Retrieving Gray-Level Information from a Binary Sensor and its Application to Gesture Detection

Orazio Gallo[1] Iuri Frosio[1] Leonardo Gasparini[2] Kari Pulli[1,3] Massimo Gottardi[2]
[1]NVIDIA, [2]Center for Materials & Microsystems, Fondazione Bruno Kessler, [3]Light

Abstract

We report on the use of a CMOS Contrast-based Binary Vision Sensor (CBVS), with embedded contrast extraction, for gesture detection applications. The first advantage of using this sensor over commercial imagers is a dynamic range of 120dB, made possible by a pixel design that effectively performs auto-exposure control. Another benefit is that, by only delivering the pixels detecting a contrast, the sensor requires a very limited bandwidth.

We leverage the sensor's fast $150\mu s$ *readout speed, to perform multiple reads during a single exposure; this allows us to estimate gray-level information from the otherwise binary pixels. As a use case for this novel readout strategy, we selected in-car gesture detection, for which we carried out preliminary tests showing encouraging results.*

1. Introduction

Several vision-based applications share similar low-level image processing stages; rather than allocating time and power for an external processor to perform them, one can design a sensor that embeds some of these transformations. Contrast, for instance, represents one of the most useful low-level features in computer vision for detection and recognition tasks. In addition, high-dynamic-range (HDR) capabilities are essential for virtually any application that needs to be deployed in outdoors scenarios: strong, direct illumination and hard shadows may affect the robustness of gesture detection algorithms. Combining these requirements in a single imager is challenging.

Commonly, HDR capabilities are achieved by capturing multiple, differently exposed pictures, which takes time and computing resources: it would be beneficial to the framerate of the system to perform HDR capture directly at the sensor level, as well as to compress data; this can minimize latency, the amount of data transferred, and the off-chip processing power required.

There exists a number of sensors that can perform some form of visual processing on chip [1, 2], or extract salient features from the scene [3, 4, 5, 6].

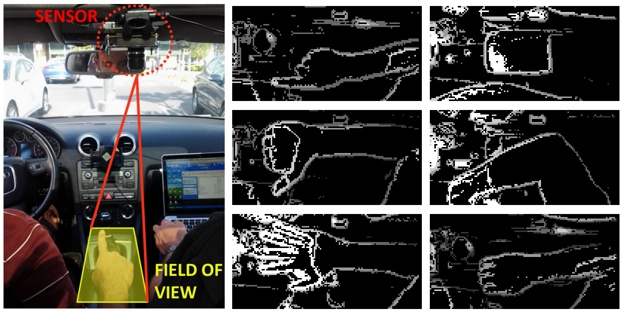

Figure 1. We modify an HDR, binary contrast sensor to enable the extraction of gray-level information. We then use this information to detect gestures in a car's cockpit. The setup for the proposed system is shown in the left column. The center column shows three typical *gestures* (see Sec. 4). The rightmost column shows *backgrounds*, where the hand is either making a fist or is absent altogether.

Several implementations of vision sensors have been proposed, that are targeted to contrast extraction. Some rely on temporal contrast, implemented through frame differencing [6, 5, 7], or frame-free light intensity change [3]; others focus on spatial contrast, either using some flavor of pixel-level connectivity [8, 9], or by performing spatial-temporal operations [1, 10]. In our work, we focus on spatial contrast, an essential cue in the context of image and video analysis.

We propose to use the contrast-based binary vision sensor (CBVS) originally proposed by Gottardi *et al*. [9]. This is a 128×64-pixel vision sensor that extracts and binarizes the spatial contrast of the acquired image *directly in the pixel* through a charge-sharing technique, and with no DC power consumption (see Sec. 2). The CBVS can achieve a dynamic range of over 100dB thanks to a pixel-level autoexposure control. It then only delivers the addresses of the active pixels in bursts of data at a rate of $\sim$ 80MB/s, thus minimizing the power it requires.

Our contribution lies in (a) the extension of the CBVS to output gray-level information from the binary pixels' values, and (b) a system for in-car gesture detection. We implement and test the system on challenging images representative of real-case scenarios (Fig. 1) and report preliminary results. Note that the extension of the sensor's capabilities does not require modifications to the original sensor's hardware: a firmware upgrade suffices.

2. Contrast-based Vision Sensor

The CBVS embeds a pixel-level processing that estimates the spatial contrast across triplets of pixels, and binarizes it before delivering it off chip [9]. The basic operating principle is shown in Fig. 2. The sensor is based on kernels of three photodiodes; let us assume that, for a given scene, PE is the darkest pixel and PO the brightest one. After reset, all the photodiodes start discharging from the reset voltage V_R, at a rate proportional to the impinging irradiance. Moreover, it can be shown[1] that the maximum voltage difference among the three photodiodes, V_C, is linearly proportional to the Weber contrast:

$$V_C = V_{PE}(t_1) - V_{PO}(t_1) = V_R\frac{I_{PO} - I_{PE}}{I_{PO}}, \qquad (1)$$

where I_{PE} ans I_{PO} are the photo-generated currents of pixels PE and PO respectively, and t_1 is the time at which the most illuminated pixel reaches a given voltage threshold. By definition, the contrast is insensitive to the absolute irradiance value, and so is the binary value obtained by applying a global threshold, V_T, to the contrast V_C of each pixel in the sensor. This is roughly equivalent to performing auto-exposure at the pixel-level, thus enabling the sensor's HDR capabilities.

Figure 3 shows a block diagram of the pixels' kernel. The analog contrast voltage V_C is the output of the CONTRAST BLOCK, while V_{curr} is the binarized value in the current frame. A 1-BIT MEM is also embedded in the pixel, which allows storing the past binary value of the contrast, and executing frame difference at the column-level (BIT LINES).

Another interesting feature of the sensor is the output data encoding. The chip only delivers the addresses of the active pixels, discarding the others. This is an efficient way to compress sparse data; indeed, in a typical scenario, the number of active pixels is smaller than 25% of the total pixel count.

3. Gray-Level Operating Mode

Leveraging the fast readout speed allowed by the sensor's architecture, we propose a strategy to extract gray-level information from the otherwise binary sensor.

To describe the proposed technique, it is worth returning to the pixel's operating principle. Note that, in the interest of clarity, in Sec. 2 we made the simplifying assumption that the voltage V_C is read when one of the pixels is completely discharged; however, the sensor's architecture allows for this reading to be performed when the most illuminated pixel reaches a certain threshold V_{TH}. Figure 4 shows the voltages of two pixel kernels that are exposed to different irradiance levels during the exposure of the same frame—PN, the third pixel of each kernel, is omitted for clarity. Pixel PO_1 sees the most incoming light, while PE_2 sees the least. In this case, only PO_1 and PO_2 can detect a contrast, being the ones exposed to the strongest irradiance for each kernel; however, PO_1 reaches V_{TH} at time t_1, soon after the start of the integration time, while PO_2 takes a longer time t_2, as shown in Fig. 4. In the mode of operation described by Gottardi *et al.* [9], the sensor's data is read at the end of the frame (*i.e.*, at the end of the exposure time); since the voltage difference between the brightest and

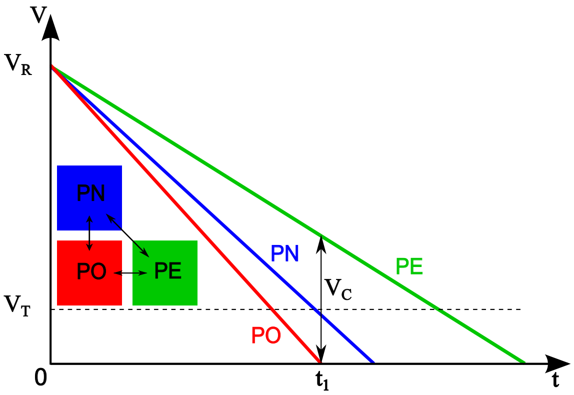

Figure 2. Basic operating principle for the pixel-level contrast estimation. Each pixel's voltage is compared against that of the two neighboring pixels PN and PE. When the pixel exposed to the highest irradiance—PO in this example—reaches a threshold voltage, here set to 0 for simplicity, it is compared with the least illuminated pixel in the kernel—PE in this example. The voltage difference V_C is proportional to the contrast between the two pixels. If V_C is larger than a threshold V_T, the output of the triplet of pixels will be 1.

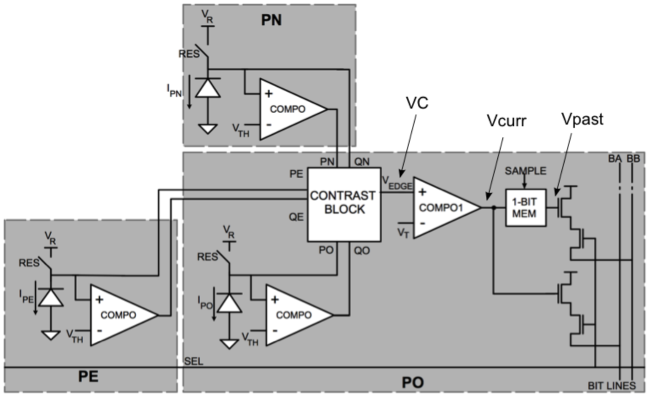

Figure 3. Block diagram of the pixel PO together with the two neighbors (PN and PE).

[1] This can be proved by looking at the similarity between the triangles in Fig. 2.

darkest pixel of the kernel is binarized *before* it is read, the information that PO$_1$ is brighter than PO$_2$, as reflected by the fact that the former becomes active much sooner than the latter, is lost. However, we observe that the sensor readout takes only $T_R = 150\mu$s regardless of the amount of active pixels in the image. Moreover, within the exposure time of a frame, pixels that are below threshold can continue to integrate light, even after the active pixels are read out. Therefore, it is possible to read out the sensor's active pixels multiple times during the same exposure.

In the example in Fig. 4, the sensor is read out $N = 4$ times, at $T_{\{1,2,3,4\}}$; we can then extract a gray-level image by assigning to pixels that are read out at time T_i the gray level $L = N - i + 1$ (recall that bright pixels become active sooner).

Note that the power consumption associated with reading out the sensor is proportional to the number of active pixels, since only their address is extracted from the sensor—their value is, after all, binary. It could therefore seem that performing multiple readouts would require significantly more power. However, in the proposed implementation, active pixels are delivered only once during the readout process. For instance, referring to Fig. 4, pixel PO$_1$, which detects a contrast at t_1, is delivered only in the readout phase T$_1$.

We can achieve this thanks to the 1-BIT MEM that equips each pixel, as shown in Fig. 3. Specifically, at the beginning of the exposure time, this memory is reset; when the output of the pixel, V_C, goes above threshold, the memory is set to 1 and stays at 1 until the end of the exposure. At each readout T$_i$, the address of the pixel is delivered only if its binarized value, V_{curr}, is different from the state stored in 1-BIT MEM. This avoids multiple transfers of the same pixel, leaving the total amount of transferred data as with the original sensor's readout strategy [9]. We implemented the proposed readout strategy on the FPGA board on the back of the sensor shown in Fig. 5.

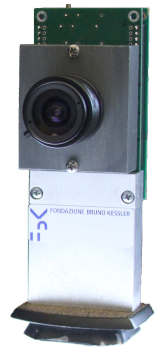
Figure 5. Our prototype.

The FPGA board we used is an Opal Kelly XEM3001v2, equipped with a Xilinx Spartan-3 XC3S400-4.The firmware temporarily stores the readout index and addresses of the rows and columns of the active pixels into the FPGA RAM before transferring the data to a host PC via USB. Note that, since a pixel can become active only once within the exposure time, the maximum size of the frame is the sensor resolution. The current implementation requires 8 16-Kbit blocks of RAM to fully store a frame, corresponding to 50% of the total RAM blocks available. The remaining logic occupies 944 (26%) slices and 54 (37.5%) I/O blocks, of which 27 are dedicated to the sensor. The finite state machine within the FPGA operates at 100 MHz, while the sensor output data transfer rate is 80 MHz (limited by the sensor itself). Such a configuration achieves a frame rate of 64 fps, limited by the USB poll rate. Higher rates could be obtained by buffering and transferring multiple frames at a time, at the cost of a larger memory requirement.

3.1. Power analysis

Although the sensor's power consumption is not central to our application, especially for such a low resolution, things can change drastically for imagers with a larger pixel count, such as VGA or WVGA formats. Two main components contribute to the CBVS's power consumption: the pixel array scanning and the data delivering. Figure 6 shows the power consumption of the sensor (single gray level mode) against the frame rate, when 25% of the pixels are active. In Active Mode the sensor delivers the contrast image; Idle Mode is a low-power mode in which the array is scanned and the total number of pixels in the array is counted, but no is data delivered. The total count can be read out only upon request. Therefore, the difference between the two curves reflects only the power consumption of the sensor during data delivery, which is the most power-hungry activity of the sensor. To avoid a dramatic increase of power consumption, it is important to deliver the same amount of data as for the standard readout strategy: this is achieved by exploiting the 1-BIT MEM as explained in Sec. 3.

The total power consumption of the sensor can be expressed as:

$$P_N = N \cdot P_{\text{scan}} + P_{\text{deliver}}, \tag{2}$$

where, once again, N is the number of readout scans, *i.e.*, the number of non-zero gray levels. At a typical frame rate, the maximum number of gray levels is $N_{\text{max}} = 220$ (see Sec. 3.2). For a common scenario with 25% active pixels, delivering the data costs about $40\mu W$. The worst case in terms of power consumption is then:

$$P_{220} = 220 \cdot 20\mu W + 40\mu W = 4.44mW. \tag{3}$$

In the case of 8 readout scans, the same used for the test (Fig. 7), the total estimated power consumption is:

$$P_8 = 8 \cdot 20\mu W + 40\mu W = 200\mu W, \tag{4}$$

which is about 3 times higher than the power burnt by the sensor in the standard operating mode, with the same frame rate. We believe that this is an acceptable price to pay, considering the additional information extracted.

3.2. Results

In this section we present images produced with the proposed readout strategy, on simulated and real data.

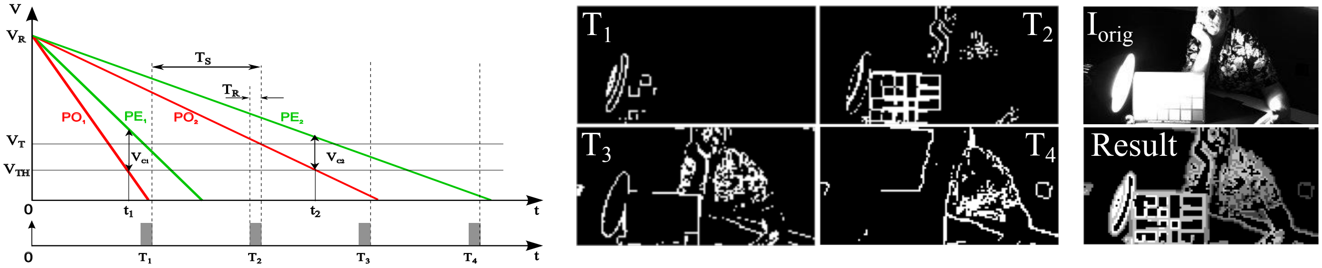

Figure 4. Illustration of the proposed technique. The graph shows two kernels of pixels (the third pixel for each kernel is omitted for clarity). In both kernels PO becomes active during exposure time, since V_C is larger than the threshold V_T. However, pixel PO$_1$ is exposed to a larger irradiance than pixel PO$_2$; to preserve this information, we can read out the sensor multiple times—four in this case. At each readout, we only output the pixels that became active after the previous readout. This is shown in the four binary frames T$_1$ through T$_4$, which were simulated from frame I$_{orig}$, image courtesy of Fairchild [11]. If we weight the four binary frames with the times at which they were read out, we can create a single, HDR, gray-level image of the scene (see frame Result).

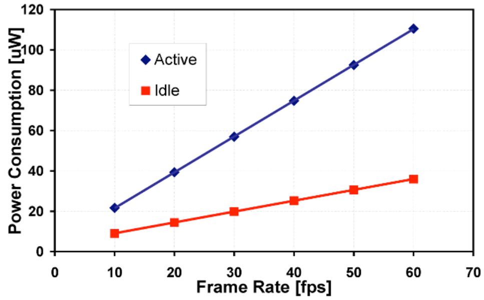

Figure 6. Measured power consumption versus frame rate for a pixel activity of 25% for a single gray level. The two curves converge to a static power consumption of about $3\mu W$.

Simulated data — We use an 8MPixel, 110dB image, I_{HDR}, as a benchmark. We resized it to the actual vision sensor dimensions, 128×64 pixels, and used $N = 4$ readouts, equally distributed over T_{EXP}. We then simulated the proposed readout strategy using I_{HDR} as an input. Figure 4 shows the original image I_{orig} gamma-compressed to visualize a large part of its dynamic range, the images extracted from the sensor at the different readout times, and the final gray-level result.

Given the sensor readout time of $150\mu s$, the maximum number of gray levels that can be extracted from the sensor working at 30fps ($T_{\text{EXP}} = 33ms$), is:

$$N_{\max} = \frac{T_{\text{EXP}}}{T_R} = \frac{33ms}{150\mu s} = 220, \tag{5}$$

which yields a gray-level resolution up to 7 bits.

Real data — After this preliminary verification, we modified the firmware of the CBVS implementing the algorithm described above. In this first prototype, we empirically chose 8 readout phases, *i.e.*, 9 gray levels, because of the favorable trade-off between power consumption overhead and quality of the resulting images. Figure 7 shows two different scenes (top row) acquired with the proposed readout strategy under different illuminations; in addition to the room's diffused fluorescent lighting, we used a 5W, wide-angle (160°), 850nm LED illuminator. Note that the LED illuminator is by no means necessary for the sensor to work properly, we only used it to illuminate different parts of the scene in order to extend its dynamic range.

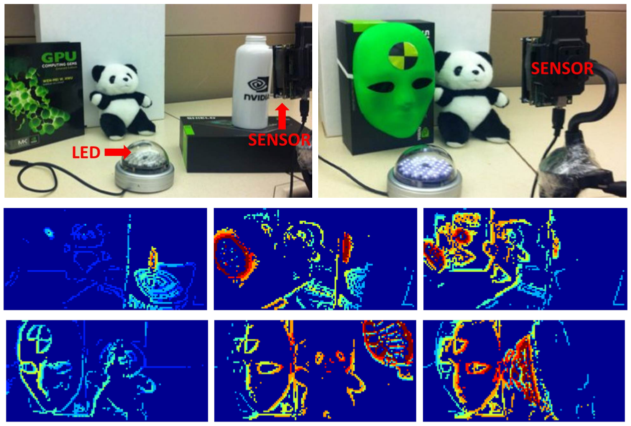

Figure 7. Images captured with the proposed technique; note that the LED illuminator visible in the images in the top row was held in different positions during capture, to simulate different HDR light distributions. The second and third row show captures of the top left scene and top right scene respectively, for three different positions of the illuminator. The eight exposure times used in this test are $T_i = \{200\mu s, 400\mu s, 800\mu s, 1.6ms, 3.2ms, 6.4ms, 12.8ms, 25.6ms\}$. To make the images more readable, a color map has been used to encode the 9 light intensities in the scene, with red indicating the brightest pixels and dark blue indicating pixels with no contrast (*i.e.*, no information on light intensity).

4. A Gesture-detection System

To verify the applicability of the CBVS for complex visual tasks, we developed a prototype system to detect hand gestures in the context of automotive. We attached the sensor to the ceiling of the car's cockpit, with a field of view roughly covering the area between the armrest and the dashboard (see Fig. 1, leftmost column). Standard, low-dynamic-range sensors would be hard-pressed to detect gestures in such a scenario, due to the combination of direct and indirect sunlight illumination, as well as the resulting hard shadows.

For this use case we seek to classify the frames output by the sensor in two categories: *gesture*, indicating the presence of a hand with at least one open finger, and *background*, which covers the remaining scenarios, even that of a hand closed into a fist, or resting on the stick-shift; note that this is necessary for the system to be applicable to real-world cases, but it does impact the robustness that is required of the classification algorithm (see Fig. 1, center and rightmost column respectively). We tackled this problem by designing a convolutional neural network trained to output the probability that a gesture is occurring in the scene, or the probability that the frame is just background.

To create and train the convolutional neural network, we used Caffe, a deep-learning framework highly optimized to run neural networks on GPUs [12]. The network comprises several layers; the first is a convolutional layer with 32 filters of size 16×16 (with a 2-pixel stride), followed by a rectified linear unit layer, a 2×2 max pooling layer, a local response normalization layer, a fully connected layer with 24 outputs, and finally a softmax layer that outputs the probabilities of gesture vs background. We trained the network for 50,000 iterations on a set of roughly 5,000 images acquired with the sensor and the proposed readout method. To improve the generalization capabilities of the net, we augmented the training set by randomly shifting and rotating the training images, obtaining a total of approximately 170,000 training samples; we also added a random gray level noise to 2% of the image pixels. Finally, during training we performed a 50% drop-out after the rectified linear units and fully-connected layers. Testing was performed on a set of 1,062 images; the testing images were not used for training, and were acquired under different illumination conditions than the training images. Such illumination conditions ranged from overcast to sunny weather; the latter, in addition to presenting a higher dynamic range, causes very strong shadows within the field of view of the camera.

The network detects gestures with a precision of 86% and 88% recall (see Table 1), even in the presence of strong light variations in the scene, such as the ones shown in the middle column of Fig. 1 and in the left column of Fig. 8. The background class was recognized with similar precision and recall. Note that our system correctly classifies as background the typical—and particularly challenging—case of a hand resting on the gearshift, as shown in the bottom right pane of Fig. 8; this hand configuration is indeed very similar to that of the gesture shown in the bottom left image of the same figure. Finally, Fig. 9 shows a few examples of incorrect classification.

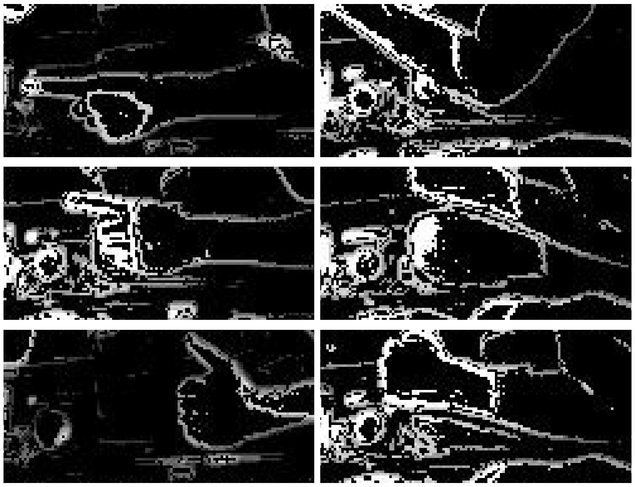

Figure 8. The left column shows three images correctly classified as *gestures* by the proposed system; what looks like artifacts on the arm and hands are the shadows due to the strong, direct sunlight. The right column shows three images correctly classified as *background* by the proposed system.

		Predicted		
		Background	*Gesture*	Recall
Ground truth	*Background*	39%	8%	84%
	Gesture	6%	47%	88%
	Precision	86%	86%	

Table 1. Confusion matrix (with precision and recall for each class) of the gesture detector based on a convolutional neural network and on the sensor described here.

5. Discussion and future work

After introducing the architecture and the operating principle of the CBVS, we described a non-standard readout strategy to extract gray-levels rather than binary images, exploiting the time of arrival of the contrast pixels. Our readout technique could inspire similar approaches for other sensor architectures.

Because the sensor only outputs the magnitude of the

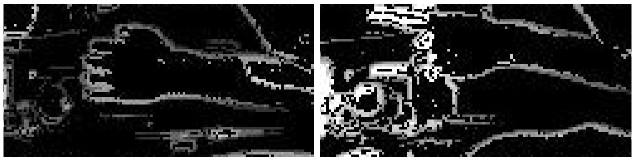

Figure 9. The left and right image are erroneously classified as *gesture* and *background* respectively by the proposed system.

gradient, the intensity image cannot be reconstructed by integration; in this sense, our prototype, shown in Fig. 5, is a simplified embodiment of the gradient camera proposed and simulated by Tumblin and colleagues [13]. However, we show that the images produced allow to perform complex computer vision tasks, such as gesture detection, in challenging scenarios.

An interesting aspect that requires further investigation is the selection of the readout times T_i. Rather than using an unnecessarily large number of levels, the readout strategy can be designed to sample T_{EXP} non-uniformly; for instance one could iteratively change the actual values of the T_i's so as to equalize the histogram of the resulting image. A strategy to optimize the choice of the readout times is left for future work.

The low-power nature of this sensor, together with the high dynamic range it can capture, make it a promising candidate for several vision and computational photography methods. For instance, the proposed gray-level sensor can be used to perform metering for HDR imaging. In stack-based HDR imaging, multiple pictures of the same scene are captured with varying exposure times and then combined into a single HDR image. To select which exposures to capture, state-of-the-art methods require several low-dynamic-range viewfinder frames to estimate the dynamic range of the scene *before* capture [14]. The same problem could be solved with a single frame from the proposed sensor. We leave this investigation for future work.

Acknowledgments

This work was partially supported by the Mobility Programme 2013 of the Fondazione Bruno Kessler.

References

[1] P. Dudek and P. Hicks, "A general-purpose processor-per-pixel analog simd vision chip," *IEEE Transactions of Circuits Systems*, 2005. 1

[2] C.-C. Cheng, C.-H. Lin, C.-T. Li, and L.-G. Chen, "iVisual: An intelligent visual sensor SoC with 2790 fps CMOS image sensor and 205 GOPS/W vision processor," *IEEE Journal of Solid-State Circuits*, 2009. 1

[3] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120dB 30mW asynchronous vision sensor that responds to relative intensity change," in *IEEE ISSCC Dig. Tech. Papers*, 2006. 1

[4] N. Massari and M. Gottardi, "A 100 dB dynamic-range CMOS vision sensor with programmable image processing and global feature extraction," *IEEE Journal of Solid-State Circuits*, 2007. 1

[5] G. Kim, M. Barangi, Z. Foo, N. Pinckney, S. Bang, D. Blaauw, and D. Sylvester, "A 467 nW CMOS visual motion sensor with temporal averaging and pixel aggregation," in *IEEE ISSCC Dig. Tech. Papers*, 2013. 1

[6] J. Choi, S. Park, J. Cho, and E. Yoon, "A 3.4 μW CMOS image sensor with embedded feature-extraction algorithm for motion-triggered object-of-interest imaging," in *IEEE ISSCC Dig. Tech. Papers*, 2013. 1

[7] N. Cottini, M. Gottardi, N. Massari, R. Passerone, and Z. Smilansky, "A 33 μW 64×64 pixel vision sensor embedding robust dynamic background subtraction for event detection and scene interpretation," *IEEE Journal of Solid-State Circuits*, 2013. 1

[8] P.-F. Ruedi, P. Heim, F. Kaess, E. Grenet, F. Heitger, P.-Y. Burgi, S. Gyger, and P. Nussbaum, "A 128×128 pixel 120-dB dynamic-range vision-sensor chip for image contrast and orientation extraction," *IEEE Journal of Solid-State Circuits*, 2003. 1

[9] M. Gottardi, N. Massari, and S. Jawed, "A 100 μW 128×64 pixels contrast-based asynchronous binary vision sensor for sensor networks applications," *IEEE Journal of Solid-State Circuits*, 2009. 1, 2, 3

[10] W. Zhang, Q. Fu, and W. N., "A programmable vision chip based on multiple levels of parallel processors," *IEEE Journal of Solid-State Circuits*, 2011. 1

[11] M. D. Fairchild, *The HDR Photographic Survey*. MDF Publications, 2008. 4

[12] Y. Jia, "Caffe: An open source convolutional architecture for fast feature embedding," http://caffe.berkeleyvision.org/, 2013. 5

[13] J. Tumblin, A. Agrawal, and R. Raskar, "Why I want a gradient camera," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005. 6

[14] O. Gallo, M. Tico, R. Manduchi, N. Gelfand, and K. Pulli, "Metering for exposure stacks," *Computer Graphics Forum (Proceedings of Eurographics)*, 2012. 6