paredes and G. R. Arce, "Stack filters, stack smoothers, and mirrored threshold decomposition," *IEEE Trans. Signal Processing*, vol. 47, no. 10, pp. 2757–2767, 1999.

[37] J. Zheng, K. P. Valavanis, and J. M. Gauch, "Noise removal from color images," *Journal of Intelligent and Robotic Systems*, vol. 7, no. 3, pp. 257–285, 1993.

[38] R. Lukac, "Adaptive color image filtering based on center-weighted vector directional filters," *Multidimensional Systems and Signal Processing*, vol. 15, no. 2, pp. 169–196, 2004.

[39] B. Smolka, K. N. Plataniotis, and A. N. Venetsanopoulos, "Nonlinear techniques for color image processing," in *Nonlinear Signal and Image Processing: Theory, Methods, and Applications*, K. E. Barner and G. R. Arce, Eds., pp. 445–505, CRC Press, Boca Raton, Fla, USA, 2004.

[40] R. Lukac, K. N. Plataniotis, B. Smolka, and A. N. Venetsanopoulos, "Generalized selection weighted vector filters," *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 12, pp. 1870–1885, 2004, Special Issue on Nonlinear Signal and Image Processing.

[41] G. R. Arce, "Multistage order statistic filters for image sequence processing," *IEEE Trans. Signal Processing*, vol. 39, no. 5, pp. 1146–1163, 1991.

[42] R. P. Kleihorst, R. L. Lagendijk, and J. Biemond, "Noise reduction of image sequences using motion compensation and signal decomposition," *IEEE Trans. Image Processing*, vol. 4, no. 3, pp. 274–284, 1995.

[43] K. J. Boo and N. K. Bose, "A motion-compensated spatio-temporal filter for image sequences with signal-dependent noise," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 3, pp. 287–298, 1998.

[44] F. J. Sanchez-Marin, Y. Srinivas, K. N. Jabri, and D. L. Wilson, "Quantitative image quality analysis of a nonlinear spatio-temporal filter," *IEEE Trans. Image Processing*, vol. 10, no. 2, pp. 288–295, 2001.

[45] J. S. Kim and H. W. Park, "Adaptive 3-D median filtering for restoration of an image sequence corrupted by impulse noise," *Signal Processing: Image Communication*, vol. 16, no. 7, pp. 657–668, 2001.

[46] O. Vainio, Y. Neuvo, and S. E. Butner, "A signal processor for median-based algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 9, pp. 1406–1414, 1989.

[47] D. S. Richards, "VLSI median filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, no. 1, pp. 145–153, 1990.

[48] N. R. Murthy and M. N. S. Swamy, "On the VLSI implementation of real-time order statistic filters," *IEEE Trans. Signal Processing*, vol. 40, no. 5, pp. 1241–1252, 1992.

[49] D. Z. Gevorkian, K. O. Egiazarian, S. S. Agaian, J. T. Astola, and O. Vainio, "Parallel algorithms and VLSI architectures for stack filtering using fibonacci$p$-codes," *IEEE Trans. Signal Processing*, vol. 43, no. 1, pp. 286–295, 1995.

[50] Ch. Chakrabarti and L. E. Lucke, "VLSI architectures for weighted order statistic (WOS) filters," *Signal Processing*, vol. 80, no. 8, pp. 1419–1433, 2000.

[51] R. Lukac, V. Fischer, and M. Drutarovsky, "3-D Adaptive LUM smoother based on reduced set of smoothing levels," in *Proc. 9th IEEE International Conference on Electronics, Circuits and Systems (ICECS '02)*, vol. 3, pp. 871–874, Dubrovnik, Croatia, September 2002.

[52] R. Lukac, V. Fischer, and N. Bochard, "Reduction of smoothing levels in LUM FTC filter structure," *Traitement du Signal*, vol. 21, no. 1, pp. 89–96, 2004.

[53] R. Lukac and S. Marchevsky, "LUM smoother with smooth control for noisy image sequences," *EURASIP Journal on Applied Signal Processing*, vol. 2001, no. 2, pp. 110–120, 2001.

[54] J. Park and L. Kurz, "Image enhancement using the modified ICM method," *IEEE Trans. Image Processing*, vol. 5, no. 5, pp. 765–771, 1996.

[55] A. Beghdadi and K. Khellaf, "A noise-filtering method using a local information measure," *IEEE Trans. Image Processing*, vol. 6, no. 6, pp. 879–882, 1997.

[56] T. Chen and H. R. Wu, "Adaptive impulse detection using center-weighted median filters," *IEEE Signal Processing Lett.*, vol. 8, no. 1, pp. 1–3, 2001.

[57] B. Smolka, M. Studer, M. Stommel, and K. Wojciechowski, "Vector median based color gamma filter," in *Proc. International Conference on Computer Vision and Graphics (ICCVG '02)*, vol. 2, pp. 678–684, Zakopane, Poland, September 2002.

[58] R. Lukac, V. Fischer, G. Motyl, and M. Drutarovsky, "Adaptive video filtering framework," *International Journal of Imaging Systems and Technology*, vol. 14, no. 6, pp. 223–237, 2004.

[59] L. E. Lucke and K. K Parhi, "Parallel processing architectures for rank order and stack filters," *IEEE Trans. Signal Processing*, vol. 42, no. 5, pp. 1178–1189, 1994.

[60] C. C. Lin and C. J. Kuo, "Two-dimensional rank-order filter by using max-min sorting network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 8, pp. 941–946, 1998.

[61] A. Hunter, "Connectionist median filtering networks," *Image and Vision Computing*, vol. 14, no. 4, pp. 277–283, 1996.

[62] R. Maheshwari, S. S. S. P. Rao, and P. G. Poonacha, "FPGA implementation of median filter," in *Proc. International Conference on VLSI Design*, pp. 523–524, Hyderabad, India, January 1997.

[63] D. Akopian, O. Vainio, S. Agaian, and J. Astola, "Processors for generalized stack filters," *IEEE Trans. Signal Processing*, vol. 43, no. 6, pp. 1541–1546, 1995.

[64] K. Chen, "Bit-serial realizations of a class of nonlinear filters based on positive boolean functions," *IEEE Trans. Circuits Syst.*, vol. 36, no. 6, pp. 785–794, 1989.

[65] B. K. Kar, K. M. Yusuf, and D. K. Pradhan, "Bit-serial generalized median filters," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS '94)*, vol. 3, pp. 85–88, London, UK, May–June 1994.

[66] S. C. Hsia and W. C. Hsu, "A parallel median filter with pipelined scheduling for real-time 1D and 2D signal processing," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E83-A, no. 7, pp. 1396–1404, 2000.

[67] M. J. Avedillo, J. M. Quintana, and H. El Alami, "Weighted order statistics filter for real-time signal processing applications based on pass transistor logic," *IEE Proceedings: Circuits, Devices and Systems*, vol. 151, no. 1, pp. 31–36, 2004.

[68] A. Gasteratos, I. Andreadis, and P. Tsalides, "Realization of rank order filters based on majority gate," *Pattern Recognition*, vol. 30, no. 9, pp. 1571–1576, 1997.

[69] I. Hatirnaz, F. K. Gürkaynak, and Y. Leblebici, "A compact modular architecture for the realization of high-speed binary sorting engines base on rank ordering," in *Proc. International Symposium on Circuits and Systems (ISCAS '00)*, vol. 4, pp. 685–688, Geneva, Switzerland, May 2000.

[70] J. Astola, D. Akopian, O. Vainio, and S. Agaian, "New digit-serial implementations of stack filters," *Signal Processing*, vol. 61, no. 2, pp. 181–197, 1997.

[71] L. Breveglieri and V. Piuri, "Pipelined median filters," in *Proc. IEEE Instrumentation and Measurement Technology Conference (IMTC '94)*, vol. 3, pp. 1455–1458, Hamamatsu, Japan, May 1994.

[72] A. Hiasat and O. Hasan, "Bit-serial architecture for rank order and stack filters," *Integration, the VLSI Journal*, vol. 36, no. 1-2, pp. 3–12, 2003.

[73] S. C. Chan , H. O. Ngai, and K. L. Ho, "A programmable image processing system using FPGA," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS '94)*, vol. 2, pp. 125–128, London, UK, May–June 1994.

[74] K. Wiatr, "Pipeline architecture of specialized reconfigurable processors in FPGA structures for real-time image preprocessing," in *Proc. 24th Euromicro Conference*, vol. 1, pp. 131–138, Vasteras, Sweden, August 1998.

[75] R. Lukac, "Binary LUM smoothing," *IEEE Signal Processing Lett.*, vol. 9, no. 12, pp. 400–403, 2002.

[76] V. Fischer, M. Drutarovsky, and R. Lukac, "Implementation of 3-D adaptive LUM smoother in reconfigurable hardware," in *Proc. 12th International Conference on Field Programmable Logic and Applications (FPL '02)*, vol. 2438, pp. 720–729, Montpellier, France, September 2002.

[77] STRATIX EP1S25 DSP Development Board, Data sheet, November 2002, ver.1.0, http://www.altera.com.

[78] P. Pavelka, V. Fischer, V. Fresse, and F. Celle, "Adaptation of Altera Stratix DSP board for real-time stereoscopic image processing," in *Proc. International Conference on Design of Circuits and Integrated Systems (DCIS '04)*, Bordeaux, France, November 2004.

[79] STRATIX Programmable Logic Data Family, Data sheet, August 2002, ver.2.1, http://www.altera.com.

[80] CYCLONE FPGA Family, Data sheet, September 2002, ver.1.0, http://www.altera.com.

Viktor Fischer received the M.S. and Ph.D. degrees in electronics from the Technical University of Košice, Slovak Republic, in 1981 and 1991, respectively. From 1982 to 1991 he was an Assistant Professor at the Department of Electronics, the Technical University of Košice. Since 1991, he has been working at the Jean Monnet University of Saint-Etienne, France, as an Invited Professor in electronics and computer science. In the Laboratoire Traitement du Signal et Instrumentation (TSI), UMR 5516 CNRS/University of Saint-Etienne, he works on signal and image processing, information security, and embedded cryptographic systems. He is also currently working with Micronic in Košice, Slovak Republic, a company oriented toward the development and production of data security hardware and software.

Rastislav Lukac received the M.S. (Ing.) and Ph.D. degrees in telecommunications from the Technical University of Košice, Slovak Republic, in 1998 and 2001, respectively. From February 2001 to August 2002 he was an Assistant Professor at the Department of Electronics and Multimedia Communications at the Technical University of Košice. Since August 2002 he has been a researcher in Slovak Image Processing Center in Dobsina, Slovak Republic. From January 2003 to March 2003 he was a Postdoctoral Fellow at the Artificial Intelligence & Information Analysis Lab at the Aristotle University of Thessaloniki, Greece. Since May 2003 he has been a Postdoctoral Fellow with the Edward S. Rogers Sr. Department of Electrical and Computer Engineering at the University of Toronto in Toronto, Canada. His research interests include digital camera image processing, microarray image processing, multimedia security, and nonlinear filtering and analysis techniques for color image and video processing. Dr. Lukac is a Member of the IEEE Circuits and Systems, IEEE Consumer Electronics, and IEEE Signal Processing Societies. He serves as a Technical Reviewer for various scientific journals and he participates as a member in numerous international conference committees. In 2003 he was awarded the NATO/NSERC Science Award.

Karl Martin received the B.A.S. degree in engineering science (electrical specialty) and the M.A.S. degree in electrical engineering, all from the University of Toronto, Canada, in 2001 and 2003, respectively. He is currently pursuing a Ph.D. in the Edward S. Rogers Sr. Department of Electrical and Computer Engineering, the University of Toronto. His research interests include multimedia security, multimedia processing, wavelet-based image coding, object-based coding, and CFA processing. Mr. Martin is a Member of both the IEEE Signal Processing Society and Communications Society. Since 2003 he has held the position of Vice-Chair of the Signals and Applications Chapter, IEEE Toronto Section.