Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2007, Article ID 52630, 10 pages
doi:10.1155/2007/52630

Research Article

Nearest Neighborhood Grayscale Operator for Hardware-Efficient Microscale Texture Extraction

Christian Mayr[1] and Andreas König[2]

[1] *TU Dresden, Lehrstuhl Hochparallele VLSI-Systeme und Neuromikroelektronik, Helmholtzstraße 10, 01062 Dresden, Germany*
[2] *TU Kaiserslautern, FB Elektrotechnik und Informationstechnik, Lehrstuhl Integrierte Sensorsysteme, Erwin-Schrödinger-Straße, 67663 Kaiserslautern, Germany*

Received 23 November 2005; Revised 1 August 2006; Accepted 10 September 2006

Recommended by Montse Pardas

First-stage feature computation and data rate reduction play a crucial role in an efficient visual information processing system. Hardware-based first stages usually win out where power consumption, dynamic range, and speed are the issue, but have severe limitations with regard to flexibility. In this paper, the local orientation coding (LOC), a nearest neighborhood grayscale operator, is investigated and enhanced for hardware implementation. The features produced by this operator are easy and fast to compute, compress the salient information contained in an image, and lend themselves naturally to various medium-to-high-level postprocessing methods such as texture segmentation, image decomposition, and feature tracking. An image sensor architecture based on the LOC has been elaborated, that combines high dynamic range (HDR) image aquisition, feature computation, and inherent pixel-level ADC in the pixel cells. The mixed-signal design allows for simple readout as digital memory.

1. INTRODUCTION

In today's integrated vision systems, their speed, accuracy, power consumption, and complexity depend primarily on the first stage of visual information processing. The task for the first stage is to extract relevant features from an image such as textures, lines, and their angles, edges, corners, intersections, and so forth. These features have to be extracted robustly with respect to illumination, scale, relative contrast, and so forth. Several integrated pixel sensors operating in the digital domain have been proposed, for example, Tongprasit et al. [1] report a digital pixel sensor which carries out convolution and rank-order filtering up to a mask size of 5×5 in a serial-parallel manner. However, in [2], implementations of a low-level image processing operator realized either as a mixed-signal CMOS computation, dedicated digital processing on-chip, or as a standard CMOS sensor coupled to FPGA processing are compared. A case is made that a fast, low-power consumption implementation is best achieved by a parallel, mixed-signal implementation. However, the downside of coding the feature extraction in hardware are severe limitations as to flexibility of the features with regard to changing applications [3], whereas software-based feature extractions could simply be partially reprogrammed to suit various applications [4]. Several architectures of mixed-signal CMOS preprocessing sensors have been implemented recently [3, 5, 6] that achieve a compromise in the form of a sensor which extracts a very general yet high-quality set of features, with the higher-level processing done in software or on a second IC [2, 5]. One operator which is very apt to this kind of implementation is the local orientation coding (LOC), which encodes the nearest neighbor grayscale texture and orientation information [7].

This paper is organized as follows: first, we will restate the basic tenets of the LOC, its origin, and the modifications realized to aid in hardware implementation and resultant feature quality. Second, we will give some of the results obtained with the operator for image decomposition and quality inspection. Third, we will document the hardware implementation using a high dynamic range (HDR) sensor and continuous analog circuits. As a last point, an outlook on future work is presented, especially the current efforts to design a neural, pulse-based version of this sensor.

$$\begin{pmatrix} 0 & 1 & 0 \\ 2 & \mathrm{R} & 4 \\ 0 & 8 & 0 \end{pmatrix} \begin{pmatrix} 1 & 2 & 4 \\ 8 & \mathrm{R} & 16 \\ 32 & 64 & 128 \end{pmatrix} \begin{matrix} n \uparrow \\ \rightarrow m \end{matrix}$$

Figure 1: Binary scaled N_4 and N_8 neighborhood coefficients $k(i, j)$.

2. LOC OPERATOR

The LOC operator was introduced by Goerick and Brauckmann [7] as a simple method to code localized textures and orientation in a 3×3 pixel neighborhood based on grayscale comparisons. It is very modest with regard to its computational demands, as no multiplication is required.

2.1. Basic tenets modifications

The outcome of the LOC operator constitutes a unique topology-specific feature number $b'(m, n)$ for every pixel $b(m, n)$, with (m, n) denoting image coordinates:

$$b'(m,n) = \sum_{i,j} \varepsilon_{m,n}(i,j). \tag{1}$$

This feature number $b'(m, n)$ is composed of a sum of the coefficients $\varepsilon_{m,n}(i, j)$, specific for the pixels neighboring pixel (m, n). The computation of $\varepsilon_{m,n}(i, j)$ and the neighborhood $(i \cdot j)$, in which this computation is carried out, is defined in (2):

$$\varepsilon_{m,n}(i,j) = \begin{cases} k(i,j), & b(m+i,n+j) \le b(m,n) - t(i,j), \\ 0, & \text{else}, \end{cases}$$
$$(i,j) \in \{(0,-1),(-1,0),(1,0),(0,1)\} \quad \text{for } N_4,$$
$$(i,j) \in \{(-1,-1),(-1,1),(1,-1),(1,1) \cup N_4\} \quad \text{for } N_8. \tag{2}$$

The pixel gray value $b(m, n)$ of the middle pixel in a 3×3 neighborhood minus a directional threshold $t(i, j)$ is compared to each gray value of the four (eight) neighbors $b(m+i, n+j)$. If the result of the comparison in (2) is positive, that is, the neighboring pixel deviates significantly from the middle pixel, $\varepsilon_{m,n}(i, j)$ constitutes the direction-dependent coefficient $k(i, j)$, otherwise zero is returned. Binary scaling of the coefficients $k(i, j)$ is of course the logical choice to make the feature number $b'(m, n)$ uniquely separable into its components, so for N_4 and N_8 neighborhoods, the codings in Figure 1 were chosen in [7].

To give an example, for an image coordinate system origin in the upper left corner, an N_8 neighborhood and $(i, j) = (-1, 0)$, $k(i, j)$ would be 8.

The threshold $t(i, j)$ is derived from the first significant minimum in a directional histogram of the complete image. The reasoning behind this is to suppress susceptibility to noise and code significant image features. If a neighboring pixel was compared directly to $b(m, n)$, noise in the 3×3 neighborhood could cause $b(m + i, n + j)$ to be slightly below the gray value of the middle pixel even though they might belong to the same feature in the image, thus giving a false response. We will not treat this directional threshold in more detail, since it will be exchanged for a more localized, omnidirectional threshold in (3) through (5). For details on the directional threshold, please see [7].

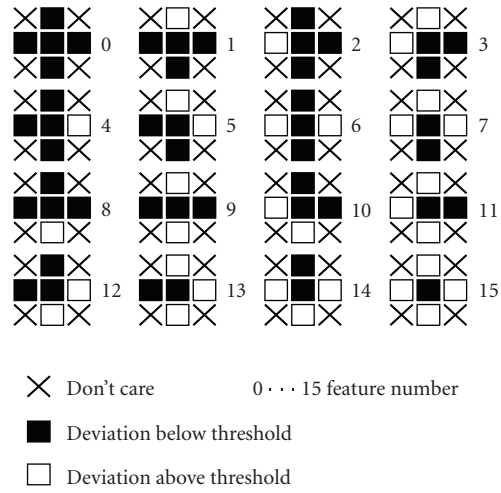


Figure 2: Possible N_4 neighborhood features.

For an N_4 neighborhood, all possible operator outcomes and their respective feature numbers are given in Figure 2. As can be seen, a variety of local grayscale texture information is captured by the operator, ranging from single significant points (feature 15), continuous lines (6, 9), terminated lines (7, 11, 13, 14), corners (3, 5, 10, 12), T-sections (1, 2, 4, 8) to complete intersections (0).

As can be seen from (1) and (2), only simple mathematical operations like addition, subtraction, and comparison are needed to compute the operator, making it an ideal choice for a low-power, optimized parallel-analog VLSI implementation. Even the feature number in a single pixel cell can be computed in parallel, by using four or eight comparators at the same time. The outcome of these analog computations, namely, the final comparison, could then be stored digitally, making for early image information extraction and condensation, as well as easy readout and feature manipulation, that is, histogram computation.

However, the LOC operator in its present form still poses some severe obstructions to a hardware implementation, especially that the image-wide directional grayscale histogram used for finding the directional threshold $t(i, j)$ in [7] does not lend itself easily to a fully integrated, parallel implementation, since such a histogram could only be computed with global connectivity. One of the objectives for the modification of the LOC operator had to be then to replace this global threshold with some kind of locally computed one. The basic idea employed in designing this modified threshold is to extract weak differences (textures), if the local pixel environment has limited change in grayscale value, that is, contrast, but to only look for strong textures if the local contrast is high. Also, directional dependency has been discarded in favor of a unidirectional significance measure, using the LOC

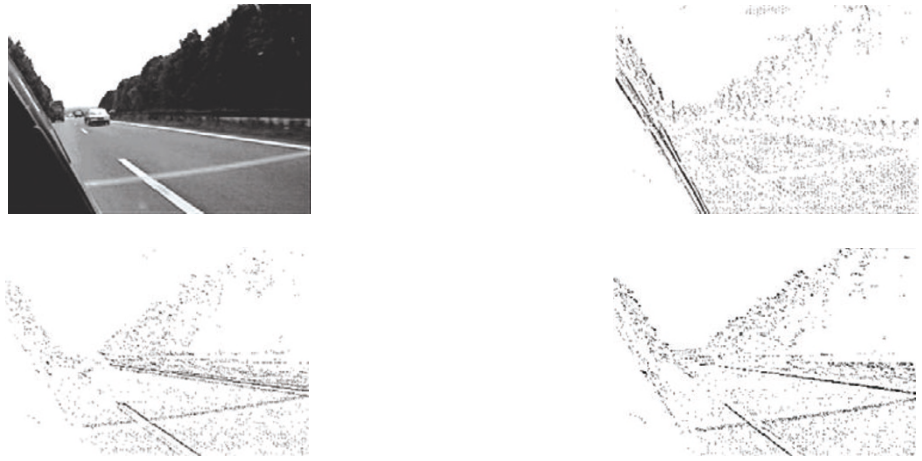


Figure 3: Sample image from car overtake monitoring system with (clockwise, from top left) original image and results for feature numbers 14, 10, and 5, respectively.

not so much for orientation extraction but rather texture and localized structure coding. This is expressed by changing from threshold $t(i, j)$, which is the same for every pixel $b(n, m)$, but differs according to direction (i, j) of the neighboring pixel, to $t(m, n)$, which is not direction dependent, but is different for every pixel $b(m, n)$ to reflect a local significance measure. Since the term feature generally denotes a large-scale object in an image, the terms structure and texture are used interchangeably in the following to denote the kind of localized pixel interdependency extracted by the LOC operator.

Several different thresholds were implemented in a software version of the operator, for example, the local standard deviation, or the absolute difference between average local grayscale and the pixel under consideration. Best results were obtained for $t(m, n)$ equal to the absolute difference (5) between the pixel grayscale value $b(m, n)$ and a Gaussian smoothing $g(m, n)$ of the picture (3), with the normalization for the Gaussian convolution mask provided by the sum of its coefficients (4). Please note that a significance assessment based on this measure is not marginal (i.e., only judges based on the same 8 pixels evaluated by the LOC), since the Gaussian smoothing has a catchment area up to the whole image, depending on its σ. The radius r used for the convolution mask has been kept to $2 \times \sigma$ for the simulations.

$$g(m,n) = \sum_{i,j} \left[b(m+i, n+j) \times \frac{1}{Z} e^{-(i^2+j^2)/2\sigma^2} \right], \quad (3)$$

$$Z = \sum_{i=-r}^{r} \sum_{j=-r}^{r} e^{-(i^2+j^2)/2\sigma^2}, \quad (4)$$

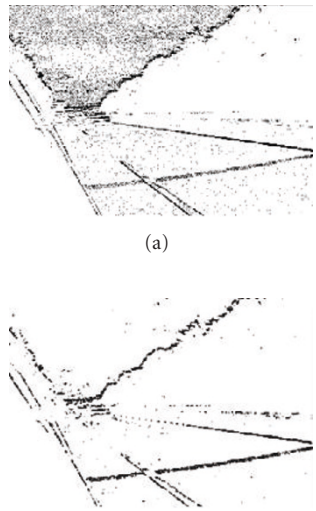
$$t(m,n) = C \times \left| b(m,n) - g(m,n) \right|. \quad (5)$$

The scaling factor C has been introduced in (5) to facilitate adapting the LOC structures to different applications, as experiments indicate that the type of LOC structure extracted from an image has to be adjusted to the application, that is, its noise levels, brightness variation in a localized context, or how much variation across pixel gray values is allowable for a texture. The second parameter used for adjusting LOC structures to the application at hand is the extension of the smoothing σ. For example, to extract LOC structure from a natural image, σ would be set to a narrow smoothing, because lighting conditions vary widely across the image, and C could then be used at a low setting of, for example, 0.5 to extract textures with very similar gray value, to, for example, find an edge with only gradually changing reflective properties along its length. On the other hand, a C of, for example, 3 would allow for discontinuities in reflective properties, with the penalty of extracting pseudotextures/structures not justified by underlying image objects, where dissimilar pixels are counted as belonging to a single LOC structure because of the wider (and in this case erroneous) catchment range. Adapting the LOC operator to an application via C and σ captures the spirit of a general yet parametrizable hardware preprocessing sensor mentioned in the introduction.

2.2. Results for software implementation

A C++ implementation of the operator and its modifications has been carried out based on a software tool for image analysis and classification [8]. The software implementation offers two output formats, either feature numbers as single-frame images (used for higher-level image processing) or feature histograms, which can be directly employed for classification purposes. A sample for the former output format is given in Figure 3.

Notice how differently oriented lines show up selectively on the resulting images. Also, the features are selective to the direction of the contrast, with features 10 and 5 showing only the upper, respectively, lower edge of the midline of the street. Since the LOC operator itself does not use global dependencies in our modified version, the output is somewhat noisy. However, this can be cleaned up efficiently based on a nearest neighborhood majority decision. The basic assumption is that feature responses based on noise will tend to be isolated,



(a)

(b)

Figure 4: Sample image from car overtake monitoring system, comparison of original feature number 10 image (a) and denoising via neighborhood majority decision (b).

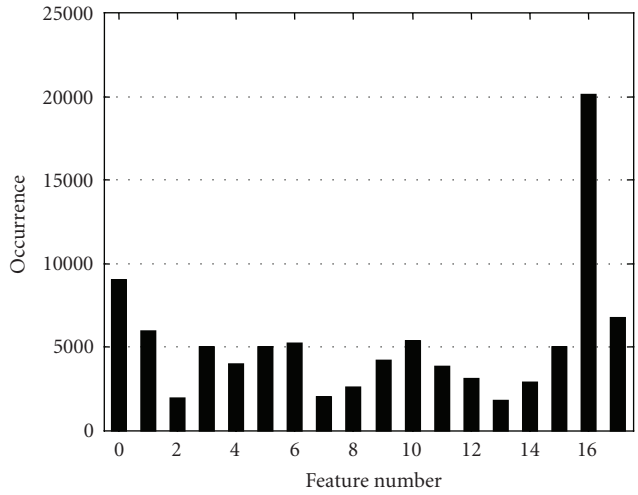


Figure 5: Sample histogram of original image Figure 2, 16 coded feature numbers plus flags for low- and high-local contrast (16, resp., 17).

while those comprising real image structure will be clustered. The best performance was found for a "simple majority," that is, at least 4 of the surrounding 8 pixels exhibit the same feature. An example for this denoising is given in Figure 4 (feature number 10 with smaller σ, leading to more noise in the bright sky area, but improved reproduction of the border sky greenery).

This denoising, while not part of the hardware implementation discussed in Section 2.3, could be incorporated very easily on the sensor, since it also depends only on local image information. The histogram output mode for the above image is shown in Figure 5.

Two additional features were introduced for the LOC modification discussed herein, low- and high-local contrast, both based on thresholding the significance criterion $t(m, n)$ (normalized by local grayvalue) to achieve a measure of picture contrast, shown in Figure 5 as feature numbers 16 and 17, respectively. This was done basically to have a "measure of quality" for the LOC features produced by a certain pixel, which could be used by subsequent processing stages. The reason is that a high-local variability indicates significant features, while low-local variability would indicate that the features are mostly computed from random variations in pixel current (i.e., noise).

The feature histogram of the street image reveals a lot of uniform image areas (regular feature number 0 and extended feature number 16), comprising for instance the sky, the uniform areas of the street, and some of the greenery on both sides. Vertical features have also been found (features 2, 4, and 6), but with a notable difference in left-right contrast (features 2 and 4), since vertical structures occur primarily on the left side of the picture caused by the recording vehicle, with a contrast oriented in only one direction. As well, the various diagonal structures in the image can be found in the histogram count of features 3, 5, 10, and 12. Terminated line features like 7 and 13 also show a noticeable difference to their counterparts 14, respectively, 11, elaborating on the images' tendency for left-right and up-down bright-dark contrasts. Figures 3–5 show that this feature computation method extracts relevant image information.

Using the feature histogram output mode, a reduced nearest neighbor (RNN) classifier [5, 8] has been trained to recognize eye shapes. Figure 6 shows the trainings and test class spaces, left half, respectively, right half, reduced to two dimensions using Sammon's mapping [8]. Axis captions are omitted because they are a nonlinear, adaptive function of the input-feature vector, and would carry little meaning with respect to the original features. The insets in the upper corners show samples for the darker class space (eye existent, EE), respectively, the lighter class space (eye not existent, ENE). The RNN classifier was trained for separating the two classes EE and ENE with 14 examples of eye regions as indicated in the inset in the upper left corner, and 27 examples of class ENE, captured from random locations of the full-head images that the eye regions were extracted from, similar to the image underlying Figure 7. After learning, the RNN classifier has been tested with a sample set including 43 examples for the ENE class and 15 examples for the EE class. The recall and precision rates are equal to 100%, that is, there are zero instances for EE classified as ENE and vice versa, although the EE class space is not as coherent in the test case (right half picture, dark area).

This example uses a feature histogram vector composed of the 16 vertical/horizontal features shown in Figure 1. An automated feature selection has been employed to single out the features having the most impact on correct classification [5, 8], thus improving classification quality and speed, since classifiers trained on very high-dimensional data tend to "learn" the training sample set, while not being able to

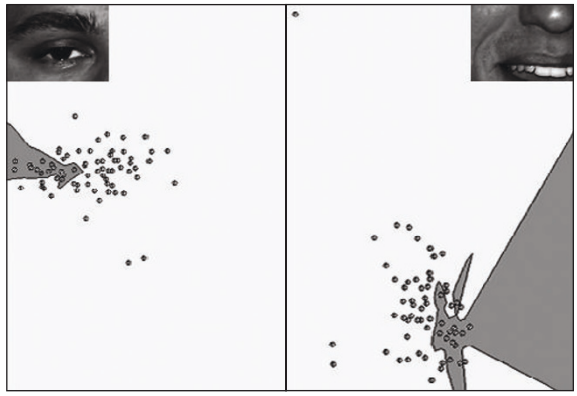


Figure 6: Trainings and test class spaces for eye sample data using LOC features.

Figure 7: Image from visual telephone image sequence "Claire" with detected eye regions marked in black or white.

generalize well. This is also evinced by the fact that if the features produced by the complete N_8 neighborhood are presented to the classifier, its classification quality decreases to 87.3%, evidently not able to cope with generalization in the context of the resultant increase in search space dimensionality. A more complete description of the experiment and comparison with results achieved, for example, for Gabor jet feature, can be found in [5]. The slight difference in classification results for LOC in [5] compared to the one reported herein is caused by the nondeterministic approach of the feature selector mentioned above.

As a real-world test of this classifier, a complete human passport image (Figure 7) has been scanned by the classifier using a scanning window of approximately eye size.

Center pixels which have elicited a positive eye response in their corresponding scanning window are marked in black or white, dependent on the local contrast so as to be best visible. Figure 7 shows that the eye regions have been detected robustly, with especially the left eye (right half of the image) having a large number of positive identifications. Faulty classifications are reported for the lower lip and part of the collar. This classification could be reached with a two IC hardware-based version of the classifier and LOC operator, with possibly a low-performance microprocessor to do a final model-based geometric analysis and select the correct eye locations. Thus, the goal of computing high-quality features and reducing data rate for subsequent high-level processing stages could easily be achieved in this image analysis/segmentation application. The operator has also been tested in a similar classification testbench with sample images of a production line for circuit breakers. The aim was quality inspection, that is, discerning and discarding faulty breakers. Testing of the LOC operator in this application also brought comparably high-classification results, proving the efficacy of the computed features for a task of quite different scope, as well as indicating the broad range of tasks the modified operator could be used on.

Even though the discussed image operator is not a very recent development [7], when compared to state-of-the-art image operators for texture and local orientation analysis [3, 6], it can be reasoned that LOC gives qualitatively similar results. As mentioned in the introduction, the aim of this research is not to develop a hardware sensor dedicated (and limited!) to one single application, but one that produces a selection of salient features comprising local image structures in such a way that subsequent software-based processing stages have a greatly reduced work load while still being able to extract high-level image information such as the examples mentioned above. Macroscopic textures/features such as the ones analyzed in [4] are characterized by a distinct local mixture of microscopic, that is, LOC texture features. Local histograms of the LOC features would thus be sufficient to separate macroscopic textures. Macroscopic image orientation could also be computed from the LOC features, with 8 main directions (in the case of an N_8 neighborhood LOC) instantly available from the increased local occurrence of single, elongated features such as feature 6 of Figure 2 (compare lines in Figure 3). Intermediate image orientations are characterized by a mix of the LOC features closest in orientation to the one exhibited by the image, which also makes them discernible in localized histograms. Even if subsequent stages need to operate on the raw image data, they could still use the hardware LOC sensor as a region-of-interest (ROI) selector, choosing to do high-level image analysis only on the regions denoted by LOC features, which indicate relevant image information (Figures 3 and 4).

Please note that the two applications shown in Figures 3, 4, and 7 are of course only basic examples for usage of the LOC sensor. Especially the eye finder is limitedly scale invariant and not at all rotation invariant, and the histogram output is prone to produce the same histogram for different image contents, as is evident from the erroneous classifications around the collar and lip. However, we believe that the two examples show the efficacy of the modified operator through their very simplicity coupled with the good results of the classification in Figures 6 and 7. Both spatial feature relationships and rotational invariance could, for example, be achieved by using the raw LOC features of Figures 3 and 4 as input for a classifier such as [4]. Spatial information could be used by training a cascaded RNN on parts of eye shapes, thus improving classification results by eliminating structures with identical overall histograms.

2.3. Hardware implementation

A hardware implementation of the adjusted LOC operator has been carried out in a 1P3M 0.6 μm CMOS process, using the maximum current range available from the photo diode to achieve a true high dynamic range (HDR) design [3, 6]. The feature computation process is carried out entirely with a current-based representation of the grayscale value, starting with the photo current itself, in order to keep the original dynamic range of the photo sensor. Thus, LOC features depending only on current differences can be computed without adjusting the sensor to surrounding illumination levels.

Circuits to achieve these computations were adapted from [9–11], with the normalization for local-contrast characterization carried out in a standard translinear circuit [10]. However, because of accuracy and dynamic range requirements, the variable current scaling (5) has been implemented in a somewhat modified translinear circuit, using fixed current multiplication in current mirrors and subsequent variable current splitting in a differential amplifier [12].

In Figure 8, the circuit for computing an absolute value current is shown, as adapted from [9]. A biasing current for P1 and P4 is derived from the (reduced) current output flowing through N2, with N1 having about one tenth of the W/L of N2. P1 and P4 in turn bias their counterparts P2 and P5. If a current is drawn from I_{in} to ground/VSSA, pMOS transistors P2 and P3 act as current mirror, and the voltage node at input I_{in} is drawn to ground because of the increased V_{GS} of P2 compared to P1 (with its smaller biasing current relative to I_{in}), thus turning off P5. Current I_{in} is then simply forwarded through P2 and P3 to N2, where it can be used as a gate voltage $V_{equIout}$ for nMOS transistors matched to N2 to distribute I_{in}. In the second case, that is, a current is flowing into I_{in} from the supply rail VDDA, the V_{GS} of P5 will increase, thus increasing the potential at node I_{in} and turning of P2, because the gate voltage of P5 is defined (i.e., fixed) by P4. In this case, P5 acts as a current conveyor or pass transistor, forwarding I_{in} to N2. Hence, irrespective of the direction of the current into I_{in} (source/sink), it will always flow through N2 in the same direction.

The complete pixel cells consist of the following (compare to Figures 9 and 10, which depict the layout and block diagram, resp.):

(i) the photo diode ⟨1⟩,
(ii) time-continuous diffusion network ⟨2⟩ for the adjustable averaging of local light levels (3),
(iii) absolute current value circuit ⟨3⟩ to compute the absolute difference between local average and photo current of the cell (5),
(iv) the current amplifier ⟨4⟩ for scaling the absolute difference to achieve different feature sensitivity [12] (5),
(v) the current mirrors ⟨5⟩ to compute the reference composed of the difference between photo current and scaled absolute difference ((1), modified with $t(m, n)$ of (5)),
(vi) the current comparators ⟨8⟩ to compare the reference to the neighbor photo currents (1),

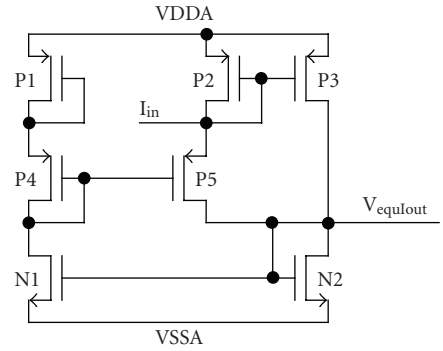


Figure 8: Circuit for computing the absolute value of a current.

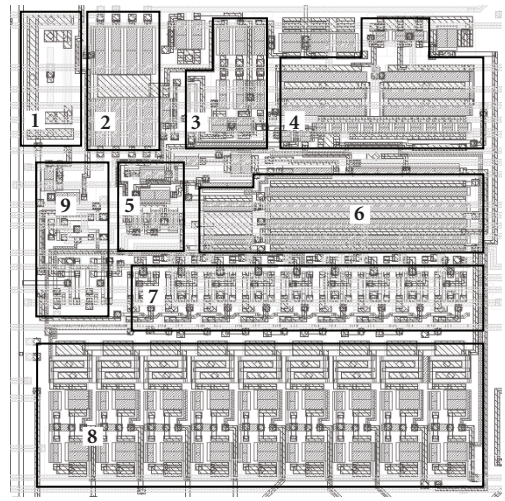


Figure 9: Layout of the pixel cell implementation.

(vii) a translinear circuit to normalize the reference with the photo current and compare the normalized result to an external threshold to achieve a measure for the variability in local photo currents ⟨6⟩,
(viii) the SRAMs to store the comparison results (9 Bit) ⟨7⟩,
(ix) digital readout circuitry ⟨9⟩,
(x) equation (2) is performed implicitly by reading each single comparison bit off-chip, rather than computing a feature sum in the pixel cell itself.

The block diagram shows the computational flow and dependencies of the image information processing units listed above. Also, the IO connectivity of the pixel cell is given, taking input from its own photo diode, from the photo diodes of the neighbors, and from the diffusion network. Global external adjustments are also fed to the cell, governing analog aspects such as the adjustment of the properties of the computed features to the postprocessing carried out on them, and parts of the computation process of the two extended features (high- and low-local contrast), such as normalization. Also, digital control signals are fed to the cell, selecting the relevant neighborhood and number of digital features computed, as

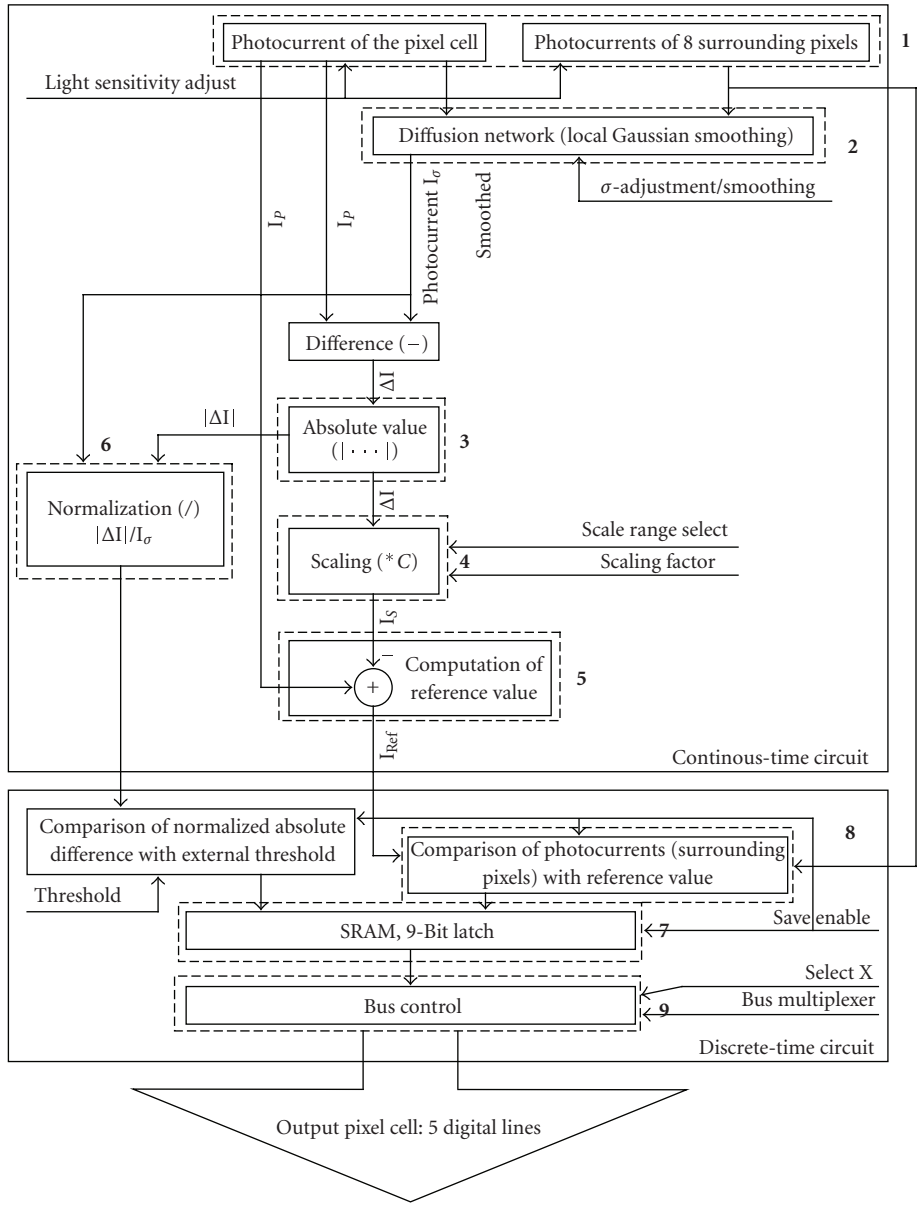


Figure 10: Block diagram of the pixel cell implementation.

well as defining the readout sequence for the tristate digital feature bus.

Figure 11 illustrates the temporal performance of a pixel cell. The photo current for the middle pixel is 20 pA and the output of the current comparators after a stimulus change at the sensor input for three selected neighbors at t = 20 ms is shown. The reference value as illustrated in Figure 10 (step ⟨5⟩) is 8.3 pA, as computed from the middle pixel photo current, the output of the resistive network of 11.6 pA, and a scaling C equal to 1. The stimuli change from uniform 0 pA to (from top to bottom) 50, 10, 5 pA.

The computation times (6.8, 14.4, resp., 24.2 ms) are comparable to the ones reported in [6]. However, because of the time-continuous nature of the analog computation in the pixel cell, the LOC features can be read from the pixel cells at any given time, there is no hardware reset or integration time needed [6], changes to lower-light levels simply take more time to propagate to the LOC feature output. Hence, there is no "frame rate" per second. The frame rate reported in Table 1 has been chosen to represent standard room lighting, with the lowest light level generating about 50 pA photo current. The entire analog feature extraction has been simulated over 4 decades of photo current, that is, 1 pA to 10 nA (13 Bit), equivalent to an operability of the sensor and feature extraction ASIC over a range from bright daylight to dark twilight.

Monte-Carlo simulations of the pixel cell have been carried out to verify the accuracy of the analog computa-

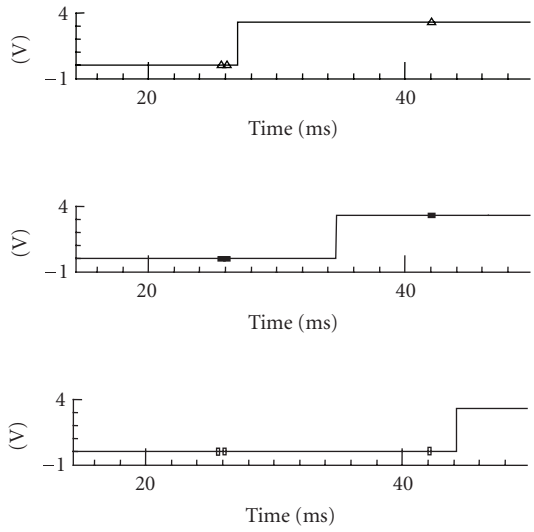


Figure 11: Response time of pixel cell from input current (i.e., brightness) change to feature output (stimuli change at 20 ms).

Table 1: (Simulated) characteristics of the LOC pixel cell/sensor array.

Technology	1P3M 0.6 μm CMOS
Pixel cell size	$(83 \times 80)\,\mu\text{m}^2 = 6640\,\mu\text{m}^2$
Pixel cell fill factor	3.8%
Pixel cell dynamic range	84 dB
Pixel cell absolute accuracy	25 dB (10 pA) 28 dB (10 nA)
ASIC size	$1560\,\mu\text{m} \times 2240\,\mu\text{m}$
ASIC frame rate	140
ASIC array size	16×26 pixel cells
ASIC power Consumption (analog)	161 μW
ASIC power consumption (digital) (without bonding capacity, only bondpads)	10 μW

tions, establishing a 25 dB accuracy at low photo currents of 10 pA, and 28 dB at 10 nA with a confidence of 90%. Two counteracting effects have been observed which act to keep accuracy almost constant over the whole dynamic range. On the one hand, translinear circuits work best at low-current levels, where all transistors are operating firmly in subthreshold [10], on the other hand, current mirrors, which are employed in various stages of the analog computation, are subject to statistical variations at low currents, improving progressively with higher current levels [9].

While the software implementation is primarily intended as a stand-alone tool in the framework of the system discussed in [8], it has also been used to verify the LOC pixel cell. The software LOC can be operated in an HDR mode with 16 Bit resolution, which has been used to show validity of the modified LOC concept, especially the new significance threshold introduced in (5), in an HDR context. Correspondingly, the radius r of the convolution mask in (3) and (4) has been extended to cover the whole image, to account for the larger dynamic range of the input image being able to influence more distant pixels. Second, a more realistic response of the resistive diffusion network, that is, "decaying exponential" [11] instead of Gaussian, has been used with the software LOC, with little change in LOC structure extraction. For example, the EE/ENE classification discussed in Section 2.2 still gives 100% classification result with "decaying exponential" smoothing. Third, the accuracy numbers obtained from the Monte-Carlo simulation have been used in the form of an artificially introduced 5% ($\cong$ 26 dB) error (uniform distribution) on the right-hand side of (1), that is, $b(n,m) - t(n,m)$. While this represents a rather crude approximation of the Monte-Carlo outcomes, it also reflects an upper bound, since the real error is more centered around a mean. Incorporating this error in the EE/ENE classification results in one erroneous classification of an eye sample (EE class) as ENE class.

The pixel cell has been realized with a size of $(83 \times 80)\,\mu\text{m}^2$. The corresponding ASIC with additional analog and digital interface and control circuitry has been manufactured, but measurement results are not yet available. It is operating in a simple scan mode, with all LOC feature latches connected to the same bus via tristate gates. The digital power consumption given in Table 1 reflects the power consumed by the latches when charging bus and bondpad capacity, as well as the power consumed by the pixel address counters and decoders, and address lines, at the indicated frame rate. Simulated performance characteristics for the pixel cell are given in Table 1.

The fill factor of the pixel cell is comparable to the one reported in [6], which carries out processing of similar complexity. Dynamic range and absolute accuracy are less, but have proven to be adequate to the application. Massari et al. [3] report a higher fill factor and smaller pixel cell size, but this is due at least partially to the smaller technology used, and the computation is somewhat simpler, relying only on the absolute value of pixel photo current differences. Adjusting for array size, (simulated) power consumption is still lower by at least an order of magnitude, due to the subthreshold working regime in our sensor. Power consumption compares even more favorably to [6]. As has been shown in [2], an analog/mixed signal implementation at this stage of technological advancement is a very competitive alternative to a purely digital feature extraction process. However, since analog circuits do not scale well with advancing technologies, power consumption and size will not shrink as rapidly as in a digital version, so a more digitally-centric LOC realization would be desirable.

2.4. Future developments

Current research work deviates from the continuous time analog implementation described herein. While the operator is quite successful and comparably easy to implement in hardware in the modified fashion, still simpler variants of it could be explored. An especially promising avenue of exploration is the field of pulse-based image processing. Given a pulsing pixel cell as an input, whose pulse rate is equivalent to the grayscale value of the pixel, it has been found that a simple rank order coding theorized from biological evidence of pulse computation is capable of producing very similar features to the ones discussed herein [13]. This rank order coding can be achieved using digital variants of synapses and neurons. Error-prone analog normalization, scaling, addition, and subtraction can all be eliminated from the pixel cell, resulting in a predominantly digital and more robust implementation, as well as reducing the design time. Also, the output signal can be easily represented in a pulse form and fed to, for example, a pulse-based clustering algorithm, or be used for various digital processing stages, since pulse computations of this nature are very similar to digital information representations. In contrast to the conventional digital image filtering discussed in [1], this processing would still be fully parallel and can be incorporated into the pixel cell in the same manner as the analog computation discussed herein.

3. CONCLUSION

We have presented a scheme for fast, computationally inexpensive, massively parallel and flexible hardware-based feature extraction. Quality of the feature extraction has been documented using a sample eye finder application as well as sample images from an early feasibility study of a car overtake monitoring system. In both cases, highly significant points of the ROI have been extracted and their efficacy in distinguishing target shapes, that is, eyes, is shown. The original image operator has been adjusted with respect to connectivity, parameters, and computational requirements for the ease of the analog/mixed-signal hardware implementation. An HDR CMOS sensor design has been carried out to take full advantage of the analog dynamic ranges and computational domains possible on a modern CMOS process while still achieving a digitally coded, data rate reduced feature output. This feature output can be used on-chip, that is, with a digital histogram computation over a selected ROI, to extract a feature vector for that ROI which can be fed directly into a classifier network or be used for further computations off-chip.

ACKNOWLEDGMENTS

The major part of the reported work has been carried out in the Projects GAME and GAMPAI which were funded in the research program VIVA SPP 1076 by the German research foundation "Deutsche Forschungsgemeinschaft." All responsibility for this paper is with the authors. The authors thank Austria Mikro Systeme International AG for the technical support and the D4D group, TU Dresden, computer science, AI Institute, for the kind providance of project-related data and information. The contributions of Michael Eberhardt, Robert Wenzel, Jens Döge, and Jan Skribanowitz to the project in general and their invaluable technical assistance to the presented work are gratefully acknowledged. Many thanks also go to the three anonymous reviewers for their helpful comments on improving the quality and clarity of this paper.

REFERENCES

[1] B. Tongprasit, K. Ito, and T. Shibata, "A computational digital-pixel-sensor VLSI featuring block-readout architecture for pixel-parallel rank-order filtering," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05)*, vol. 3, pp. 2389–2392, Kobe, Japan, May 2005.

[2] A. Elouardi, S. Bouaziz, A. Dupret, J. O. Klein, and R. Reynaud, "Image processing vision system implementing a smart sensor," in *Proceedings of the 21st IEEE Instrumentation and Measurement Technology Conference (IMTC '04)*, vol. 1, pp. 445–450, Como, Italy, May 2004.

[3] N. Massari, M. Gottardi, L. Gonzo, D. Stoppa, and A. Simoni, "A CMOS image sensor with programmable pixel-level analog processing," *IEEE Transactions on Neural Networks*, vol. 16, no. 6, pp. 1673–1684, 2005.

[4] B. Zitová, J. Kautsky, G. Peters, and J. Flusser, "Robust detection of significant points in multiframe images1," *Pattern Recognition Letters*, vol. 20, no. 2, pp. 199–206, 1999.

[5] A. König, C. Mayr, T. Bormann, and C. Klug, "Dedicated implementation of embedded vision systems employing low-power massively parallel feature computation," in *Proceedings of the 3rd VIVA-Workshop on Low-Power Information Processing*, pp. 1–8, Chemnitz, Germany, March 2002.

[6] P.-F. Rüedi, P. Heim, F. Kaess, et al., "A 128 × 128 pixel 120-dB dynamic-range vision-sensor chip for image contrast and orientation extraction," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 12, pp. 2325–2333, 2003.

[7] C. Goerick and M. Brauckmann, "Local orientation coding and neural network classifiers with an application to real time car detection and tracking," in *Proceedings of the 16th Symposium of the DAGM and the 18th Workshop of the ÖAGM*, W. Kropatsch and H. Bischof, Eds., Springer, New York, NY, USA, 1994.

[8] A. König, M. Eberhardt, and R. Wenzel, "A transparent and flexible development environment for rapid design of cognitive systems," in *Proceedings of 24th Euromicro Conference*, vol. 2, pp. 655–662, Vasteras, Sweden, August 1998.

[9] A. Günther, "Design of a library of scalable, low-power CMOS cells for classification and feature extraction in integrated cognition systems," Diploma thesis, University of Technology, Dresden, Germany, April 2000.

[10] B. A. Minch, "Analysis and synthesis of static translinear circuits," Tech. Rep. CSL-TR-2000-1002, Computer Systems Laboratory, Cornell University, Ithaca, NY, USA, 2000.

[11] L. Raffo, "Analysis and synthesis of resistive networks for distributed visual elaborations," *Electronics Letters*, vol. 32, no. 8, pp. 743–744, 1996.

[12] C. Mayr, "Current scaling in current-mode CMOS circuits," in *Proceedings of Dresdner Arbeitstagung Schaltungs- und Systementwurf (DASS '05)*, pp. 91–96, Dresden, Germany, April 2005.

[13] C. Mayr and R. Schueffny, "Image pulse coding scheme applied to feature extraction," in *Proceedings Image and Vision Computing New Zealand (IVCNZ '05)*, pp. 49–54, Dunedin, New Zealand, November 2005.

Christian Mayr received the Dipl.-Ing. (M.S.) degree in electrical engineering (information science and VLSI design) from the University of Technology Dresden, Dresden, Germany, in 2003. Since 2004, he has been a Research Assistant with the Endowed Chair for Neural Circuits and Parallel VLSI Systems, Department of Electrical Engineering, University of Technology Dresden, pursuing a Ph.D. degree in "image processing in pulse-coupled neural networks." Research interests include optimization tools such as genetic algorithms, immune systems on-chip, bioinspired circuits in general, information processing in spiking neural nets in both simulation and hardware, and mixed-signal VLSI design, for example, pixel sensors and CMOS subthreshold circuits. He is the author or coauthor of 14 publications in the subject areas mentioned above and has acted as a reviewer for NIPS conferences.

Andreas König studied electrical engineering, computer architecture, and VLSI design at Darmstadt University of Technology and obtained the Ph.D. degree in 1995 from the same university, Institute of Microelectronic Systems, in the field of neural network application and implementation. In 1995, he joined Fraunhofer-Institute IITB for research on visual inspection and aerial/satellite image processing. In 1996, he was appointed as an Assistant Professor for electronic devices at TU Dresden, where he established a research group on intelligent embedded vision systems. In 2003, he was appointed as a Professor for integrated sensor systems at TU Kaisers-lautern. Research activities of the newly established group are in the field of multisensor system application and integration with a particular focus on the exploitation of bioinspiration for aspects of adaptation, fault tolerance, and learning capability for sensor systems as well as reconfigurable mixed-signal electronics with applications in robust embedded sensor systems and sensor networks. He is a Senior Member of the IEEE, Organizer and Chapter Chair of the German chapter of the IEEE Computational Intelligence Society, Board Member of KES and HIS journals, and Member of IEEE CI, CAS, and ED societies. He is the author or coauthor of more than 100 publications in his research field.