A Low-Power VGA Vision Sensor with Embedded Event-Detection for Outdoor Edge Applications

Yu Zou, *Member, IEEE,* Massimo Gottardi, *Member, IEEE,* Michela Lecca, and Matteo Perenzoni, *Member, IEEE*

***Abstract*—We report on a low-power VGA vision sensor embedding event-detection capabilities targeted to battery-powered vision processing at the edge. The sensor relies on an always-on Double-Threshold Dynamic Background Subtraction algorithm (DT-DBS). The resulting motion bitmap is de-noised, projected along xy-axes of the array of pixels and filtered to robustly detect moving targets even in noisy outdoor scenarios. The chip operates in Motion Detection (MD), applied on a QQVGA sub-sampled image, looking for anomalous motion in the scene at $344\mu W$, and in Imaging Mode (IM), delivering full resolution gray-scale images with associated Local Binary Pattern (LBP) coding and motion bitmaps at $8fps$ and $1.35mW$. The $4\mu m$ pixel vision sensor is manufactured in a 110nm 1P4M CMOS and occupies $25.4mm^2$.**

***Index Terms*—Low-power vision sensors, motion detection, background-subtraction, event-detection, local binary pattern.**

I. Introduction

LOW power consumption and energy management are of main importance for long-lasting battery-powered sensor nodes. This issue is even more challenging for vision systems at the edge, where visual tasks are executed close to the sensor, given the large amount of information to be managed in real-time and with a limited energy budget.

Recently, some image sensors have been reported performing very low power consumption [1–9] in the order of tens of μW. Nevertheless, in such sensors, off-chip image processing and wireless communication still remain the main sources of power consumption, placed about one or two orders of magnitude above that one of the sensor, therefore they should be used carefully. One approach in this direction is to limit the use of these resources only when really needed, triggered by events occurring in the scene. Here, the most straightforward solution is to provide the image sensor with event detection capabilities so that it generates an alert signal switching ON the external processor, which is normally in idle state, to execute further visual processing. In this regard, different techniques of trigger on-motion have been implemented to activate the external computing resources and to enhance the system energy efficiency [1–5] Nevertheless, the main drawbacks in this approach are the custom pixel design and the reduced performance against the software-based algorithmic counterpart. In fact, while analog processing might be very compact and energy-efficient, the algorithm programmability is very limited as well as its performance, reducing the sensor node reliability and its lifetime. This is especially true in outdoor applications, where the event-detection algorithm needs to cope with noisy scenarios, with the risk of generating a large rate of false positives, that activate the external processor uselessly with a dramatic increase of the system average power consumption. The typical approach of published vision sensors with embedded event-detection rely on simple motion detection algorithms, like Frame Difference (FD) and Background Subtraction (BS), that are easy to be implemented on chip but often unreliable for most real applications. Here, we describe a vision sensor for real-time event detection in outdoor scenarios [10]. The sensor embeds a double-threshold dynamic background subtraction algorithm (DT-DBS), running continuously and performing motion detection with energy constraints and it is connected to an external processor which is activated only when a potential event is revealed. While sensors embedding FD and BS exhibit very low-power performance, this work achieves an accurate detection of the event with a low rate of false positives and limited power resources. Based on this result, we aim at introducing a novel framework for the evaluation of low-power sensors, taking into consideration not only their standalone energy efficiency but also their performance in real-world applications. From this point of view, the proposed sensor outperforms other devices in the state-of-the-art. This work focuses on the architecture and performance of the chip-embedded algorithm rather than on the image sensor performance. Although a standard 3T pixel was used, this implementation is fully compliant with a Pinned Photo-Diode (PPD) imager.

The paper outlines as follows. Section II and III describe respectively the chip-embedded algorithm and the overall sensor architecture with its basic building blocks. Section IV presents the experimental results and compares the sensor with similar devices from electrical and functional point of view. Finally, Section V concludes the paper.

Y. Zou, M. Gottardi, M. Lecca and M. Perenzoni are with Fondazione Bruno Kessler, Trento 38123 Italy.

II. Double-Threshold Dynamic Background Subtraction and Event Detection

Detecting an event in a video means to identify one or more objects that change their position or appearance over time, i.e. over a sequence of frames. This operation is of great importance for many applications, such as video-surveillance, human behaviour understanding and monitoring, traffic control, robot navigation, and it is particularly challenging because of the wide range of circumstances under which an event can be observed. Variations of the illumination, shadows, noisy acquisitions, changes of perspective and of the motion speed, occlusions, complex and dynamic backgrounds are some of the many factors that must be taken into account to develop a robust algorithm for event detection. BS and FD are two popular methods for which several on-chip implementations exist

[1–4, 6–8]. BS [11] performs event detection by subtracting each frame from a reference image, which models the static environment of the event and which is usually updated across the video in order to manage possible ambient changes. Any pixel whose intensity value is significantly different from the corresponding background intensity value is labeled as *motion pixel*. Mathematically, the motion map H is a binary image defined as follows [11]:

$$H(i) = \begin{cases} 1 & \text{if } d(V(i), B(i)) > TH \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where i is a pixel, V is a frame, B is the background image, d is a function measuring intensity differences and TH is a pre-defined threshold. In the simplest BS implementation, d is the absolute value of the difference between $V(i)$ and $B(i)$. In this case, the on-chip implementation requires a digital frame buffer to store the background image B for long terms in order to be compared with the current frames. A main drawback of BS is its sensitivity to background variations and noise, that often occur in outdoor scenarios. To overcome these issues, more sophisticated methods have been developed, such as adaptive background subtraction techniques, statistic-based approaches, multiple thresholds for motion detection [12–16]

FD can be considered as a special case of BS. In fact, FD defines the motion pixels through the conditional equation (1) but replaces the background image B with a frame V' consecutive to V. The equation (1) with $B = V'$ can be implemented on silicon in a straightforward way, with low power consumption and with low area occupancy. In this case, FD is much more efficient than BS since it does not require to build, save and update a background model. FD is generally more robust to changes of the environment than BS, that needs to be updated accordingly, but as a drawback, it is very sensitive to the frame rate, to the speed of the moving objects and to noise. As a result, in many real-world applications, FD produces a lot of false positives representing a critical parameter especially in energy-efficient systems. Better performance is generally achieved by temporal difference approaches, where any frame V is compared with two or more consecutive frames, e.g. [17], and by block-wise frame difference methods, where the frame differences are computed between image patches instead of pixels, e.g. [18].
Statistical analysis, optical flow information, additional features and post-processing algorithms are also employed to improve the results both on BS and FD [19–21], while some works even propose to combine BS and FD, e.g. [22]. Nevertheless, the computational pipeline of such approaches is extremely hard to be embedded on a low-power sensor with memory constraints.

To overcome the mentioned limitations, this work proposes a novel vision sensor architecture that embeds a DT-DBS algorithm for the real-time detection of moving objects in grey-level videos depicting indoor and outdoor scenarios. The background is here continuously modelled at each pixel by two thresholds, which update over time. A pixel is thus detected as *motion pixel*, or *hot pixel*, if its intensity value falls out of the range bounded by the two thresholds. The resulting motion label map is de-noised through programmable morphological filters applied on the image and on the horizontal and vertical projections of the motion pixels. When the sensor detects sufficiently large regions of motion pixels, it generates an alarm that is sent to the external processor for further actions. Despite the hardware implementation of the algorithm requires additional resources and hardware overhead compared with standard FD and BS algorithms, it has been proved to provide a more reliable result, characterized by a very low rate of false positives, especially in outdoor scenarios. This means that the external processor connected to the sensor is turned ON only when the probability of a real alert is high, granting a highly efficient energy management.

A. Double-Threshold Dynamic Background Subtraction

In the proposed algorithm, the background is represented by two thresholds V_{Max} and V_{Min}, associated to each pixel and updated at each frame. Let V_i be the intensity value at the pixel P_i and let V_{Max_i} and V_{Min_i} be the corresponding threshold values in the background model. The operating principle of

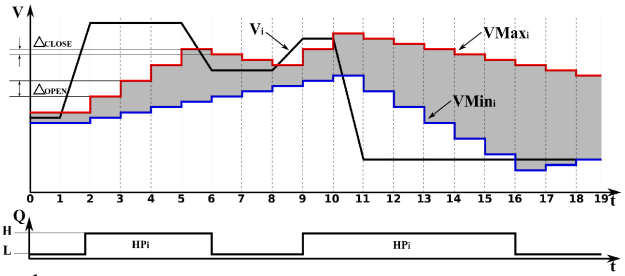


Fig. 1: Operating principle of the sensor embedded DT-DBS. In this case, the pixel voltage V_i changes abruptly while V_{Max_i} and V_{Min_i} approach V_i at two different speeds, according to (2) and (3).

the DT-DBS algorithm of Fig. 1 is described by (2) and (3), which regulate the two thresholds update (V_{Max_i}, V_{Min_i}):

$$\begin{aligned} &\text{if } V_i - V_{Max_i} > 0 \text{ then } V_{Max_i} \leftarrow V_{Max_i} + \Delta_{OPEN}, \\ &\text{if } V_i - V_{Max_i} \leq 0 \text{ then } V_{Max_i} \leftarrow V_{Max_i} - \Delta_{CLOSE}, \end{aligned} \quad (2)$$

$$\begin{aligned} &\text{if } V_i - V_{Min_i} < 0 \text{ then } V_{Min_i} \leftarrow V_{Min_i} - \Delta_{OPEN}, \\ &\text{if } V_i - V_{Min_i} \geq 0 \text{ then } V_{Min_i} \leftarrow V_{Min_i} + \Delta_{CLOSE}, \end{aligned} \quad (3)$$

while (4) defines the conditions for P_i to be hot ($HP_i = 1$), i.e. whose changes are anomalous compared to its past history:

$$(V_i - V_{Min_i} < -\Delta_{HOT}) || (V_i - V_{Max_i} > \Delta_{HOT}), \quad (4)$$

where Δ_{OPEN}, Δ_{CLOSE} and Δ_{HOT} are user-defined algorithm parameters. V_{Max_i} and V_{Min_i} act as two low-pass filters of V_i and have asymmetric behaviours, according to their position with respect to V_i. Their asymmetric behaviour generates a grey-zone between the two thresholds inside which the pixel is considered *normal* (CLOSE). If the pixel is outside the safe-zone (OPEN) and sufficiently far from it, as stated by (4), it is labeled as *hot pixel* (HP_i=1).

B. Post-Processing

The DT-DBS algorithm allows the tuning of the sensitivity of the mechanism that generates the hot pixels so that after a certain time, repetitive intensity variations are ignored.

Variations of the pixel signal caused by moving patterns, such as waves or leaves in the wind, are therefore suppressed after a certain number of frames. More precisely, the time response of the algorithm can be tuned according with the dynamics of the scene. Although this algorithm requires slightly larger computing resources than FD and BS, it is more efficient in suppressing noise produced by irrelevant events such as repetitive motion, which is peculiar of outdoor scenarios. Removing noise at the early stage of image processing allows to discount the computational burden of post-processing algorithms. Nevertheless, our experiments showed that further denoising of the hot pixel map is in general necessary. In our sensor, this operation is performed on-chip by a programmable erosion filter applied on a 3x3 pixel kernel, that removes isolated pixels as well as small connected components. The user-defined erosion threshold must take into account the distance of the sensor from the acquired scene, the characteristics of the scene itself (e.g. forests, urban places) and of the objects to be detected in (e.g. cars, humans). By default, the erosion threshold is fixed to 1 pixel. This value performs well in most of our experiments. After the erosion operation, the resulting hot pixels are projected along the vertical and horizontal axes of the image. These x- and y- projections P_X and P_Y are designed to remove horizontal or vertical wired regions, thus they prevent the alarm generation in case of images containing only sparse linear, tiny aggregations of pixels as those produced for instance by see waves. Therefore, P_X and P_Y act as an additional de-noising filter. In the proposed sensor, P_X and P_Y are stored as 1D vectors and scanned to identify their maximum numbers of contiguous pixels D_X and D_Y. These values undergoes a test to verify the following condition:

$$(T_{X_L} < D_X < T_{X_H}) \wedge (T_{Y_L} < D_Y < T_{Y_H}), \qquad (5)$$

where T_{X_L}, T_{X_H} and T_{Y_L}, T_{Y_H} are the target constrains, stored in the on-chip REGISTERS and usually related to the aspect ratio of the objects to be detected. If D_X and D_Y do not satisfy the condition (5), then no alarm is generated since the moving pixels do not match with the expected size of the objects to be detected.

C. DT-DBS versus FD and BS

The DT-DBS algorithm has been tested on a dataset of 40 gray-level VGA videos in comparison with FD and BS. The videos used in these experiments depict differently lighted outdoor scenarios with moving cars, people and bikes. The brightness of these videos, computed as the average of the frame brightness over time, vary from 52 to 192 levels of intensity, corresponding to low-light and very lighted outdoor scenarios, captured respectively in a shadowed environment and in an open space at noon. The DT-DBS algorithm, as embedded in the chip, has been simulated through a software implemented in C++. Such a software, that exactly reproduces the sensor response, enables a fair, real-time comparison of our DT-DBS approach with two algorithms exploiting FD and BS respectively. In this comparative analysis, FD and BS take as input a VGA video and under-sample it to a QQVGA. FD detects the motion map by computing the map D of the pixel-wise absolute distances between two subsequent frames V_1 and V_2, smoothing D by a Gaussian filter on a 3×3 kernel and thresholding the result so that only pixels in V_2 with a smoothed intensity difference above 2 intensity levels are retained. BS implements the method in [23], that detects events by a statistical analysis. The background is modeled by a Gaussian mixture model that is iteratively updated over time and a pixel is labeled as a motion one if its probability density function does not match that of the background. For each video, the resulting binary motion maps output by our sensor, FD and BS are up-scaled back to VGA. From their qualitative analysis, we observed that: (1) DT-DBS, FD, BS perform similarly on videos characterized by a good illumination and with well contrasted objects (see Fig. 2(a)), while their accuracy on event detection decreases in case of low-light or saturation (see Fig. 2(b)); (2) FD is usually more noisy than BS and DT-DBS, especially when removing the Gaussian filter (see Fig. 2(c)); (3) the performance of BS strongly depends on the background model and on the procedure updating it (see Fig. 2(d)). Some videos showing the motion maps computed by DT-DBS, FD and BS are available starting from the web-page http://tev.fbk.eu/node/183. As a general conclusion, we observe that DT-DBS is more adequate than FD and BS for detecting moving objects in outdoor noisy scenarios.

III. Sensor Architecture

The vision sensor architecture, shown in Fig. 3, embeds a VGA imager with column-level readout (*Amplifiers*) and Analog-to-Digital Conversion (*ADCs*), a processing layer (*Processors*) executing DT-DBS, to generate the *hot pixel* motion bitmap through two-thresholds/pixel (Vmax, Vmin), stored in the 6T *SRAM* and updated at every frame. Residual noise in the motion bitmap is cleaned up by a programmable erosion filter (*Erosion Filter*) applied over a 3x3 pixel kernel to deliver the final bitmap, which is also used to build the two projection vectors (*x-Projection*, *y-Projection*) generating the alert signal to trigger the external processor. Algorithm parameters and other sensor settings are stored into the 16 x 8b registers (*REGISTERS*).

A. Imager

The imager consists of a VGA rolling shutter array of 3T $4\mu m$ pixels. The column-level single-slope ADCs have a pitch of $8\mu m$, therefore the 640 channels have been split into two 320 x 2 channels, placed at the top and bottom sides of the array, serving odd and even columns respectively. Each 320-channels ADC block has its own ramp generator. Under Motion Detection (MD), the always-on DT-BSMD is applied on a 120 x 160 pixels image (QQVGA, obtained subsampling the VGA array. This means that, while the top-side ADC is OFF, only half bottom-side ADC bank works (i.e. 160 channels are ON) to guarantee the motion detection algorithm to be continuously active, while minimizing the power consumption of the sensor. In Imaging Mode (IM), the sensor delivers full resolution VGA image (DATA) by multiplexing the two 8b outputs (GREY_T/GREY_B).

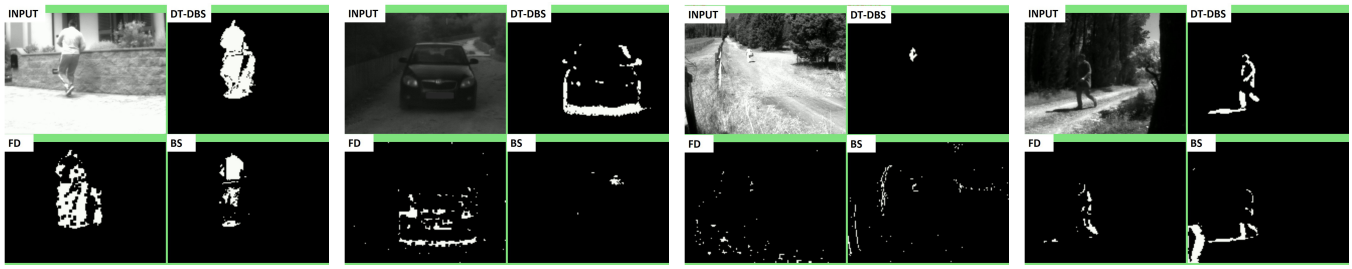


Fig. 2: Comparison among DT-DBS, FD and BS algorithms. The different motion bitmaps have been obtained through simulations of some gray-level videos of outdoor scenarios. (a) on videos characterized by good light conditions and containing well contrasted objects, the three method perform similarly; (b) in case of low-light all the algorithms perform poorly, loosing parts of the objects of interest or returning very noisy motion maps; (c) differently form DT-DBS, on this video containing saturated pixels, FD and BS do not detect the moving object while return noise; (d) BS performs worse than DT-DBS and FD due to a non appropriate background model.

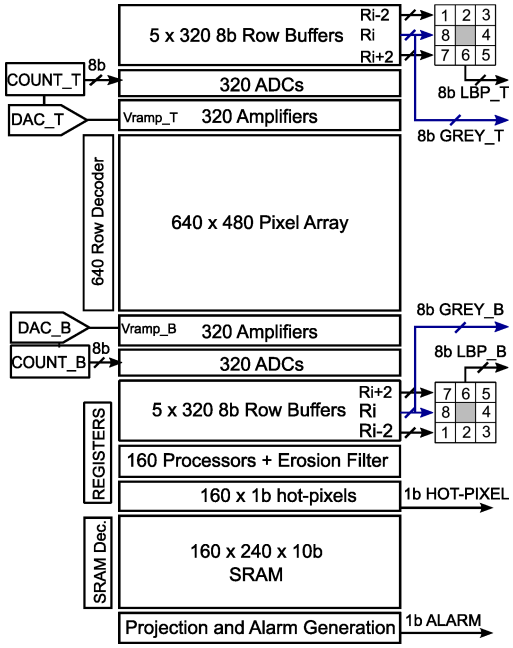


Fig. 3: Block diagram of the VGA vision sensor architecture.

B. Column-Level Amplifier

Fig. 4 shows the column-level single-slope ADC with the pixel readout and its timing diagram. After row selection (Rsel=EN=H), the pixel voltage Vp is stored onto capacitor C1, with Sch=H and S=H, while the output voltage of amplifier is precharged to Vpre. Then C1 is connected to the input of the amplifier (S=L), integrating charge onto C2 with a 2x gain. In a second phase, the pixel is reset and its value Vrst=VDDR is stored onto C1 (S=H), with inverted polarity Sch=L. Then, C1 is connected to the amplifier, subtracting the reset charge from the charge stored on C2. The resulting voltage at the output of the amplifier will be: $Vpre - 2 \times (Vrst - Vp)$.
The pixel readout phase has now concluded and the amplifier is converted into a voltage comparator by opening the feedback loop (PRE=H) and connecting the terminal of C2 to the voltage ramp (Vramp). Since Vramp starts with its high value (Vh), the node A is pulled-up abruptly, unbalancing the comparator, which pulls down Vout. The decreasing voltage ramp, generated by the DAC_B, can now start, pushing node A down toward ground. As soon as node A reaches Vref, the comparator switches and Vout increases toward Vdd sampling the content of the digital counter COUNT_B onto the 8b latches completing the conversion.

C. Column-Level Processor

The processing layer executes the DT-DBS on a sub-sampled image (120 x 160 pixel), as described in [10]. The output is a motion bitmap that is cleaned-up by the programmable erosion filter. The final bitmap is used to build the x-,y-projection vectors that are used to generate the alert signal. The DT-DBS, depicted in Fig. 1 and described by (2), (3) and (4), is implemented in a mixed-mode (analog-digital) by partially exploiting the single-slope ADC operations to compare V_i (4) with the two thresholds and to check if the pixel is inside (CLOSE) or outside (OPEN) the gray-zone and to verify the HP conditions. To complete the DT-DBS, (3) must be executed on the same pixel. This would require either to duplicate the electronics, exploiting the same voltage ramp, or to run twice the ramp executing (2) and (3) sequentially on the same row. However, both solutions imply a large overhead, the first on silicon area, with column-level processor pitch constraints, the second on time and power consumption. To address this issue, we made the assumption that neighbouring pixels have similar behaviours. Therefore, to simplify the implementation, we applied (2) on a pixel $P_{i,j}$ while (3) on $P_{i+1,j}$, so that we avoid to operate the voltage ramp twice per pixel, exploiting instead the regular image readout operation of the ADC, thus reducing power consumption and avoiding additional circuitry. Experimental results, demonstrated that this assumption is more than acceptable. Moreover, the binary output of the two operations, executed on $P_{i,j}$ and $P_{i+1,j}$, are put in OR, turning into the final pixel status ($HP_{i,j}$). The advantage of exploiting the ADC voltage ramp to partially implement the DT-DBS algorithm, is that it simplifies the required electronics: the 8b comparison is made with an identity comparator using 8 XORs in OR; the voltage difference $V_i - V_{Maxi}$ (Eq. 4) is done with a binary counter, which is activated only under OPEN conditions.
For each selected pixel, the processor retrieves one of the two thresholds (V_{Max}/V_{Min}) from the SRAM while, after signals comparison, the threshold is updated with Δ_{OPEN} or Δ_{CLOSE} and restored into the memory.

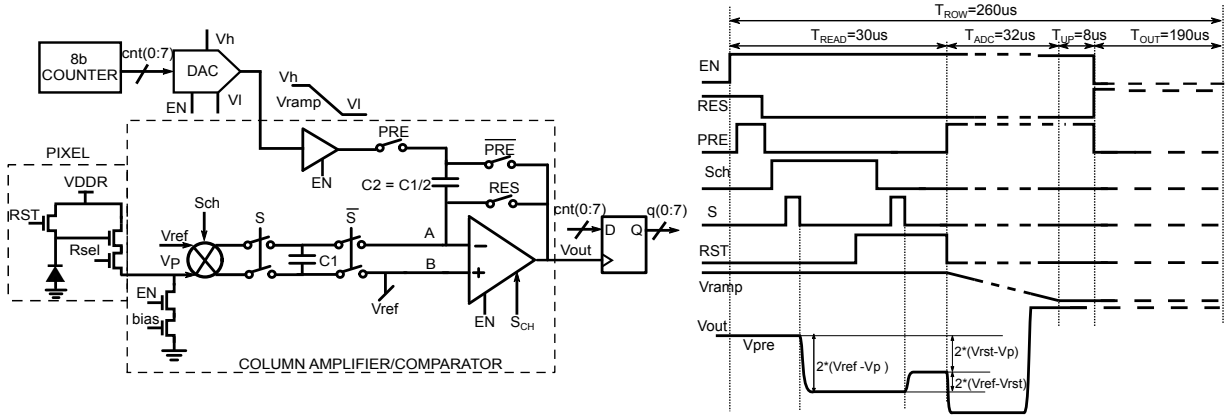


Fig. 4: Schematic of the single-slope ADC where, after the pixel readout, the chopper amplifier is converted into a comparator as part of the ADC. On the right, the timing diagram of the pixel readout (T_{READ}), ADC (T_{ADC}), threshold update (T_{UP}) and data dispatching (T_{OUT}) is shown for a sensor frame rate of 8fps.

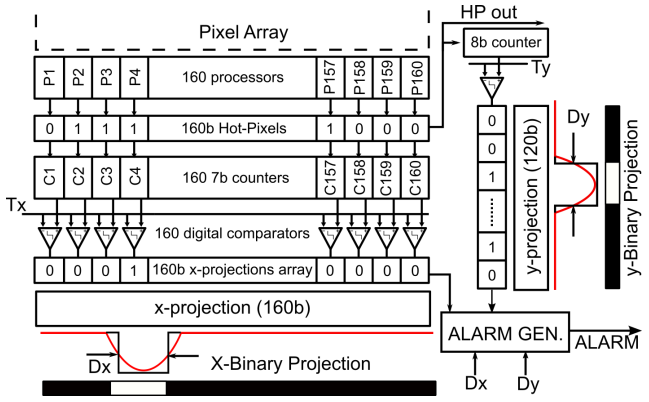


Fig. 5: Schematic of the circuit for alert generation relying on the HP XY projections.

D. Programmable Erosion Filter

The 120 x 160 pixel raw motion bitmap, generated by (2), (3) and (4), is de-noised by a programmable 3x3 pixel kernel erosion filter. Generic filter is applied on pixel $P_{i,j}$ whose binary value is stored in $H_{i,j}$. The pixels of the three consecutive rows and same j-th column $(H_{i-1,j}, H_{i,j}, H_{i+1,j})$ are summed together and the result is summed to those of the *(j-1)-th* and to those of *(j+1)-th* columns, providing a 4b output Q(0:3). The final result is compared with the user-defined threshold NH(0:3), stored in one of the REGISTERS of Fig. 3:

$$if\,(H_{i,j}=1)\wedge\left(\sum_{j-1}^{j+1}\sum_{i}^{i+2}{}_{k}{}_{h}H_{h,k}\right)\geq NH\rightarrow HP_{i,j}=1 \quad (6)$$

The 160 erosion filters provide a motion bitmap to generate the alert signal and to be sent off-chip for further processing.

E. Alert Generation

At each row, the detected HPs contribute to the generation of two x-y motion projection vectors (Fig. 3). These vectors are low-pass filtered and binarized with the two user-defined thresholds D_X and D_Y. ALARM is generated only when the HPs form a region of a certain size and aspect ratio, which is defined by the constraints of (5). The size and the aspect ratio of an object are geometric features that may help to distinguish an object from another and to reject false positives. Of course, the specific values of these features depend on features like the distance of the sensor from the acquired scene and the sensor focal length. In most applications, this information is available and can be employed to set up the variability ranges of the size and the aspect ratio for a set of objects of interest, like cars, humans, boats and so on.

F. Local Binary Patterns

Local Binary Patterns (LBPs) [24] are visual features encoding directional local contrasts over a pre-defined image patch. They capture local micro-structures of the image, like e.g. edges, corners, flat regions, lines, and codify them in binary vectors. LBPs and their distribution of the image are widely employed in many machine vision applications, such as texture analysis [25], face detection [26], and hand gesture recognition [27]. For this reason, despite the LBPs are here not involved in the event detection, we decided to embed their computation of the proposed chip and to deliver them to the processor for further visual tasks. Mathematically, the LBPs are defined as follows. Let x be a pixel and let $y_1,\ldots,y_N$ be N pixels equispaced over a circumference centered at x and with radius R. The LBP code at x is the vector

$$LBP_C(x)=[s(V(y_1)-V(x)),\ldots,s(V(y_N)-V(x))], \quad (7)$$

where V is a frame and s is the function such that $s(t)=1$ if $t\leq 0$, while $s(t)=0$ otherwise. For any $j=1,\ldots,N$, the pixel y_j has coordinates $R(\cos 2\pi/N, \sin 2\pi/N)$: when these coordinates do not fall on integer number, the intensity value of y_j is interpolated by considering its neighbors. By definition, the LBP code is invariant against changes of the illuminant intensity and thus against shadows. Invariance against rotations of $\frac{2\pi h}{N}$ ($h\in\mathbf{Z}$) degrees can be obtained by a circular bitwise cyclic shift of the entries of $LBP_C(x)$. The LBP code is often mapped on a single integer number, that we call *LBP value* and is obtained as follows:

$$LBP(x)=\sum_{j=1}^{N}2^j s(P(j)-P(x)). \quad (8)$$

Some sensors embedding the LBP computation have been recently developed. For instance, the works [28] and [29]

propose two low-power sensors that compute the LBPs on a 3×3 window respectively along the directions $k\pi/2$ and $k\pi/4$, with $k \in \mathbf{Z}$. As for the sensor described in [29], the LBP codes computed by the sensor proposed here differ from the standard ones defined in [30] in the geometry of the pixel neighborhood, which is a square instead of a circle. This choice avoids to interpolate the intensity value of any pixel with not integer coordinates. For any x, the proposed sensor considers a 5×5 window centered at x and computes the code $LBP(x)$ by comparing the intensity of x with the intensities of the pixels displaced as shown in Figure 3. The use of the 5×5 windows enables the exploration of a wider region than that considered in [28] and [29] and it is also compliant with the architectural characteristics of the proposed sensor. Precisely, the LBP codes are computed by the sensor pixel-by-pixel during the read-out phase and then output along with the gray level image by simply multiplexing the 8b output bus of Fig. 3 (DATA), with SEL0. Since odd and even pixels are read out by the BOTTOM and TOP blocks of Fig. 3, computing LBP codes on a 3x3 pixel kernel as in [28] and [29] is complex to be implemented from the layout point of view. Therefore, in the adopted solution pixels of odd/even rows and columns refer to the 8 pixels of the kernel, as depicted in Fig. 6 c). This allows the LBP processing to be decoupled and executed separately by the TOP and BOTTOM blocks. The correct LBP image readout (SEL0=H) is performed by multiplexing the output bus (SEL1=H), in the same way as it is done for the gray-scale image. Fig. 6 a) and b) show an example of grayscale image and related LBP image directly executed by the sensor. In comparison with the standard LBPs, the LBPs output by the sensor have a lower level of invariance against in-plane rotations: $\frac{k\pi}{2}$ degrees versus the $\frac{k\pi}{4}$ degrees of the standard LBPs, for any $k \in \mathbf{Z}$. Apart from this difference, we observed that the LBPs of the sensor and the standard ones perform similarly on the description and matching of textured images. In particular, this performance has been measured on a case study, regarding the illuminant invariant texture retrieval. To this purpose we considered the public dataset Outex [31], consisting of 68 textures, each represented by 20 pictures captured under three lights with color temperature T_1 = 2300 K, T_2 = 2856 K and T_3 = 4000 K. For each pair i, j = 1, 2, 3, $i \neq j$, the images acquired under T_i and those acquired under T_j have been taken respectively as queries and references. Therefore, the queries differ from the references only in the illumination. Each image (query and reference) has been split in its three color components and each component has been described by the distribution of its LBPs values. The three LBP distributions have been then concatenated in a single histogram, that has been used as texture descriptor. For each query Q, the references have been sorted from the most to the least similar to Q. Here the similarity between Q and any reference R was defined in terms of the L^1 difference between their descriptors: the lower this distance, the more similar Q and R are. The accuracy on the matching was measured by the rank $\rho(Q)$, i.e. a parameter related to the position of the reference R corresponding to Q in the sorted list and measured as the ratio $\rho(Q) = \frac{M-P_Q}{M-1}$, where M is the number of references and P_Q indicates the position of R in the ordered list. The closer $\rho(Q)$ to one, the best is the matching accuracy. For both the methods, the mean rank is greater than 0.998. In the 80% of the images the ranks output by the two methods are the same, while in the remaining 20% they differ on average by less than the 1%, meaning that the two LBPs computation have similar performance.

TABLE I: Main chip characteristics

Parameter	**value**	
Pixel array	480 x 640	
Pixel pitch	4 μm	
Fill Factor	49 %	
Column FPN	0.16 %	
DR	53 dB	
SNR	41 dB	
Random Noise	2.6 mVrms	
Frame Rate	8 fps (typ.) up to 15 fps	
Supply voltage	3.3V/1.2V	
Power (8fps)	**MD**	**IM**
Pixel Array	24 μW	151.2 μW
ADC	151.8 μW	660 μW
Digital	168 μW	538.8 μW

IV. Experimental Results and Comparison with similar low-power sensors

Table I shows the main chip characteristics. The electrical characteristics of the sensor have been compared with similar recently published vision sensor chips [1-4] and reported in Table II. It is worth noticing that, the sensor can simultaneously deliver full resolution VGA grayscale, LBP coding and QQVGA motion bitmap. On the other side, except [1], the Figure Of Merit (FOM) of this work is not competitive neither in Motion ($2.24nW/pixel*fps$) nor in Imaging Modes ($549pW/pixel*fps$) compared with the other sensors. In the last two rows of Table II, the values of the power consumption are reported for each chip, both in Imaging and Motion Modes. The normalization was done referring each sensor to the same amount of pixels of this work (QQVGA for MD and VGA for IM) and to the same frame rate of 8fps.

Table III shows the required energy needed by the sensor to execute the DT-DBS algorithm on-chip with erosion filter and alert detection through x-y projections, compared with the energy required to executed the same processing off-chip with an external processor [32]. Due to the mixed-mode implementation of the algorithm, it was not possible to estimate the average power consumption of the processing layer. Therefore, the total digital power of the sensor ($168\mu W$) was used although this value is overestimated since it includes the LBP computation, which cannot be disabled.

Moreover, we tried to compare the performance of event detection of the sensor against those sensors based on FD. For this purpose, we chosen a scenario in which a boat is approaching the coast in a sunny and windy day. This scenario is very critical since it sets severe requirements to the event-detection. Fig. 7 reports a snapshot, extracted from a 8fps 500 frames video, acquired with the sensor setup of Fig. 9, showing: a) the VGA grayscale image; b) the related QQVGA motion bitmap generated by the sensor; c) a hypothetical motion bitmap generated simulating the FD algorithm applied

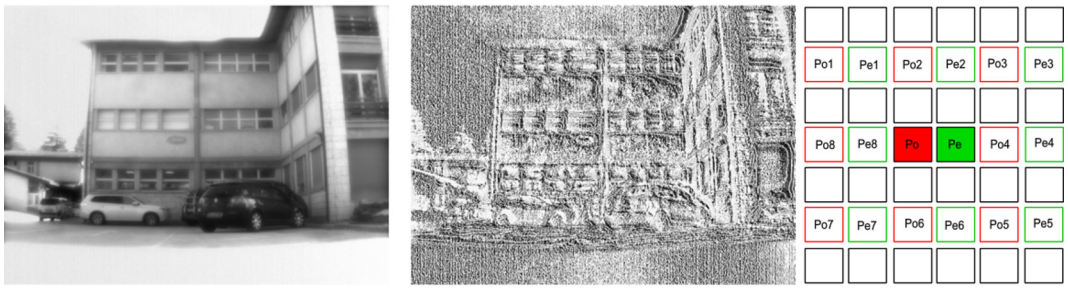


Fig. 6: a) Gray-scale image captured by the fabricated sensor with b) its LBP coding delivered by the sensor. c) 5x5 pixel kernel used to compute LBPs. Pixel *Pe* refers to the neighbouring pixels *Pe1-Pe8* and is computed with the Bottom-Side block, while *Po* refers to *Po1-Po8* and is executed by the Top-Side block of the VGA array.

TABLE II: Comparison of the electrical performance of this work with those of similar low-power vision sensors

Parameter	**this work**	**Kumagai** [1]	**Choo** [2]	**Choi** [3]	**Kim** [4]	**Zhong** [5]
Technology	$0.11\mu m$ 1P4M	$90nm$ 1P4M	$65nm$ 1P3M	$0.11\mu m$ 1P4M	$0.11\mu m$ 1P4M	$0.18\mu m$ 1P6M
Pixel array	640 x 480	1536 x 2560	792 x 528	256 x 256	128 x 128	256 x 216
Supply Voltage	3.3V/1.2V	1.8V/1.0V	-	1.3V/0.8V	1.2V/0.6V	1.2V/0.8V
Motion resolution	160 x 120	16 x 5	32 x 20	128 x 128	48 x 16	128 x108
Motion detect.	DT-DBS	FD + BS	FD	FD	FD	BS + FD
Alert gener.	xy-projections	pixel-count	pixel count	pixel count	pixel count	pixel count
Features extr.	Temp. change xy-proj, LBP	Temp. change	Temp. change	Temp. change HOG	Temp. change	Temp. change Obj. Detection
Pow. cons.(Imaging)	$1.35mW@8fps$	$90mW@60fps$	$392\mu W@5.6fps$	$51\mu W@15fps$	$29\mu W@19fps$	$21.14\mu W@15fps$
Pow. cons (Motion)	$344\mu W@8fps$	$1.1mW@10fps$	$9.5\mu W@5.6fps$	$3.31\mu W@15fps$	$1.1\mu W@30fps$	$2.36\mu W@15fps$
FOM [Power/pixel*fps] (Motion Detection and Imaging Mode)						
FOM_{MD}	$2.24nW$	$1.37\mu W$	$2.65nW$	$13.5pW$	$47.7pW$	$11.4pW$
FOM_{IM}	$549pW$	$403pW$	$167.5pW$	$51.94pW$	$152pW$	$25.5pW$
Normalized power consumption - QQVGA (MD), VGA (IM), 8fps						
Motion Detection	$344\mu W$	$211mW$	$407\mu W$	$2.07\mu W$	$7.33\mu W$	$1.75\mu W$
Imaging Mode	$1.35mW$	$990\mu W$	$412\mu W$	$128\mu W$	$229\mu W$	$62.7\mu W$

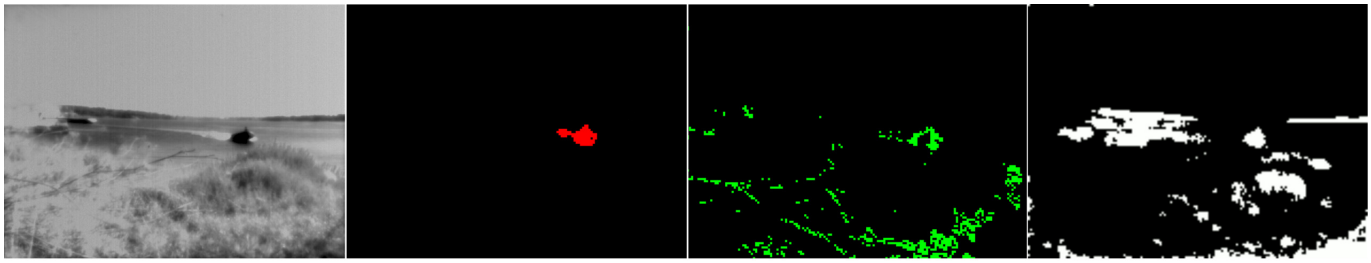
TABLE III: Energy to compute the DT-DBS algorithm on-chip, as this work does with no need of the external processor, and off-chip implementation using the external processor [32].

Paramenter	**Value**	
Sensor Power Consumption for Frame Capture [1]	176 μW	
Image Size (Bytes)	120 x 160	
Frame Rate	8 fps	
Sensor Energy for Frame Capture [2]	22 μJ	
Ext. Processor Power (Sleep) (0.8V, 0.32MHz)	30 μW	
Ext. Processor Power (Active) (1.0V, 150MHz)	27 mW	
DT-DBS (QQVGA) + Alert Generation	**on-chip**	**off-chip**
Frame Transfer Time (50MHz)	-	384 μs
Energy for Frame Transfer	-	10.4 μJ
Frame Processing Time (1.0V, 150MHz)	-	4.1ms
Sensor Avg. Power Consumption for Processing	168 μW	-
Processing Energy per Frame	21 μJ	111 μJ
Processing Energy + Sleep Energy per Frame	-	125 μJ
Total Energy [3]	43 μJ	147 μJ

[1] The sensor average power consumption in MD is $344\mu W$ at 8fps. The estimated quote for image capture and A/D conversion is $176\mu W$, while that one for embedded image processing is $168\mu W$;
[2] Energy per frame capture: $176\mu W$/ 8 fps = $22\mu J$;
[3] Energy per frame capture + frame transfer (off-chip only) + DT-DBS image processing and alert generation.

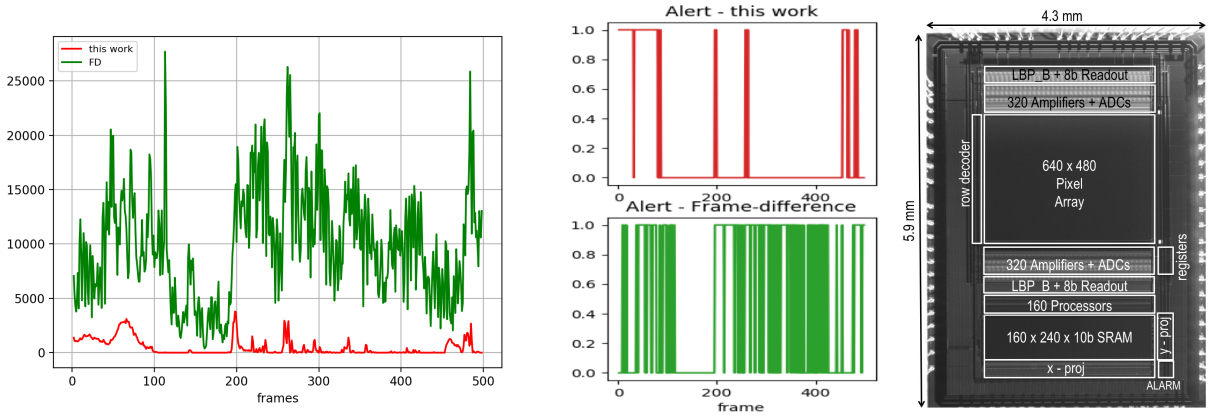
on a sub-sampled format (120 x 160 pixels) of a) and de-noised with the same 3x3 pixel erosion filter as that one used in b). While the bitmap, delivered by the sensor and shown in b) with red pixels, is very clean although the noisy scenario, with waves and swaying vegetation, the motion bitmap in c) with green pixels looks quite noisy so that the boat cannot be even distinguished from the moving background. Plotting the number of HPs generated by the sensor (red) and those of the FD counterpart (green), as in Fig. 8 a), it is visible that the HP activity of FD is about one order of magnitude larger and most of it is due to the noise. According to the video ground truth, the moving boat is inside the scene until frame 90, generating a continuous alert. From 90 to frame 500, the boat is outside the scene and no event is present. The graphs of Fig. 8 b) show the alert signals generated by the two approaches. Starting from frame 90, while the sensor generates 5% of false alerts, the FD counterpart has 46.5% of false positives, thus turning ON the external processor more frequently to execute image processing tasks, with a larger waste of power.

Table V reports an estimation of the required energy per frame vision systems, based on the sensor listed in II, with the best FOM in MD and IM operating modes, interfaced with the processor [32] to execute a people counting task through CNN [33] on alert. As expected, the energy under alert is dominated by the processor. The total estimated energy of the system takes into account the rate of false alerts generated by the two sensors in the case of the video of Fig. 7, from frame 91 to 500, where no alert should be generated. It is possible to see how, in this outdoor scenario, the proposed approach has about 8 times better energy efficiency against the FD counterpart.



a) b) c) d)

Fig. 7: a) VGA grayscale image captured with the fabricated chip; b) motion bitmap delivered by the proposed vision sensor in a real outdoor scenario; c) motion bitmap generated simulating FD algorithm applied on the video in a) and scaled to QQVGA resolution in order to be normalized with this sensor. d) Hot-pixel bitmap generated by traditional Background Subtraction (BS). The number of hot-pixels generated by FD and BS is much larger than that of DT-DBS.



a) b) c)

Fig. 8: Hot-pixel activity of this sensor compared to the FD counterpart. The comparison refers to the video of Fig. 7. From frame 90 to 500 the boat is outside the scene and no events should be detected; b) activity of the alert signal generated by the two approaches; c) microphotograph of the VGA sensor measuring 25.4 mm^2.

a) b)

Fig. 9: Sensor prototype: a) the vision sensor is driven by a FPGA and interfaced with a PC through which it is possible to change the algorithm parameters and grab grayscale videos with related motion bitmap, projection vectors and alert; b) test setup used in outdoor scenario.

TABLE IV: Estimated processor [32] energy per frame needed to execute people counting task on QVGA grayscale images at 4fps.

Paramenter	Value
Frame rate	4 fps (T = 250ms)
Grayscale Image (QVGA)	320 x 240
Proc. Power - Sleep - P_{SL}	30 μW
Proc. Power - FC - P_{FC}	27 mW
Proc. Power - 8 Cores - P_{8C}	37 mW
Img. Trans. Time (50MHz) - T_{IT}	1.54 ms
Img. Trans. Energy ($E_{IT} = P_{FC} * T_{IT}$)	**41.5** μJ
Img. Proc. Time (150MHz) - T_{IP}	167 ms
Img. Proc. Energy ($E_{IP} = P_{8C} * T_{IP}$)	**6.18** mJ

Referring to the estimated energy per frame, reported in Table V, (9) defines the figure of merit quantifying the energy efficiency of an event-based vision system. The proposed FOM takes into account the sensor's energy per frame in the two operating modes (MD, IM) as well as the energy per frame associated to the external processor (Sleep and Active Modes) to execute a desired vision task on a certain image size at a given frame rate:

$$FOM = \overbrace{[SE_{MD} + P_{SL} * T]}^{(1)} * (1 - R_A) + \overbrace{[SE_{IM} + (P_{FC} + P_{IM}) * (T_P) + P_{SL} * (T - T_P)}^{(2)} * R_A \quad (9)$$

with $T_P = T_{IT} + T_{IP}$ (Table. IV).
The rate of alert (R_A) might be estimated simulating the chip-embedded algorithm through video datasets of the use case of interest.

TABLE V: Estimated energy per frame of vision systems working at 4fps and based on the following sensors interfaced with [32] and referring to the outdoor scenario of Fig. 7 (from frame 90 to 500).

Parameter	this work	Kumagai [1]	Choo [2]	Choi [3]	Kim [4]	Zhong [5]
Sensor Energy (MD) [1] - SE_{MD}	43 μJ	26.4 mJ	50.9 μJ	259 nJ	917 nJ	219 nJ
Sensor Energy (IM) [2] - SE_{IM}	42.2 μJ	29.3 μJ	12.9 μJ	3.98 μJ	7.15 μJ	1.96 μJ
Vision Syst. Energy (MD) [3] - VSE_{MD}	50.5 μJ	26.4 mJ	58.4 μJ	7.76 μJ	8.42 μJ	7.72 μJ
Vision Syst. Energy (IM) [4] - VSE_{IM}	6.27 mJ	6.28 mJ	6.25 mJ	6.23 mJ	6.24 mJ	6.23 mJ
Rate of Alert - R_A	5%	46%	46%	46%	46%	46%
Total Syst. Energy [5]	**361** μJ	**17.1** mJ	**2.91** mJ	**2.87** mJ	**2.87** mJ	**2.87** mJ

[1] ($SE_{MD} = FOM_{MD} * QQVGA * 4fps$)
[2] ($SE_{IM} = FOM_{IM} * QVGA * 4fps$)
[3] VSE_{MD}: term (1) of (9);
[4] VSE_{IM}: term (2) of (9);
[5] System energy estimated with (9) and accounting for False Positives: Img. Capture+Img. Transfer+Img. Processing.

V. Conclusion

A low-power VGA vision sensor has been presented embedding a custom DT-DBS technique combined with motion bitmap projections to generate an alert in case of a moving target in the scene. The on-chip algorithm exhibits a high reliability in noisy outdoor scenarios, minimizing the rate of false positives, which is one of the most critical parameters in event-based systems. While event-based CMOS vision sensors are typically compared only from the electrical point of view, we demonstrated that performance evaluation should be undertaken in an holistic way, involving sensor, external processor and algorithm performance.
In this regard, in Tab. V we compared the estimated energy efficiency of this work with that one of [5] by adopting the proposed FOM (9). Although [5] has lower sensor's FOM, both in MD and IM, the proposed sensor exhibits about 7 times better energy performance in a noisy outdoor scenario.

Acknowledgment

The authors would like to thank Nicola Furlan for his support in the system design and to Manuele Rusci for valuable discussions. This work was funded by the EU H2020-FCT-2014 FORENSOR Project, under Grant n. 653355 (https://cordis.europa.eu/project/rcn/194854/brief/en).

References

[1] O. Kumagai and al., "A 1/4-inch 3.9Mpixel low-power event-driven backilluminated stacked CMOS image sensor," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2019 IEEE International*, Feb. 2018, pp. 86–88.

[2] K. D. Choo, L. Xu, Y. Kim, J. Seol, X. Wu, D. Sylvester, and D. Blaauw, "Energy-efficient low-noise CMOS image sensor with capacitor array assisted charge-injection SAR ADC for motion-triggered low-power IoT applications," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2019 IEEE International*, Feb. 2019, pp. 96–98.

[3] J. Choi, S. Park, J. Cho, and E. Yoon, "A 3.4 uW CMOS Image Sensor with Embedded Feature-Extraction Algorithm for Motion-Triggered Object-Of-Interest imaging," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International*, Feb. 2013, pp. 478–479.

[4] G. Kim, M. Barangi, Z. Foo, N. Pinckney, S. Bang, D. Blaauw, and D. Sylvester, "A 467nW CMOS Visual Motion Sensor with Temporal Averaging and Pixel Aggregation," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International*, Feb. 2013, pp. 480–481.

[5] X. Zhong, M. Law, C. Tsui, and A. Bermak, "A Fully Dynamic Multi-Mode CMOS Vision Sensor With Mixed-Signal Cooperative Motion Sensing and Object Segmentation for Adaptive Edge Computing," *IEEE Journal of Solid State Circuits*, vol. 2, no. 1, pp. 1–14, Jan 2020.

[6] M.-T. Chung and C.-C. Hsieh, "A 0.5v 4.95μw 11.8fps PWM CMOS Imager with 82dB Dynamic Range and 0.055% Fixed-Pattern Noise," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2012, pp. 114–115.

[7] M. Gottardi, N. Massari, and S. A. Jawed, "A 100μW 128x64 Pixels Contrast-Based Asynchronous Binary Sensor for Sensor Network Applications," *IEEE Journal of Solid State Circuits*, vol. 44, no. 5, pp. 1582–1592, May 2009.

[8] S. Hanson and D. Sylvester, "A 0.45-07 V Sub-Microwatt CMOS Image Sensor for Ultra-Low Power Applications," in *Proc. IEEE Symposium on VLSI Circuits 2009*, Kyoto, June 2009, pp. 176–177.

[9] T.-H. Hsu and et. al, "A 0.8V Multimode Vision Sensor for Motion and Saliency Detection with Ping-Pong PWM Pixel," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2020 IEEE International*, Feb. 17 2020, pp. 110–111.

[10] Y. Zou, M. Gottardi, M. Lecca, and M. Perenzoni, "A Low-Power VGA Vision Sensor with Event Detection through Motion Computation based on Pixel-Wise Double-Threshold Background Subtraction and Local Binary Pattern Coding," in *Proc. IEEE ESSCIRC 2019*, Sept. 2019.

[11] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Review and evaluation of commonly-implemented background subtraction algorithms," in *2008 19th International Conference on Pattern Recognition*. IEEE, 2008, pp. 1–4.

[12] A. Mittal and N. Paragios, "Motion-based background subtraction using adaptive kernel density estimation," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2. Ieee, 2004, pp. II–II.

[13] S. Brutzer, B. Höferlin, and G. Heidemann, "Evaluation of background subtraction techniques for video surveillance," in *CVPR 2011*. IEEE, 2011, pp. 1937–1944.

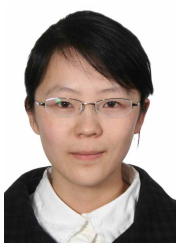
[14] M. Piccardi, "Background subtraction techniques: a review," in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, vol. 4. IEEE, 2004, pp. 3099–3104.

[15] I. Haritaoglu, D. Harwood, and L. S. Davis, "W/sup 4/: Who? When? Where? What? A real time system for detecting and tracking people," in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, April 1998, pp. 222–227.

[16] I. Haritaoglu and D. Harwood and L. S. Davis, "Wsup 4: real-time surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 809–830, Aug 2000.

[17] W. Shuigen, C. Zhen, and D. Hua, "Motion detection based

on temporal difference method and optical flow field," in *2009 Second International Symposium on Electronic Commerce and Security*, vol. 2, May 2009, pp. 85–88.
[18] H. Wei and Q. Peng, "A block-wise frame difference method for real-time video motion detection," *International Journal of Advanced Robotic Systems*, vol. 15, no. 4, p. 1729881418783633, 2018. [Online]. Available: https://doi.org/10.1177/1729881418783633
[19] N. M. Oliver, B. Rosario, and A. P. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 831–843, 2000.
[20] M. Heikkila and M. Pietikainen, "A Texture-Based Method for Modeling the Background and Detecting Moving Objects," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 4, pp. 657–662, April 2006.
[21] S.-G. Wei, L. Yang, Z. Chen, and Z. feng Liu, "Motion detection based on optical flow and self-adaptive threshold segmentation," *Procedia Engineering*, vol. 15, pp. 3471 – 3476, 2011, cEIS 2011.
[22] Y. Li, Z.-x. Sun, B. Yuan, and Y. Zhang, "An improved method for motion detection by frame difference and background subtraction," *Journal of Image and Graphics*, vol. 14, no. 6, pp. 1162–1168, 2009.
[23] Z. Zivkovic and F. Van Der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern recognition letters*, vol. 27, no. 7, pp. 773–780, 2006.
[24] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.
[25] L. Nanni, A. Lumini, and S. Brahnam, "Survey on LBP Based Texture Descriptors for Image Classification," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 3634–3641, Feb. 2012.
[26] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006.
[27] Y. Ding, H. Pang, X. Wu, and J. Lan, "Recognition of hand-gestures using improved local binary pattern," in *2011 International Conference on Multimedia Technology*. IEEE, 2011, pp. 3171–3174.
[28] M. Gottardi and M. Lecca, "A 64×64 pixel vision sensor for local binary pattern computation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 5, pp. 1831–1839, 2018.
[29] X. Zhong, Q. Yu, A. Bermak, C.-Y. Tsui, and M.-K. Law, "A 2pj/pixel/direction mimo processing based cmos image sensor for omnidirectional local binary pattern extraction and edge detection," in *2018 IEEE Symposium on VLSI Circuits*. IEEE, 2018, pp. 247–248.
[30] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.
[31] T. Ojala, T. Maenpaa, M. Pietikainen, J. Viertola, J. Kyllonen, and S. Huovinen, "Outex-new framework for empirical evaluation of texture analysis algorithms," in *Object recognition supported by user interaction for service robots*, vol. 1. IEEE, 2002, pp. 701–706.
[32] E. Flamand and D. Rossi and F. Conti and I. Loi and A. Pullini and F. Rotenberg and L. Benini, "GAP-8: A RISC-V SoC for AI at the Edge of the IoT," in *Procs. of IEEE 29th Int. Conf. on Application-Specific Systems, Architectures and Processors (ASAP)*, July 2018.
[33] S. Benninger, M. Magno, A. Gomez, and L. Benini, "EdgeEye: A Long-Range Energy-Efficient Vision Node For Long-Term Edge Computing," in *Proc. 2019 Tenth Int. Green and Sustainable Computing Conference (IGSC)*, Oct. 2019, pp. 1–8.



Yu Zou received the bachelor's degree in 2010 from Tongji University, Shanghai, China and Politecnico di Milano, Milano, I. She received the M.S. degree and Ph.D degree from Politecnico di Milano, Milano, Italy, in 2012 and 2016 respectively. Since 2016, she joined Fondazione Bruno Kessler, Trento, Italy, as a research scientist working in Integrated Radiation and Image Sensors (IRIS) group of Materials and Microsystems (CMM) division. Her research interests are in the fields of low-power algorithm on-chip CMOS image sensors and single-photon avalanche diode (SPAD) based image sensors.

Massimo Gottardi is senior researcher at the Centre for Materials and Microsystems of the Fondazione Bruno Kessler, Trento, Italy. His current research interests include energy-efficient CMOS vision sensors, hardware-oriented algorithms for sensor-embedded image processing, memristor-based architectures for adaptive sensory systems and IoT. He served as reviewer for several international journals and is co-author of more than 140 papers in refereed journals and international conferences.

Michela Lecca received the Ms. Degree in Mathematics from University of Trento (I) in 2002. In the same year she joined the Research Unit Technologies of Vision, Fondazione Bruno Kessler (I), where she is currently working as a Permanent Researcher. Her research interests include all the aspects of low-level image processing, unsupervised object recognition and image retrieval, machine color constancy, and their implementation on smart embedded systems. Michela authored several research papers published in international journals and conference proceedings. She has taken part in the program committee of many editions of Applied Computing Machine Symposia, and carries on a reviewing activity for several international conferences and journals. She is a member of the International Association IAPR - CVPL.

Matteo Perenzoni graduated in Electronics Engineering from the University of Padova, Italy, in 2002. In 2004, he joined FBK, Trento, Italy, as a researcher working in the Integrated Radiation and Image Sensors (IRIS) research unit. Meanwhile, he also taught courses on electronics and sensors at the University of Trento for master and doctorate school, from 2006 to 2010. During 2014, he has been visiting research scientist in the Microelectronics department of TU Delft, Netherlands. Currently, he is leading the IRIS research unit in FBK, working in the field of radiation and image sensor using custom and CMOS technologies. He currently serves as member of the technical program committee of the European Solid-State Circuit Conference (ESSCIRC) since 2015 and of the International Solid-State Circuit Conference (ISSCC) since 2018. His research interests include advanced CMOS image sensors with a focus on single-photon detection, THz image sensors and optimization of analog integrated circuits.