Hindawi Publishing Corporation
EURASIP Journal on Image and Video Processing
Volume 2011, Article ID 698914, 13 pages
doi:10.1155/2011/698914

Research Article

Three Novell Analog-Domain Algorithms for Motion Detection in Video Surveillance

Arnaud Verdant,[1] Patrick Villard,[1] Antoine Dupret,[2] and Hervé Mathias[3]

[1] *CEA, LETI, MINATEC, 17 Rue des Martyrs, 38054 Grenoble Cedex 9, France*
[2] *ESYCOM-ESIEE Paris, 2, Boulevard Blaise Pascal, Cité DESCARTES, BP 99, 93162 Noisy le Grand Cedex, France*
[3] *IEF, Bâtiment 220, Université de Paris 11, 91405 Orsay Cedex, France*

Correspondence should be addressed to Antoine Dupret, a.dupret@esiee.fr

Received 1 May 2010; Revised 1 October 2010; Accepted 8 December 2010

Academic Editor: Dan Schonfeld

As to reduce processing load for video surveillance embedded systems, three low-level motion detection algorithms to be implemented on an analog CMOS image sensor are presented. Allowing on-chip segmentation of moving targets, these algorithms are both robust and compliant to various environments while being power efficient. They feature different trade-offs between detection performance and number of a priori choices. Detailed processing steps are presented for each of these algorithms and a comparative study is proposed with respect to some reference algorithms. Depending on the application, the best algorithm choice is then discussed.

1. Introduction

Motion detection in video surveillance with CMOS Image Sensors (CIS) requires high performance but it also needs to meet power consumption constraints, especially for remote sensing applications.

One way to address this issue is to design ASICs with specific image processing architectures. It allows some low level local analog processing to be performed at the sensor level (prior to A/D conversion), which is particularly power efficient. Thanks to submicron CMOS processes, the in-sensor processing can be performed without significantly impairing the device resolution and sensitivity. In the case of embedded video surveillance with a major concern on autonomy, such a physical motion detection implementation is a particularly interesting task to investigate since it allows extracting relevant information from a scene prior to broadcasting. This could be used to adapt the sensor's performance such as ADC resolution. Power consumption for capturing, storing, and transmitting the video would so be reduced. However, specific adapted algorithms have to be developed concurrently. Since such sensors have to be fully autonomous, these algorithms have to be both robust and compliant to various environments while being at the same time computationally and power efficient.

In the case of quasisteady camera (video still), adaptive environment modeling constitutes a key point in motion segmentation for surveillance systems. Among many works focusing on computer vision, the visual surveillance problem is discussed in [1], where conventional approaches for motion detection are presented. Implementation of optical flow measurement is also an interesting well-known technique in [2, 3]. These precedent approaches focus on optimizing motion detection in CIS but are not concerned with very low power image processing. In addition, optical flow methods based on Two-Frame Differential Method (i.e., Lucas and Kanade [4] or Horn and Schunk [5]) are based on hypotheses such as illumination steadiness. Such hypotheses are not always relevant, especially when objects move fast with respect to the frame rate. The aperture problem also constitutes a limitation to their straightforward implementation. Hence, these algorithms require iterative multiresolution processing as to extract information.

On the other hand, motion detection achieved by estimating background is based on weaker hypotheses. Background updating is an essential task since real-time

algorithms for embedded systems have to be efficient in a large number of situations, that is able to adapt their sensitivity to the scene. Image segmentation with difference to background and adaptive threshold has been studied in [6], where the signal variance is computed from recursive average computations and then compared to a threshold obtained by averaging background variance over all the pixels. This method has been improved in [7] where its inherent trailing effect is compensated by a confidence weight representing the confidence of a pixel being part of the foreground. Adaptive threshold for motion detection in outdoor environment has been explored in [8]. The histogram of a distant matrix (obtained with Principal Component Analysis technique) and the variance of a mean image allow adapting the threshold level according to outdoor conditions. Other approaches based on multiple background estimations [9] or adaptive background estimation [10] have also been proposed.

All the precedent methods are efficient but require many operations. Due to the reduced processing resources available in CMOS Image Sensors, computational efficiency is so required yet keeping enough robustness. In order to perform low power motion detection in CIS, other methods based on background modeling have been proposed. In [11] low-level motion detection algorithms are presented and in [12], an efficient algorithm based on Σ-Δ modulation for artificial retinas is described. In this work, robustness improvement to false positives is achieved with local thresholding. For each pixel, background estimation and variance are computed with nonlinear operations to perform adaptive local thresholding.

In our proposed motion detection scheme for increased autonomy, such algorithms [11, 12] need to be improved in terms of false positives and detection efficiency while only using low power operations. The developed algorithms based on low-level computations are designed to be implemented on a versatile analog architecture allowing a wide range of operators and compact processing steps. In this paper, after a short presentation of our architectural choices and their consequences on the associated algorithms (part 2), we describe the motion detection algorithms we take as reference (part 3). We then present the developed motion detection algorithms with associated results and estimated power consumption (part 4). Finally, we discuss the algorithms performance from different points of view in order to balance purely simulated results according to targeted application.

2. Constraints and Targeted Architecture

2.1. Programmable Architecture. The considered programmable computational unit (Figure 1) is a low power SIMD machine based on analog processing [13]. It is composed of an $A \times B$ photosensors array to which an array of $A \times (mB)$ analog memory points (Analog RAM) is associated, where m is the number of memory elements per pixel. In our implementation, we have chosen $m = 3$. Indeed, the analog memory is constrained by technological trade-offs such as silicon area and immunity to noise. The capacitive density is linked to technological parameters (with a typical value of $0.9\,\text{fF}/\mu\text{m}^2$). The temporal noise specifications of our architecture also impose a lower bound for capacitance value ($\sqrt{(kT/C)} = 90\,\mu\text{V}$ for a typical value of C about 500 fF). According to these two parameters, 3 memory elements allow to keep reasonable memory area with regard to pixel matrix, while providing enough robustness with regard to noise and impact of parasitic capacitances. A and B may be up to 1024. The so-formed matrix is bordered on one side by a vector of A switched capacitor analog processors. A column of multiplexers selects the column of pixels or memories to be used by the processor. A sequencer, implemented by a digital IP CPU, delivers the successive processor instructions. For each processor instruction, the switches configurations for the OTA and for the associated analog registers are fixed. Hence, motion detection is directly performed on the pixel gray levels (voltage signals). The matrix does not embed Bayer filter. Thus, demosaicing is not required.

This architecture is implemented using a $0.35\,\mu$m CMOS process. It features a $10\,\mu$m pixel pitch with a standard fill factor (30%). With small parasitic capacitors and 3.3 V voltage swing, it constitutes a good compromise with respect to larger or to deep sub-micrometer processes. Moreover, leakages are also reduced compared to more advanced technologies, thus reducing static power consumption as well as defects in Analogue RAM (ARAM).

In order to take advantage of the SIMD architecture parallelism, the motion segmentation has to be performed independently for each pixel. The corresponding processing so requires many identical operations to be performed iteratively. Provided that the variables involved in the computations are independent, a parallel implementation of algorithms is thus possible and interesting in order to reduce the global power consumption. An analog-based computational system is an efficient response to these constraints.

With such an architecture, performing motion detection algorithms in the analog domain can be achieved with little power requirements. For example, mixing capacitors charges at pixel level [14] efficiently performs pixel averaging. A digital counterpart implementation would require numerous computations and power consuming data transfers.

The chosen programmable architecture globally enables the implementation of "simple" algorithms at a much reduced power cost. "Simple" is to be understood as stepwise linear algorithms based on a reduced temporal or spatial convolution kernel. From available basic operators, different low level algorithms can be implemented by suitably programming the architecture. The various operations required by our algorithms can be performed with this parallel architecture, relying on

(i) pixel average,

(ii) recursive average (i.e., weighted sums),

(iii) fixed step increments/decrements,

(iv) storage (state).

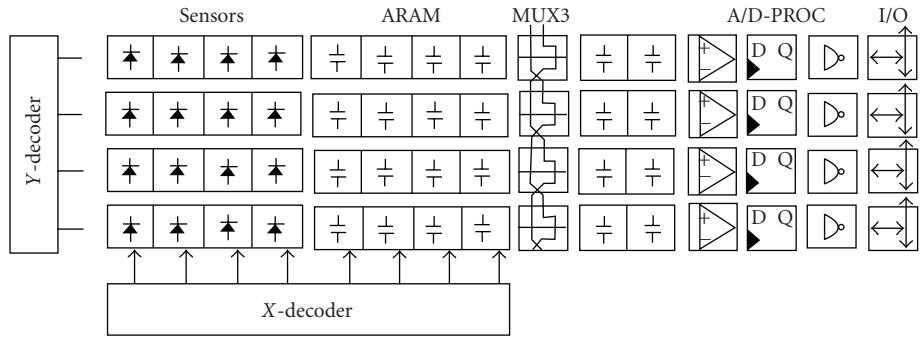


Figure 1: Sensor architecture.

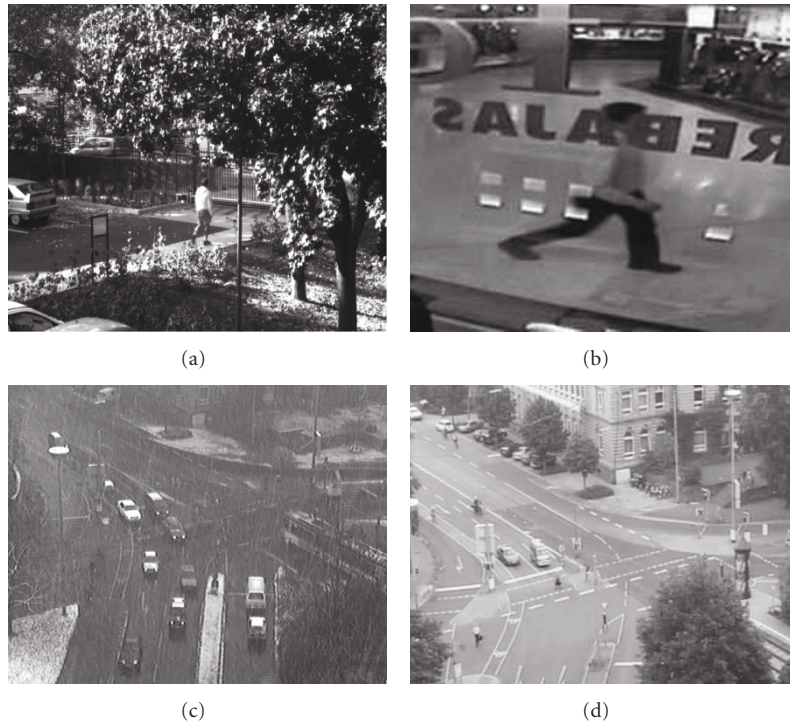


Figure 2: Tested sequences for motion detection.

The most used operators are addition, multiplication of a variable by a fixed coefficient, increment, absolute value, and comparison. Conditional operations are needed, their executions depending upon comparison results referred to states.

Our analog-based architecture has been shown to overcome its digital counterparts in [15] in the context of a low power CMOS image sensor based on a waking up scheme for which the presented algorithms have been optimized.

2.2. Methodology. Concluding on algorithm performance is achieved by measuring motion detection performance on Matlab, as well as induced power consumption and temporal noise effect of CMOS devices using a SystemC model of the system (architecture and algorithm).

As to validate our algorithms performance, we have used different 8 bit sequences representative of indoor and outdoor conditions: *Walk* (IEF's sequence, rustling foliage), *Pets 2002* (strobe light), *dtneu_schnee* (falling snow), and *kwbB* (i21http://www.ira.uka.de/), respectively (a), (b), (c), and (d) on Figure 2 and *Hall Monitor* (Figure 4). For instance, the falling snow in the *dtneu_schnee* sequence and the rustling foliage of *Walk* sequence both introduce parasitic changes of pixels' grey level and constitute realistic tests for the robustness of our algorithms. In our sequences, the objects to be detected are humans or cars.

2.3. Metrics Choice and Performance Evaluation. Performance metrics are based on [16]. During the simulation, motion segmentation is performed on gray level images resulting in binary images containing "moving" and "static" pixels. Each image is then divided in blocks of 10×10 pixels. If a block contains more than a predefined number of moving pixels, this block is then considered as a region

of interest (ROI). From experimental evaluations based on a hand generated ground truth, an ROI can be considered as active when 5 to 10% of the pixels are "moving". Measurements for reference algorithms as well as proposed new ones are based on this value. For each frame, the state of each block is stored in a vector. This vector is compared to a reference which indicates ground truth information for the current frame. The number of True Positives and False Positives and Negatives can thus be counted (TP, FP, TN, FN).

Our considered performance criteria are

(i) Detection Rate (DR = TP/(TP + FN)), which is the ability of the algorithm to detect moving objects,

(ii) False Alarm Rate (FAR = FP/(TP + FP)) which estimates detection quality,

(iii) False Positive Rate (FPR = FP/(FP + TN)), which is representative of algorithm robustness.

In our sequence, nonrelevant motion concerns static elements of the scene or other elements such as snow in *dtneu_schnee* sequence, rustling foliage in *Walk* and *kwbB* sequences and strobe light in *Pets 2002* sequence.

We have developed a faithful, Cycle Accurate, SystemC behavioral model of the architecture [17]. This model enables to jointly simulate the proposed algorithms and the processing architecture. This SystemC modeling is used to determine the number of instructions and the instruction rate required for each algorithm. The SystemC modeling also enables checking the consistency between the results obtained by the model and purely algorithmic results. A log file allows tracing instructions and data, hence enabling to check the whole coherence of the architecture for any conflicts during the parallel processing.

In order to take into account the impact of the nonidealities introduced by the analog parts and to get an accurate evaluation of power consumption, the analog blocks composing the architecture have been described at a low level, down to simple components like switches, capacitors, OTAs. For all these elementary blocks, relevant nonidealities have been modeled with respect to the target CMOS technology and validated thanks to classical electrical simulations (Spice-like). The power consumptions given in the next parts derive from this SystemC modeling of our architecture. Some hints about these aspects of the works have been exposed in [17].

3. Starting Point: ΣΔ and RA Algorithms

The embedded power motion detection algorithms have to meet two requirements: limited complexity, as to comply with our CIS computational limitations and high performance. In order to perform adaptive motion detection, background modeling has been chosen because of its computationally efficient implementation. In [11], two techniques allowing adaptive background modeling are presented. These algorithms perform local computations (i.e., from each pixel value) in order to generate low pass filtering on the observed scene. Approaches based on connected-component extraction, object merging, clustering are not explored here, because they require too intensive calculations with regard to the aimed architecture.

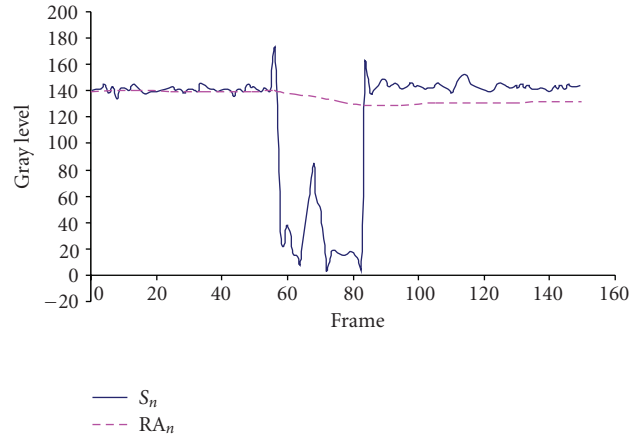


Figure 3: Background estimation (RA_n) with recursive average filtering for a temporal pixel variation (S_n) as a function of time.

3.1. Background Estimation Using ΣΔ and Recursive Average Algorithms. The autonomous remote CIS we develop must perform motion detection in unknown and potentially changing environments. In such configurations, algorithms must meet hard constraints of robustness and adaptability. Markovian algorithms are generally used to face these situations. However, with respect to the considered power consumption and computational constraints, we had to simplify algorithms of this class while preserving their robustness.

As reference algorithms, we consider the Recursive Average (RA) algorithm and the ΣΔ algorithm, respectively, presented in [11, 12]. Both feature simple arithmetic computations. Moreover, the ΣΔ algorithm, which follows the Markov model and has been used for real-time implementations in [18, 19], provides high robustness.

3.1.1. Recursive Average: Principle. A first technique exposed in [11] relies on recursive operations. Considering a pixel value S_n (from 0 to 255), its background estimation RA_n is obtained from (1), with a large time constant fixed by N.

$$RA_n = RA_{n-1} - \frac{1}{N}RA_{n-1} + \frac{1}{N}S_n. \quad (1)$$

As to evaluate the impact of time constants and other algorithm parameters, we plot the temporal variations of a pixel grey level along with its filtered output. The slower the to be detected object, the higher the required time constant. Figure 3 illustrates low pass filtering of a pixel signal using RA. Not surprisingly from Figure 3, we can see that a proper choice of N, depending on frame rate, enables to extract background from moving objects. Yet this representation will help us explain the other algorithms. The visual impact of N is shown on Figure 4 showing estimated background with two different time constants.

(a)

(b)

(c)

FIGURE 4: Estimated background from an original image (a) (*Hall Monitor* sequence), with $N = 25$ (b) and $N = 28$ (c).

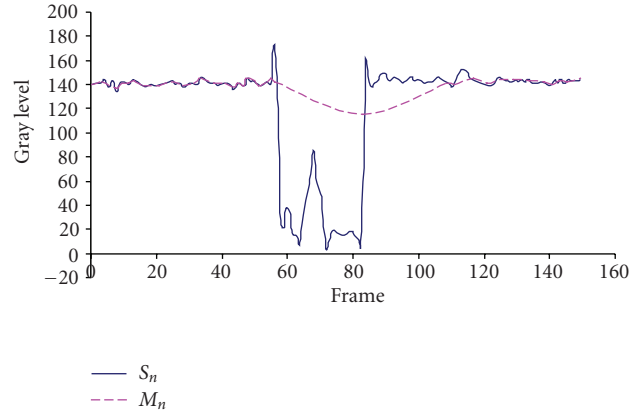


FIGURE 5: Background estimation with Σ-Δ modulation. S_n is the pixel gray level value, M_n is the estimation of the background as a function of time.

FIGURE 6: Result of background estimation on *Hall Monitor* sequence with Σ-Δ modulations. Notice the trailing effect generating a "ghost".

Motion is then considered when the absolute difference between the estimated background and the processed pixel level is greater than a static global threshold (2).

$$\text{if } |\text{RA}_n - S_n| \geq \text{threshold} \longrightarrow \text{motion}. \tag{2}$$

This algorithm so performs basic motion detection while being well suited for our analog implementation. However, local thresholding must be considered to improve robustness. Motion detection performance is exposed on Table 1.

3.1.2. ΣΔ: Principle. The second method presented in [12] is based on nonlinear operations with Σ-Δ modulations. According to successive comparisons with signal value (3), a variable M_n is here incremented (4) or decremented (5) by a constant value so as to fit the pixel level S_n.

$$\Delta_n = M_{n-1} - S_n \tag{3}$$

$$\text{if } \Delta_n > 0 \longrightarrow M_n = M_{n-1} - 1 \tag{4}$$

$$\text{if } \Delta_n < 0 \longrightarrow M_n = M_{n-1} + 1. \tag{5}$$

As for RA on Figure 3, Figure 5 illustrates low pass filtering of a pixel signal with Σ-Δ modulation method.

Considering an analogue implementation, the main advantage of this method is that it features more flexibility than the RA algorithm. Indeed, estimated background variations can be adjusted by incrementation/decrementation steps, whereas time constant values of recursive averages are limited by the physical implementation of the computation. In our architecture, these time constant values are fixed by the ratios of the capacitances on which the signals charges are shared.

Figure 6 shows the estimated background obtained with Σ-Δ modulations on the *Hall Monitor* sequence.

For motion detection, based on the same modulations than (4) or (5), a variable V_n is generated. It can be interpreted as the signal variance and allows to threshold the absolute difference Δ_n between the pixel signal S_n and the estimated background M_n (Figure 7). Motion is detected when Δ_n is higher than V_n.

$$\begin{aligned} &\text{if } V_n > N \cdot \Delta_n \longrightarrow V_n = V_{n-1} - 1, \\ &\text{if } V_n < N \cdot \Delta_n \longrightarrow V_n = V_{n-1} + 1, \\ &\text{if } \Delta_n > V_n \longrightarrow \text{motion}. \end{aligned} \tag{6}$$

Instead of the global threshold used in RA, the ΣΔ algorithm so computes a local adaptive threshold for each

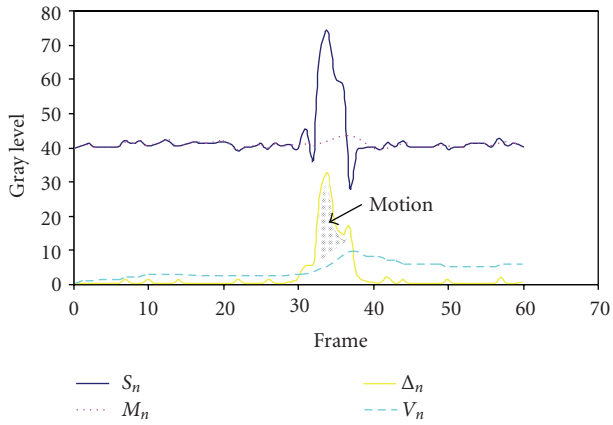


Figure 7: ΣΔ algorithm. S_n is the pixel gray level value and M_n the background estimation, and V_n the threshold of Δ_n.

Table 1: Motion detection performance of two state-of-the-art algorithms.

	Grey level sequence	Performance metrics (%)	
		RA	ΣΔ
Detection Rate (DR)	*Hall*	97.3	94.2
	kwbB	97.8	94.6
	Walk	100	99.1
	Pets 2002	95.8	93.3
	dtneu_schnee	99.9	91.6
False Alarm Rate (FAR)	*Hall*	79.3	16.3
	kwbB	81.7	32.4
	Walk	84.8	86.7
	Pets 2002	85.0	28.3
	dtneu_schnee	54.8	43.7
False Positive Rate (FPR)	*Hall*	42.0	2.5
	kwbB	15.4	2.7
	Walk	59.2	60.5
	Pets 2002	16.5	1.6
	dtneu_schnee	24.3	14.5

pixel as to achieve more robustness on noisy elements, while keeping enough sensitivity on static background. Thanks to the observed scene nonuniformity, local thresholding is computed according to the temporal activity of each zone. Moreover, this algorithm features no trailing effects, at the cost of a poor band pass filtering capability.

3.1.3. Recursive Average and ΣΔ Performance. Table 1 presents the motion performance of state-of-the-art algorithms. The N value used for the RA algorithm is 2^5. The N value used for the ΣΔ algorithm (required for threshold processing) is 15.

RA exhibits poor robustness. Indeed, this algorithm requires setting a global threshold that constitutes the main limitation of this method since no sensitivity adaptation according to scene activity can be performed. Moreover, RA exhibits phase shifting resulting in trailing effects and poor band pass filtering. More specifically, this algorithm does not allow high frequency rejection along with background subtraction.

The motion detection performance exposed for the ΣΔ algorithm clearly shows the interest of local adaptive thresholding compared to the global one used by the RA algorithm.

However, the on-chip motion detection information can be used to adapt the sensor performance (e.g., higher ADC accuracy on moving pixels). In order to keep a reasonable global power consumption (a few mW), an improved robustness of these on-chip motion detection analog domain algorithms is still required while keeping high detection rate.

4. Algorithms

We now describe our three designed motion segmentation algorithms for CIS:

(i) a first algorithm running with no a priori determination of constant, based on scene activity to adapt its sensitivity,

(ii) a second algorithm using band pass filtering in order to reduce false positives upon high frequency pixel variations,

(iii) finally, an algorithm featuring only one constant to determine a priori, and reducing the trailing effect induced by recursive averaging.

4.1. Scene-Based Adaptive Algorithm (SBA). In order to improve adaptability, we now present the Scene-Based Adaptive (SBA) algorithm. This algorithm derives from the ΣΔ algorithm in [12]. It performs motion segmentation on gray level sequences with no a priori constant determination, like the N constant used in ΣΔ. Based on Σ-Δ modulations, the SBA algorithm is also compliant with the reduced available computational resources of CIS architectures, thus eliminating true Markovian approaches.

Our idea is to get rid of constants related to the background of the scene. The detection of grey level variations resulting from motion derives from the absolute difference Δ_n between the last extremum and the current pixel value S_n (Figure 8). Instead of detecting grey level variations like in (4) and (5), this filter requires no constant setting.

The Δ_n value generated is now used to perform adaptive motion detection with the technique presented below.

First, the mean value $M1_n$ of Δ_n is computed (7). Considering that insignificant motions of the background introduce only small variations changes, the idea is to favor large signal variations at the expense of small ones. A convex function is so needed to amplify $M1_n$. Therefore, (8) introduces $M2_n$ which is an approximation of $M1_n^2$. Indeed, our switched capacitor architecture enables only multiplication between a digital number (i.e., the steps of Δ_n) and an analog value (i.e., $M1_n$).

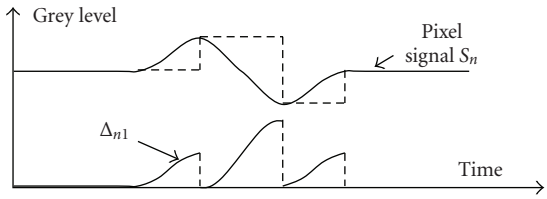


Figure 8: Extracting the signal's variations (Δ_n) according to SBA.

In order to reduce the trailing effects, the next step consists in building an adjustable increment, much like in adaptive ΣΔ. A third variable $M3_n$ is thus obtained from the signal value (9). Indeed, $M3_n$ derives from a Σ-Δ modulation of the signal value using an increment equal to $M2_n$. If the absolute difference between $M3_n$ and S_n is larger than $M2_n$ (10), then the pixel variation is reckoned as relevant and motion is detected.

$$\begin{aligned} &\text{If } M1_{n-1} < \Delta_n \longrightarrow (M1_n = M1_{n-1} + 1) \\ &\quad \text{else if } M1_{n-1} > \Delta_n \longrightarrow (M1_n = M1_{n-1} - 1) \end{aligned} \tag{7}$$

$$\begin{aligned} &\text{if } M2_{n-1} < M1_n \cdot \Delta_n \longrightarrow (M2_n = M2_{n-1} + 1) \\ &\quad \text{else if } M2_{n-1} > M1_n \cdot \Delta_n \longrightarrow (M2_n = M2_{n-1} - 1) \end{aligned} \tag{8}$$

$$\begin{aligned} &\text{if } M3_{n-1} < S_n \longrightarrow (M3_n = M3_{n-1} + M2_n) \\ &\quad \text{else if } M3_{n-1} > S_n \longrightarrow (M3_n = M3_{n-1} - M2_n) \end{aligned} \tag{9}$$

$$\text{if } |M3_n - S_n| > M2_n \longrightarrow \text{motion}. \tag{10}$$

The absolute difference between S_n and $M3_n$ can be seen as the maximal estimated signal dispersion. A larger variation than the estimated one is considered due to a relevant moving object (10). Apart from the increment or decrement level, this algorithm runs without any a priori fixed constant.

Figure 9 illustrates SBA computations of a pixel signal. In absence of motion, one can notice that $M3_n$ fits S_n ($|M3_n - S_n| = 0$). Compared to ΣΔ, the estimator of the background can have a steeper slope when large signal variations occur. Reciprocally, small changes of the pixel grey level lead to long time constants.

Figure 10 illustrates motion detection performed with the ΣΔ and SBA algorithms. In the presented algorithm, some trailing effect can be observed but with a better robustness: in this illustration, the rustling foliage is filtered while motion detection is preserved on the pedestrian.

4.2. Recursive Average with Estimator Algorithm (RAE). In various outdoor situations, many false alarm sources can be encountered. Despite the fact that the static background encountered in urban area does not provide such constraints, weather conditions in the same areas can lead to increased FPR and FAR. In [12], no high frequency rejection is performed, thus implying numerous false positives.

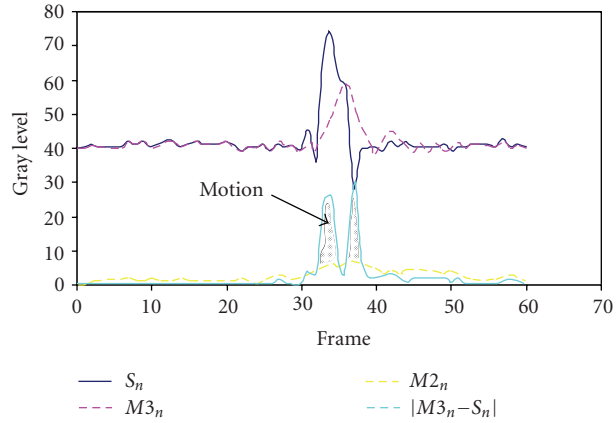


Figure 9: Second computation of a pixel signal with SBA algorithm. S_n is the pixel gray level value, with $M2_n$ and $M3_n$ as, respectively, expressed in (8) and (9).

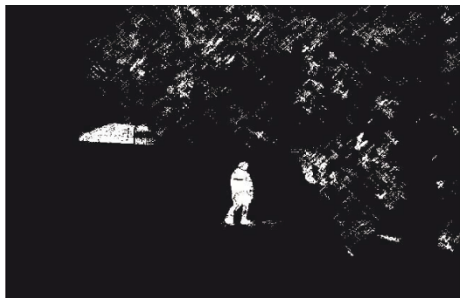
Figure 12(b) illustrates motion detection, performed at a crossroad under falling snow, with the ΣΔ algorithm. In order to improve motion detection robustness by rejecting high frequency variations, we have designed an algorithm featuring band pass filtering. It is also based on recursive average which can be compactly implemented considering charge transfer between capacitances. Though having the same degree of complexity, the designed algorithm is thus optimized for an analog-based architecture, compared to delta modulation.

4.3. Recursive Average with Estimator Algorithm (RAE). In various outdoor situations, many false alarm sources can be encountered. Despite the fact that the static background encountered in urban area does not provide such constraints, weather conditions in the same areas can lead to increased FPR and FAR. In [12], no high frequency rejection is performed, thus implying numerous false positives.

Figure 12(b) illustrates motion detection, performed at a crossroad under falling snow, with the ΣΔ algorithm. In order to improve motion detection robustness by rejecting high frequency variations, we have designed an algorithm featuring band pass filtering. It is also based on recursive average which can be compactly implemented considering charge transfer between capacitances. Though having the same degree of complexity, the designed algorithm is thus optimized for an analog-based architecture, compared to delta modulation.

This algorithm is thus based on a background estimation extracted from the difference between two low pass filters. The computation of two recursive averages ($RA1_n$ (12) and $RA2_n$ (13)), each with its own time constant (fixed by the N and M parameters), allows here to define a band pass filter: the slowest is used to bring out the background while the other, with short lag, filters out the signal's fast perturbations. For each pixel, the main computation steps are described below. n represents the frame index, S_n the current gray level

(a)



(b)

(c)

Figure 10: (a) Original image, (b) Motion detection with ΣΔ, and(c) Motion detection with SBA.

value for the considered block, and $k \cdot \delta_n$ a local threshold (14).

$$\mathrm{RA1}_0 = S_0, \quad \mathrm{RA2}_0 = S_0, \tag{11}$$

$$\mathrm{RA1}_n = \mathrm{RA1}_{n-1} - \frac{1}{N}\mathrm{RA1}_{n-1} + \frac{1}{N}S_n, \tag{12}$$

$$\mathrm{RA2}_n = \mathrm{RA2}_{n-1} - \frac{1}{M}\mathrm{RA2}_{n-1} + \frac{1}{M}S_n \tag{13}$$

$$\text{if } \Delta_n = |\mathrm{RA1}_n - \mathrm{RA2}_n| > k \cdot \delta_n \longrightarrow \text{motion}. \tag{14}$$

An adaptive threshold based on the temporal variations of this absolute difference allows detecting motion. If this estimator Δ_n becomes larger than a local threshold $k \cdot \delta_n$, which depends on the Δ_n temporal activity, motion is detected. Δ_n acts as a band-pass filter selecting only moving objects of interest in the scene. The adaptive threshold is obtained by using δ_n, the recursive average of Δ_n, as a variable amplifying gain for the threshold (17). The increase of the threshold level $k \cdot \delta_n$, due to signal variations, can

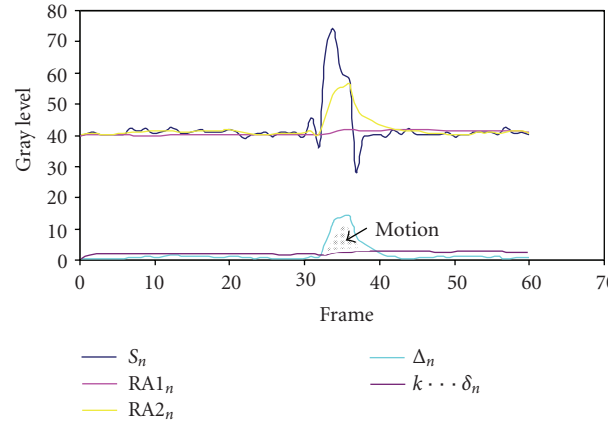


Figure 11: Computation of a pixel signal with the RAE algorithm. S_n is the pixel gray level value with the variables $\mathrm{RA1}_n$, $\mathrm{RA2}_n$, Δ_n, and δ_n as, respectively, expressed in (12), (13), (14), and (17).

be seen on Figure 11. With this method, $k \cdot \delta_n$ directly depends on Δ_n perturbation level, periodicity or persistence. To prevent saturation (considering either analog or fixed point implementation), δ_n is amplified rather than Δ_n. The time constant of this threshold must be quite large with respect to pertinent scene motions in order to adapt the sensitivity to persistent perturbations only.

These recursive operations with few memory requirements make this algorithm easy to implement on our architecture. The time constant for fast recursive average can be determined in order to allow an efficient fast perturbations filtering while not inducing significant trail effect. Considering the z-transform of the recursive average, the time constant is given as follows:

$$\frac{\mathrm{RA1}(z)}{S(z)} = \frac{z}{N(z-(1-1/N))} = \frac{z}{N(z-e^{-T_e/\tau})}, \quad \text{with } \tau = \frac{-Te}{\ln(1-1/N)}. \tag{15}$$

The response to a step function with amplitude A of the transfer function defined by Δ_n is expressed in (16), with N and M being the constants used in (12) and (13).

$$\Delta_n = |\mathrm{RA1}_n - \mathrm{RA2}_n| = A \cdot \left| \left(\frac{M-1}{M}\right)^{n+1} - \left(\frac{N-1}{N}\right)^{n+1} \right|. \tag{16}$$

In this algorithm, the two constants (M, N) depend on the to-be detected objects properties (i.e., size and speed) and on the frame rate. However, knowing the type of object to be detected, local adaptive thresholding is achieved. In the following section, these (M, N) constants have been, respectively, set to $(2^2, 2^4)$ for the simulations performed on the reference sequences, with a 25 Hz frame rate. The class of objects to detect here are cars or pedestrians. The power of two based sizing for M and N facilitates our analog implementation with regard to component matching. With $M = 2^4$, the 95% rise time is $3\tau = 1.533$ s

which corresponds approximately to 50 frames at 25 fps. Considering tested videos, this value has experimentally shown efficient background estimation. Choosing $N = 4$ is a good compromise between implementation constraints and filtering efficiency (in order not to reduce DR, while improving FAR).

$$\delta_n = \delta_{n-1} - \frac{1}{P}\delta_{n-1} + \frac{1}{P}\Delta_n. \tag{17}$$

The constant P has been set to 2^6 ($3\tau = 6.285$ s or 200 frames). The k constant can be typically set around 2 and can be increased in order to reduce false positives.

Figure 11 illustrates computations of a pixel signal using the proposed algorithm.

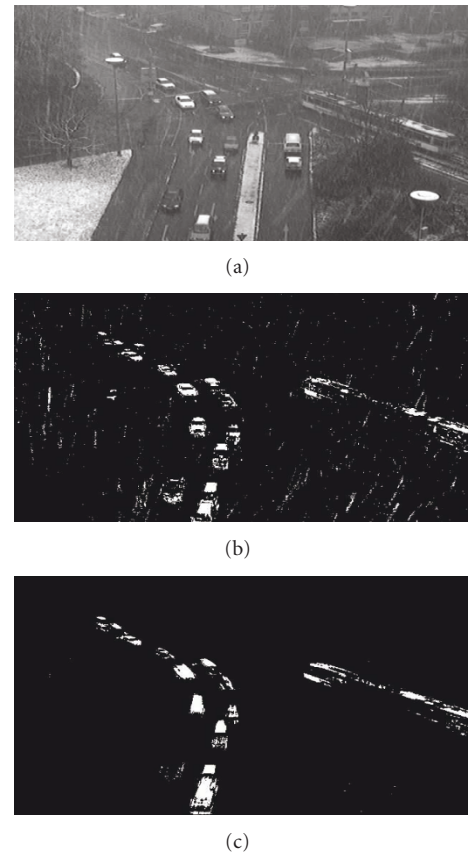
One can notice that this algorithm can bring efficient filtering of high frequency perturbations. However, some trailing effect is observed with the RAE algorithm (not obtained with ΣΔ). Figure 12 illustrates RAE applied on the *dtneu_schnee* sequence with falling snow. With the same sensitivity as ΣΔ, this algorithm allows to filter these high frequency perturbations.

4.4. Adaptive Wrapping Thresholding Algorithm (AWT). Although being robust and computationally efficient, the ΣΔ and RAE algorithms require determining some constants. According to the known frame rate, the *M, N, and P* constants of RAE as well as the increment level of ΣΔ can be determined a priori. However, the RAE k constant or the ΣΔ N constant allows adjusting the algorithm sensitivity in accordance with the amplitude of noisy elements. In order to avoid defining a priori constants, an Adaptive Wrapping Thresholding motion detection algorithm (AWT), based on recursive average operations with a reduced number of constants, is presented in this section. Unlike common algorithms based on recursive low pass filtering [6], this algorithm also limits the trailing effect due to phase shifting.

We thus propose an algorithm based on recursive average operations performing local adaptive thresholding from each pixel signal (Figure 13). In the two precedent algorithms (SBA and RAE), motion detection is performed by thresholding temporal variations (Δ_n). We propose here to compute two wrapping variables in order to detect significant variations of the signal. These two variables are used to define the upper and lower bounds between which the grey level of the signal should remains. In order to take into account the variations of the background, those two variables are updated using a low pass-filter. Yet the time constant of these filters can be much larger than the ones used in ΣΔ and even SBA.

This algorithm relies on a background estimation for each pixel signal from which we estimate the signal standard deviation. This standard deviation is then used to estimate a maximum range for background variations. If the value of a considered pixel moves outside this estimated range of background variations, we consider that motion occurs.

First of all, background estimation ($\mathrm{RA1}_n$) is computed recursively (19). The temporal variations (Δ_n) are extracted as absolute difference between the pixel signal (S_n) and the background estimation (20). The mean deviation of the estimated background variations ($\mathrm{RA2}_n$) is then calculated from (Δ_n) (21). In a fourth step, two variables ($\mathrm{RA3}_n$ and $\mathrm{RA4}_n$) are computed (22) and (23), which allow here to define the estimated range of maximum background variations. Motion is then considered according to (24).



(a)

(b)

(c)

Figure 12: Motion segmentation with the ΣΔ algorithm ($N = 5$) (b) and the RAE algorithm (c).

$$\mathrm{RA1}_0 = S_0; \quad \mathrm{RA2}_0 = 0; \quad \mathrm{RA3}_0 = S_0; \quad \mathrm{RA4}_0 = S_0, \tag{18}$$

$$\mathrm{RA1}_n = \mathrm{RA1}_{n-1} - \frac{1}{N}\mathrm{RA1}_{n-1} + \frac{1}{N}S_n, \tag{19}$$

$$\Delta_n = |\mathrm{RA1}_n - S_n|, \tag{20}$$

$$\mathrm{RA2}_n = \mathrm{RA2}_{n-1} - \frac{1}{N}\mathrm{RA2}_{n-1} + \frac{1}{N}\Delta_n, \tag{21}$$

$$\mathrm{RA3}_n = \mathrm{RA3}_{n-1} - \frac{1}{N}\mathrm{RA3}_{n-1} + \frac{1}{N}(S_n + \mathrm{RA2}_n), \tag{22}$$

$$\mathrm{RA4}_n = \mathrm{RA4}_{n-1} - \frac{1}{N}\mathrm{RA4}_{n-1} + \frac{1}{N}(S_n - \mathrm{RA2}_n) \tag{23}$$

$$\text{if } S_n > RA3_n + \mathrm{RA2}_n \quad \text{or} \quad S_n < RA4_n - \mathrm{RA2}_n \longrightarrow \text{motion}. \tag{24}$$

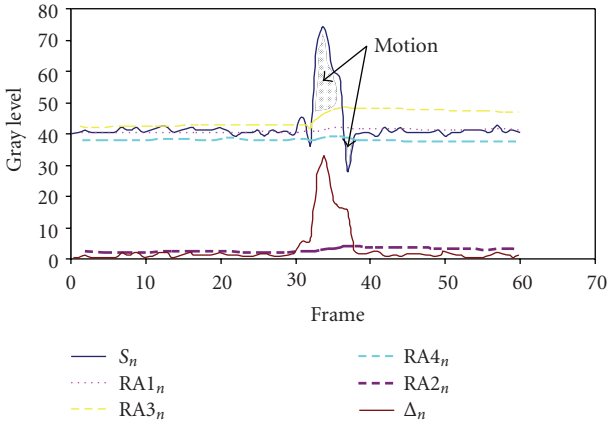


FIGURE 13: Computation of a pixel signal with AWT algorithm. S_n is the pixel gray level value, with the variables $RA1_n$, $RA2_n$, $RA3_n$, $RA4_n$, and Δ_n as, respectively, expressed in (19), (21), (22), (23), and (20).

Hence this algorithm relies on a constant, N, allowing to determine the time constant of recursive averages (equivalent to increment/decrement levels of the $\Sigma\Delta$ algorithm [12]). However, no additional constant is required to handle sensitivity, unlike $\Sigma\Delta$ or RAE where a coefficient is required to set the threshold level. Computations of $RA3_n$ and $RA4_n$ allow here to define adaptive thresholding directly from the signal variations (Figure 13).

Furthermore, this method allows reducing the trailing effect observed with common motion detection algorithms based on recursive average. Indeed, recursive average based on signal level induces phase shifting and trail effect on target. With this algorithm, the double condition in motion detection with $RA3_n$ and $RA4_n$ reduces the trailing effect (Figure 14).

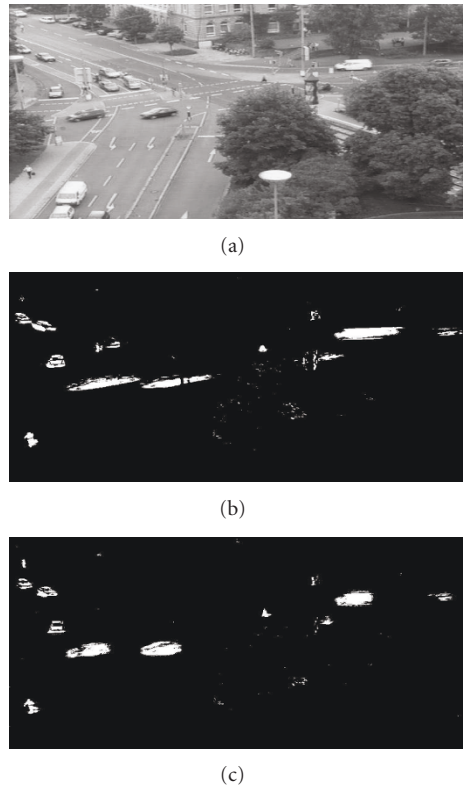
Unlike $\Sigma\Delta$, SBA or RAE, there is no need for a multiplication operation. From our analog implementation point of view, this constitutes an improvement since there is no need to implement multiple capacitors to get a wide range of constants for multiplication.

5. Results

5.1. Algorithms Performance. Table 2 exposes the different results of the state-of-the-art algorithms (RA and $\Sigma\Delta$), as well as new ones (SBA, RAE, and AWT).

Simulations performed on sequences with the SBA algorithm without any arbitrary constant (Table 3) provides quite similar detection rate along with close FAR and FPR measurements, compared to $\Sigma\Delta$ measurements (Table 2). This algorithm thus provides equivalent detection efficiency and robustness, with no need for constant settling, thus showing improved adaptability. Although it does not feature a high frequency rejection, a satisfying detection performance is achieved on gray level sequences.

The results exposed on Table 4 show that RAE is equivalent to $\Sigma\Delta$ in terms of DR for all sequences. However, better results are obtained by our algorithm with respect to



(a)

(b)

(c)

FIGURE 14: Comparison between RA algorithm (b) and AWT (c) algorithm on *kwbB* sequence.

FPR and FAR. This algorithm so features different variables allowing motion segmentation on gray level sequences with a good sensitivity and high frequency rejection. However, a constant k allowing threshold setting is required and some trailing effect is generated.

The AWT algorithm results are slightly below the performance levels of RAE. However, no a priori choice of threshold sensitivity has been made. Hence these results highlight interesting performance about motion detection without environment knowledge.

The *Walk* sequence denotes reduced robustness here. Although rustling foliage is efficiently filtered out by our algorithms, the motion of the tree branches has the same speed and amplitude characteristics as the objects to be detected (e.g., humans). The single processing is not robust to such motion.

The power consumption is proportional to the Number of Instructions (NOI). From SystemC simulations applied to 320×240 30 fps video sequences, we have estimated a power consumption below 5 mW for the worst case (SBA algorithm). This is less than the power consumption of a state of the art 3 M samples/s 10-bit Successive Approximation Register (SAR) ADC designed in the same technology, that is between 10 and 20 mW. The SAR are known to be the least power consuming ADC architectures. This validates the relevance of the algorithm architecture codesign since a digital implementation of those algorithms would require such an ADC plus a digital processing unit. Furthermore,

Table 2: Motion detection performance.

	Grey level sequence	Performance metrics (%)				
		RA	ΣΔ	SBA	RAE	AWT
Detection Rate (DR)	*Hall*	97.3	94.2	93.5	94.8	92.8
	kwbB	97.8	94.6	94	96.4	96.6
	Walk	100	99.1	99.3	99.5	99.3
	Pets 2002	95.8	93.3	94.1	93	94.6
	dtneu_schnee	99.9	91.6	90.1	87.5	90.1
False Alarm Rate (FAR)	*Hall*	79.3	16.3	14.9	12.6	16.7
	kwbB	81.7	32.4	27.4	26.4	36.8
	Walk	84.8	86.7	83.4	85.7	85
	Pets 2002	85.0	28.3	43.4	26.2	29.8
	dtneu_schnee	54.8	43.7	54.9	11.9	45.2
False Positive Rate (FPR)	*Hall*	42.0	2.5	2.2	1.8	2.5
	kwbB	15.4	2.7	1.7	1.7	3.0
	Walk	59.2	60.5	46.7	56	52.9
	Pets 2002	16.5	1.6	3.9	1.2	1.6
	dtneu_schnee	24.3	14.5	22.1	1.8	13.3
Number of Instructions		6	30	43	21	32

Table 3: Motion detection performance.

Algorithm	Average parameter variation on 5 sequences (%)		
	DR	FAR	FPR
RA	—	—	—
ΣΔ	−0.9	50.8	161.3
SBA	−13.3	9.2	−11.6
RAE	0.7	4.7	8.9
AWT	−0.2	−4.4	−8.3

Table 4: Average motion detection performance.

Algorithm	Performance metrics (%)		
	FAR	DR	FPR
ΣΔ	41.5	94.6	16.3
SBA	44.8	94.2	15.3
RAE	32.6	94.2	12.5
AWT	42.7	94.7	14.6

our analog processing unit derives from a SAR ADC; therefore, the scaling of the CMOS technology brings the same improvements as for the classical SAR ADC.

So as to take into account technological parameters in these simulations, temporal noise had been added in these sequences via our SystemC model. Indeed, in our architecture, several noise sources create signal variations that can be interpreted as relevant motion. In our model, the 8-bit images are converted into voltage signal on a 1.8 V dynamic range. An additional Gaussian noise with a 1.1 mV standard deviation is added to each image. During processing, a second Gaussian noise source with a 0.25 mV standard deviation is added to each operation to model analog processor nonidealities.

Table 3 presents the impact of noise on analog processing on the different motion detection parameters considered.

We can see that in the case of SBA and ΣΔ, DR is reduced while FAR is increased. For these two algorithms, noise induces less sensitivity on relevant part of the scene, while decreasing global robustness. These results highlight the lower robustness of these two algorithms when implemented in our analog architecture. Concerning the RAE algorithm, both DR and FAR are amplified. This can be due to an insufficient threshold amplification. For AWT algorithm, the whole parameters are decreased. The threshold amplification is too high for this one, leading to less sensitivity on the whole images. However, the noise added on recursive average-based processing (RAE, AWT) induces fewer variations for the selected parameters. Thus we can consider that the recursive average-based methods are more robust than the ones based on Δ modulations (ΣΔ, SBA), when implemented in our analog architecture.

5.2. Discussion. In the precedent part, we have presented 3 robust and fast new algorithms and compared them to the reference ΣΔ algorithm. Based on particular parameters allowing the measurement of motion detection performance, such as detection rate or false positive rate, we have determined the robustness or detection efficiency of these algorithms. The average results for the tested sequences are presented on Table 4.

However, these results must be balanced by some factors. Indeed, we can define some criteria allowing taking into account implementation constraints such as power consumption or other limitations like the kind of targeted application for motion detection algorithm. We have exposed below some of the criteria, which can be found according to

Table 5: Balanced algorithm performance according to selected criteria.

Algo.	Criteria						
	1	2	3	4	5	6	7
RA	−	− −	−	−	− −	+ +	
ΣΔ	−	+	−	+	±	+	− −
SBA	+	+	−	−	+	−	−
RAE	−	+	+ +	−	+	+	+
AWT	+	+	+	+	±	−	+

motion detection context. Table 5 illustrates the rates of each algorithm according to these criteria.

(1) settings: the fewer the required constants for adapting threshold level or time constants, the more autonomous the left-behind sensor,

(2) adaptation: threshold level evolution according to pixel temporal activity,

(3) high frequency rejection: high frequency noise filtering of pixel signal (band pass filtering),

(4) trailing effect: artefacts or motion segmentation distortion due to phase shifting induced by algorithm,

(5) robustness: number of generated false positives,

(6) computational efficiency: induced power consumption (mainly depending on the number of instructions in our implementation),

(7) robustness with regard to analog implementation (temporal noise).

These qualitative results show that, depending on the aimed application, an algorithm can prevail on another, even if its motion detection performance is worse. However, AWT and RAE are better suited for an analog implementation.

6. Conclusion

Three algorithms developed using a codesign approach have been presented. They perform motion detection at reduced power consumption while ensuring fast and robust computation. Compared to classical sensors performing motion detection downstream the image acquisition, the offered processing capabilities are somehow limited, but the chosen analog architecture, on which they are implemented, offers a better compromise between power consumption and algorithm performance. Moreover, considering only the algorithmic aspect of the works, significant improvements have been brought in terms of self-adaptability to the scene. Constants involved in the presented algorithms are indeed mostly depending on the nature of the objects to be detected (speed and size).

Though these algorithms have been tailored for a dedicated architecture, a real-time implementation on a standard digital processor (e.g., an ARM920T) is however possible but at a significantly higher power consumption (roughly some 100 mW for the processor alone).

Finally, an ASIC is currently being designed as to provide an experimental validation of the concept. One of its main features is that the pixel area ($10 \times 10\,\mu m^2$) is very close to state-of-the-art pixels in similar technology ($0.35\,\mu$m CMOS).

References

[1] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Transactions on Systems, Man and Cybernetics Part C*, vol. 34, no. 3, pp. 334–352, 2004.

[2] A. Moini, A. Bouzerdoum, K. Eshraghian et al., "An insect vision-based motion detection chip," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 2, pp. 279–284, 1997.

[3] S. Mehta and R. Etienne-Cummings, "Normal optical flow measurement on a CMOS APS imager," in *Proceedings of the IEEE International Symposium on Cirquits and Systems (ISCAS '04)*, vol. 4, pp. 848–851, May 2004.

[4] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pp. 674–679, April 1981.

[5] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[6] S. Joo and Q. Zheng, "A temporal variance-based moving target detector," in *Proceedings of the IEEE Workshop on Performance Analysis of Video Surveillance and Tracking (PETS '05)*, January 2005.

[7] M. F. Abdelkader, R. Chellappa, Q. Zheng, and A. L. Chan, "Integrated motion detection and tracking for visual surveillance," in *Proceedings of the 4th IEEE International Conference on Computer Vision Systems (ICVS '06)*, p. 28, January 2006.

[8] J. F. Vázquez, M. Mazo, J. L. Lázaro et al., "Adaptive threshold for motion detection in outdoor environment using computer vision," in *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE '05)*, vol. 3, pp. 1233–1237, June 2005.

[9] W. Pan, K. Wu, Z. Chai, and Z. S. You, "A background reconstruction method based on double-background," in *Proceedings of the 4th International Conference on Image and Graphics (ICIG '07)*, pp. 502–507, August 2007.

[10] J. Guo, D. Rajan, and E. S. Chng, "Motion detection with adaptive background and dynamic thresholds," in *Proceedings of the 5th International Conference on Information, Communications and Signal Processing*, pp. 41–45, December 2005.

[11] J. Richefeu and A. Manzanera, "Motion detection with smart sensor," in *Proceedings of the 9th Congress Young Searchers in Computer Vision (ORASIS '05)*, May 2005.

[12] A. Manzanera and J. C. Richefeu, "A new motion detection algorithm based on Σ-Δ background estimation," *Pattern Recognition Letters*, vol. 28, no. 3, pp. 320–328, 2007.

[13] S. Moutault, H. Mathias, J. O. Klein, and A. Dupret, "An improved analog computation cell for Paris II, a programmable vision chip," in *Proceedings of the IEEE International Symposium on Cirquits and Systems (ISCAS '04)*, pp. 453–456, May 2004.

[14] M. Massie, C. Baxter, J. P. Curzan, P. McCarley, and R. Etienne-Cummings, "Vision chip for navigating and controlling micro unmanned aerial vehicles," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '03)*, vol. 3, pp. 786–789, May 2003.

[15] A. Verdant, A. Dupret, H. Mathias, P. Villard, and L. Lacassagne, "Adaptive multiresolution for low power CMOS image sensor," in *Proceedings of the 14th IEEE International Conference on Image Processing (ICIP '06)*, vol. 5, pp. 185–188, San Antonio, Tex, USA, September-October 2007.

[16] J. Black, T. J. Ellis, and P. Rosin, "A novel method for video tracking performance evaluation," in *Proceedings of the IEEE Workshop on Performance Analysis of Video Surveillance and Tracking (PETS '03)*, pp. 125–132, October 2003.

[17] A. Verdant, P. Villard, A. Dupret, and H. Mathias, "SystemC validation of a low power analog CMOS image sensor architecture," in *Proceedings of the IEEE North-East Workshop on Circuits and Systems (NEWCAS '07)*, pp. 903–906, August 2007.

[18] L. Lacassagne, M. Milgram, and P. Garda, "Motion detection, labeling, data association and tracking, in real-time on RISC computer," in *Proceedings of International Conference on Image Analysis and Processing (ICIP '99)*, pp. 520–525, Venice, Italy, 1999.

[19] J. Denoulet, G. Mostafaoui, L. Lacassagne, and A. Mérigot, "Implementing motion Markov detection on general purpose processor and associative mesh," in *Proceedings of the 7th International Workshop on Computer Architecture for Machine Perception (CAMP '05)*, pp. 288–293, Palermo, Italy, July 2005.