J Real-Time Image Proc (2010) 5:275–289
DOI 10.1007/s11554-009-0148-7

SPECIAL ISSUE

Integrated video motion estimator with Retinex-like pre-processing for robust motion analysis in automotive scenarios: algorithmic and real-time architecture design

Stefano Marsi · Sergio Saponara

Received: 31 July 2009 / Accepted: 23 December 2009 / Published online: 10 February 2010

Abstract The paper presents a novel technique for robust motion analysis in real automotive scenarios based on integrated Retinex-like pre-processing algorithm with block matching video motion estimator. Both algorithmic and real-time hardware design issues are discussed. The benefits of the proposed technique are manifold: the entire system is more robust; the estimated motion vectors are more reliable and less dependent on critical ambient conditions like shadows or flashes; the proposed algorithm may allow to perform motion estimation using very few bits and running as a 2- or 1-bit transform, still maintaining good performances. Real-time hardware implementation is achieved by design and synthesis in 65 nm CMOS standard-cells technology of an Application Specific Instruction-set Processor. Design optimizations for both the processing core and the memory organization are presented. With respect to the state of the art the proposed hardware implementation ensures bounded circuit complexity, low power consumption and reprogrammability of the technique.

Keywords Robust motion analysis · Video processing for automotive applications · Motion estimation · Retinex filters · Application Specific Instruction-set Processor (ASIP) · Hardware for real-time processing

S. Marsi
DEEI, University of Trieste, via Valerio 10, 34127 Trieste, Italy

S. Saponara (✉)
Department of Information Engineering, University of Pisa, via G. Caruso 16, 56122 Pisa, Italy
e-mail: sergio.saponara@iet.unipi.it

1 Introduction

Today, there is a growing interest in using electronic systems for traffic monitoring where images acquired by cameras, installed along the highways or at entrance of a gallery, are processed in real time to analyze their motion content and consequently produce a warning if critical traffic conditions are detected. Similarly, the motion analysis of the scenes acquired by on-board automotive cameras can be used for driver assistance and active safety systems: as example systems for line departure warning, collision avoidance, blind spot warning [1–5].

The core of these systems is the algorithm, and its hardware implementation, for motion estimation and analysis. Due to the stringent requirements of automotive and transport systems such technique should achieve real-time processing, should be compatible with the bounded cost margin and the large volume market of automotive applications, and moreover should be robust to work well also in real road scenarios where illumination conditions are not controlled (flashes, camera saturation phenomena and rapid and sharp luminance variations can occur very frequently).

These requirements are particularly challenging for on-board video acquisition/processing since the adopted cameras, due to limited budget in terms of cost, size, and power consumption typically do not include techniques for adaptive and real-time compensation of changing luminance/contrast conditions.

As consequence a key issue for video-based systems in automotive and transport applications is the realization of algorithms and real-time hardware implementations ensuring a motion analysis robust to environmental luminance variations. Several works have been proposed in literature at algorithmic level which address separately one of the two problems: motion analysis but considering

controlled luminance conditions, or image/video filtering to improve luminance conditions but without motion analysis. The literature is poor concerning integrated techniques covering both tasks and ensuring an optimized implementation of robust motion analysis algorithms and relevant hardware in real-time and with a bounded complexity budget. Moreover, in most cases the considered reference scenarios are consumer photo/video cameras for entertainment and video communications; detailed analysis of real automotive and transport system scenarios are still in a starting phase.

To address the above issues, this work proposes the integrated use of a block matching motion estimator with Retinex-like pre-processing for robust video motion analysis in automotive scenarios. Both algorithmic and hardware architectural aspects are considered. Experimental results using image/video acquired in real road scenarios prove that the proposed technique ensures better coherency of the motion vector (MV) field and increased robustness to variations of the illumination conditions. As far as the hardware implementation is concerned the design of an Application Specific Instruction-set Processor (ASIP) is proposed. The ASIP can ensure both real-time processing and limited complexity, like a custom integrated circuit (ASIC), but still allowing for re-programmability of the used techniques, within the proposed algorithmic class, like a software-programmable DSP.

The rest of the paper is organized as follows. Section 2 reviews the state of the art for robust video motion analysis. Sections 3 and 4 present the algorithm and the ASIP architectural implementation of the new integrated video motion estimator with Retinex-like pre-processing for robust motion analysis in automotive scenarios. CMOS hardware implementation results are discussed in Sect. 5. Conclusions are drawn in Sect. 6.

2 State-of-art of video motion estimators for robust motion analysis

It is well known that canonical motion estimation (ME) algorithms typically suffer from some common drawbacks: they are computationally very expensive and furthermore they lack in robustness when there are irregular lighting conditions. Moreover, it could be noted that their main applications are in video coding, where the similarity between the blocks is more important than the reliability of the estimated MV field. On the contrary, in this paper our target is to find a correct MV field as a first step for further motion analysis algorithms to be employed in intelligent transportation systems.

The effectiveness of the algorithm in the estimation of the correct motion, especially under critical ambient conditions, and its computational complexity unfortunately play opposite roles in the design of the entire system. In [6] for instance a very robust method for motion estimation based on a minimum entropy process has been proposed; however, its complexity makes it suitable only for offline operation. On the contrary, the usage of some well known fast search techniques [7] like the "three step search (TSS)", or the "diamond search" that select a subset of the possible candidate locations could be very appropriate when the target is the video coding. These fast algorithms indeed allow a significant reduction in the computational effort, but on the other hand the simplifications in the searching area often introduce significant inconsistencies in the MVs.

Another approach to the problem could be a suitable preprocessing of the input images, to make them more insensitive to the ambient conditions and thus to improve the effectiveness to the following ME algorithm. In [8–10] the authors propose, like in this paper, a Retinex approach to make the ME algorithm more robust to the brightness variations. However the strategy they adopt to extract the luminance is quite expensive since it is based on a Discrete Cosine Transform; moreover, their goal is in the compression of the video sequence, and the fidelity of the estimated MVs is not a priority. Another application of the pre-processing, often found in the literature, is in the simplification of the motion estimation algorithm itself. In particular, 1-bit [9, 11] and 2-bit [12–14] transform schemes have been proposed to reduce the computational complexity of the motion estimation process by enabling simple Boolean EX-OR matching instead of more complex comparator architectures, with a considerable simplification in the hardware implementation of the system [15, 16].

In this paper, we propose a Retinex-based pre-processing algorithm able to improve the performances of the subsequent motion estimation algorithm. The benefits are manifold: the entire system is more robust, the estimated MVs are more reliable and less dependent on critical ambient conditions like shadows or flashes. Moreover, the proposed algorithm may allow to perform motion estimation using very few bits and running as a 2- or 1-bit transform, still maintaining good performances.

3 Integrated video motion estimator and Retinex-like pre-processing algorithm for robust motion analysis in automotive scenarios

3.1 Algorithm description

It is well known that the human eye is able to efficiently distinguish the details of an image, both when the subject is placed in direct sunlight and when it is in shadow or, even worse, in a dark zone. The human visual system (HVS) is

able also to discriminate between luminance variations due to different illuminators or to shadows present on the scene and luminance variations related to the object details, colors and properties. On the contrary, in electronic vision it is very difficult to discriminate between the two different luminance variation cases. The presence in the processed images of shadows, flashes or of different light sources can alter significantly the automatic perception of the objects motion, and in such critical conditions most motion estimation algorithms lose effectiveness.

To make a motion estimation algorithm much more robust to these critical condition a suitable way could be to preprocess the input images to keep them more insensitive to the possible variations in the scene illumination.

The "Retinex" technique tries indeed to solve these problems through a very simplified but quite effective model of the human vision system [17]. The name "Retinex" itself comes in fact from a contraction of the words "retina" and "cortex" and it stresses that the information obtained by the HVS in the observation of a subject comes from two distinguishable processes: the first is carried out by the retina, that acquires the image, while the second one, operated by the cerebral cortex, concurs to recognize the objects independently of their illumination; for example, we are able to recognize the same object if it is placed both in full sunlight and in a shadow zone or if it is illuminated by an artificial light [18]. This psychophysical phenomenon is often called "brightness/lightness constancy,'' or, more generally, "color constancy''.

The basic concept in the Retinex method is to separate the illumination and reflectance components of an acquired image. It is assumed that the available luminance data is the product of an illumination and a reflectance, and the values of the latter are determined as the ratio between the luminance and an estimation of the illumination.

In such a way it is possible to process independently the illumination and the reflectance components and to extract the information we need. We will demonstrate that the usage of the reflectance data to perform motion estimation is more suitable and more robust when some critical conditions, like flashes or shadows, are present in the scene.

The core of the proposed algorithm is in a reliable estimation of the illumination. Various methods have been proposed in the literature in order to perform this task. All these techniques are based on the fact that the illumination signal is presumed to change quite smoothly in the parts of the image illuminated from the same luminous source, but it can also present abrupt variations when the scene is illuminated by different light sources (for example, when a part of the scene is illuminated by direct sunlight while other parts are in shadow).

In order to obtain an adequate estimation of the illumination, the various methods up to now proposed in the literature are quite complicated: in fact they use either algorithms based on multi-resolution techniques [18, 19] or iterative methods which need to be applied several times to the same image [20]. In all these cases the algorithms are so complex that they result absolutely unsuitable in real-time applications.

In this article, we use a quite simple algorithm for the extraction of the illumination. This algorithm, mainly based on rational filters, is based on the idea of realizing the filter in a recursive way. The employment of this typology of filters allows to obtain very good results with small computational effort, suitable for real-time applications [21].

The core of the entire system resides in the algorithm for the illumination estimation, which separates the two fundamental components of illumination and reflectance in the input images. Intuitively it can be accepted that in a natural image the illumination typically changes very smoothly between contiguous pixels, with the exception of some particular cases, like the presence of luminous sources within the scene, or transitions among various light sources. A good estimator of the illumination must take into account all these cases. Consequently, the goal is to realize an edge-preserving (for abrupt transitions) low-pass (for smooth transitions) filter with a narrow pass-band. The latter obviously implies a wide impulse response of the filter, and is the primary reason for the high computational complexity of the algorithms which can be found in the literature. Indeed, a wide impulse response requires the use of either extremely large masks, or quadtree decomposition techniques or, alternatively, repeated applications of a simpler operator. All these solutions conflict with real-time processing requirements.

To combine both the narrow-band request and the adaptivity of the system, a particular typology of operator, i.e., the recursive rational filter (RRF), is considered here. Mainly based on the structure of the rational filters [22], this algorithm introduces an innovative aspect, the spatial recursivity of the operator. This characteristic is fundamental to obtain narrow bands using only few input taps.

Let us consider a simple first order IIR low-pass filter with a unitary gain for $\omega = 0$; it can be expressed by the following equation:

$$y(n) = (1 - a)x(n) + ay(n - 1) \tag{1}$$

where $x(n)$ and $y(n)$ are respectively the input and output samples of the filter, while the a coefficient assumes values between 0 and 1 and defines the filter bandwidth: when a is close to 1 the pass-band is narrow, viceversa when a is close to 0 the filter tends to become "transparent" with $y(n) = x(n)$.

Now, remembering that the goal is to obtain an edge-preserving low-pass filter, we can take advantage of the characteristic just exposed in order to make the filter in (1)

adaptive. To obtain the adaptability of the filter an expression is introduced which locally modifies the value of a according to the characteristics of the input signal. Such function makes a assume a value next to 0 when the input data shows abrupt transitions, and makes it increase up to a suitable maximum value when the input signal is smoother. The proposed function is the following:

$$a = \alpha \frac{S}{S+1} \tag{2}$$

where

$$S = \frac{H}{\left(\log \frac{\varepsilon + x(n-1)}{\varepsilon + x(n+1)}\right)^2 + \delta} \tag{3}$$

where α is the maximum value that the a coefficient can assume, and sets the minimal bandwidth of the filter; S is an appropriate sensor, suitable to locate the presence of edges: when $x(n-1)$ and $x(n+1)$ have similar values, S will become large; on the opposite the value of S decreases considerably in presence of a sharp edge, i.e., when $x(n-1)$ and $x(n+1)$ assume very dissimilar values. Finally, H permits to set the intensity of the sensor response. Both ε and δ are small values used in order to avoid to perform the log operation on inconsistent data and that the denominator becomes zero.

It can be noticed that following the Retinex theory the sensor has been designed to be sensitive to the relative transitions of the signal [estimated by a division between $x(n-1)$ and $x(n+1)$] rather than to the absolute transitions [that could be obtained with the difference operation between $x(n-1)$ and $x(n+1)$].

The described algorithm is extended in two dimensions in a separable form. Two similar sensors S_h and S_v are used to estimate respectively the abrupt horizontal and vertical transitions in the signal, where obviously n and m represent the spatial coordinates of the two-dimensional signal:

$$S_\mathrm{h} = \frac{H}{\left(\log \frac{\varepsilon + x(n,m-1)}{\varepsilon + x(n,m+1)}\right)^2 + \delta}, \tag{4}$$

$$S_\mathrm{v} = \frac{H}{\left(\log \frac{\varepsilon + x(n-1,m)}{\varepsilon + x(n+1,m)}\right)^2 + \delta} \tag{5}$$

The values of S_h and S_v are then applied to the filtering operator

$$y(n,m) = \frac{\alpha[y(n-1,m)S_\mathrm{v} + y(n,m-1)S_\mathrm{h}] + [(S_\mathrm{v}+S_\mathrm{h})(1-\alpha)+1]x(n,m)}{S_\mathrm{v}+S_\mathrm{h}+\varepsilon} \tag{6}$$

It can be noted that in uniform areas S_h and S_v assume large values and the effect of (6) is very similar to the one of a bidirectional IIR low-pass filter. On the contrary, along the edges the value of one or of both sensors (depending on the edge direction) decreases and the contribution along the perpendicular direction in (6) is limited.

The spatial recursivity of the algorithm is very effective to obtain a narrow-band low-pass effect but presents the drawback of introducing a phase distortion in the output image. We equalize such distortion by using a two-dimensional extension of the well known time-reversal method [23]. For such reason the filter must be applied four times to each input image. During these iterations the pixels are processed along all the possible directions from top to bottom and from left to right and viceversa.

Since the Retinex theory has been developed to discriminate between the luminance changes due to alterations in the illumination (flash or shadows in the scene) and the luminance changes related to the intrinsic properties of the objects, we have employed this method to improve the performances of motion estimation algorithms and to make them more insensitive to critical conditions. The input frames $I_i(x, y)$ are processed one by one using (4–6) to obtain a luminance estimation $L_i(x, y)$, then a pixel by pixel division provides an estimate of the reflectance information $R_i(x, y)$:

$$R_i(x,y) = \frac{I_i(x,y)}{L_i(x,y)} \tag{7}$$

This signal has a unitary mean value, and we found convenient to map it using a suitable nonlinear function as follows:

$$R'_i(x,y) = K \log(R_i(x,y)) + 0.5 \tag{8}$$

where K is a suitable gain which together with an offset of 0.5 transforms the input data in an image with all positive pixel values in the range 0–1. Eventually the logarithmic mapping is suitable to make the division operation indicated in (7) realizable by a subtraction operator, which is very simple to implement.

$$R'_i(x,y) = K[\log(I_i(x,y)) - \log(L_i(x,y))] + 0.5 \tag{9}$$

It could be noted that dissimilarly to other algorithms recently presented in the literature [21, 24, 25], in this case the algorithm is used just to estimate the reflectance. In fact when the aim is to improve the visual quality of the bad illuminated image, it is important not to discard completely the luminance information, and a suitable processing of both reflectance and luminance signals followed by a recombination can lead to very pleasant result. On the contrary in this paper the priority is to improve the fidelity of the MV field obtained through a ME algorithm also when the input images are acquired under critical conditions. We have proved that for such application the reflectance signal maintains all the information needed for

a good ME, and moreover the algorithm results simpler and less sensible to luminance variations.

The images so pre-processed are now ready to be processed by a block matching motion estimation algorithm. In Fig. 1, we present an example of an input image, a SIF (320 × 240 pixels) frame acquired in a real road scenario, its luminance estimation and the result of the proposed algorithm. It could be noted that all the details both inside and outside the tunnel are more distinguishable, independently of the local illumination. The algorithm has been tuned with the following principal parameters: $\alpha = 0.98$, $K = 2$, $H = 1e{-}3$, however, the results do not change significantly if variations in the parameters are provided. These parameters have been constantly maintained for all the experiments illustrated in this paper. In particular with α it can be defined the size of the details to be emphasized (thus higher values can be suggested when using larger image size), K tunes the emphasis level and H controls the behavior in the presence of sharp edges. Besides the images showed in this paper, we have made several experiments applying the proposed algorithm to different images acquired from real road scenarios. In our experiments, we have also tried to modify the parameters into a certain variation range $0.8 < \alpha < 0.999$, $1 < K < 4$ and $1e{-}4 < H < 1e{-}2$ finding that the choice is not critical for the performances of the following ME algorithm. The results achieved in the experimental test campaign confirm that the performances of the proposed technique, showed for the example images in Figs. 1, 2, and 3, are also valid for the other test images. The camera used is a low cost consumer electronic camera with auto-iris compensation.

3.2 Improving the motion estimation robustness to luminance variations

Using the proposed technique as a pre-processing, the next step is to verify the advantages of this technique when coupled with a motion estimation algorithm.

To avoid the problem of local minima and to perform our experiments in a standard, basic ME setting, we have used a full-search (FS) algorithm which operates with 16 × 16 blocks, one previous frame used for matching, and a scanning area of ±16 pixels in both horizontal and vertical directions. It has been noted through several tests that, in non-critical conditions, the result obtainable in terms of MVs from a pre-processed sequence compared to the original one typically coincide. On the contrary when some variations in the scene illumination appear, the benefits of the pre-processing become evident. For example, Fig. 2 presents a couple of frames from a sequence showing a car entering a tunnel (30 frames/s SIF video format). The camera is accomplishing a horizontal panning to follow the car; however, due to prospective, pixels representing the back of the car move faster than those in the front part.

The problem with such frames is twofold: the former is due to the high dynamic of the scene that makes difficult a good discrimination of the details both in the well-illuminated areas and in the dark ones. The latter is related to a barely visible adaptation of the auto-iris of the camera that, moving from a well-illuminated zone to a darker one, tries to compensate the reduction in the global illumination with a small opening of the iris. It can be noted that the proposed pre-processing algorithm is able to compensate for both problems and allows the estimation of a more realistic MV field. Another example of the benefits achieved using the proposed technique is shown in Fig. 3. In such a case the problems are related to the non uniform shadows of the tree leafs which are present both on the scene and on the camera lens. Without the pre-processing algorithm the final result suffers from many poorly estimated MVs. On the contrary, using the proposed technique the details are easier to discriminate and the motion estimation vectors become more accurate.

3.3 Robustness to quantization

Another way to exploit the advantages of the pre-processing algorithm resides in the possibility to simplify the realization of the subsequent motion estimation algorithm. In particular, we prove that using the proposed algorithm it is possible to obtain good motion estimation results even if the latter is accomplished using only a few bits. In such a way the cost versus performances ratio of the entire system could significantly decrease.

To test this possibility, we have first realized a ground truth test bench: using two identical frames, we have processed one of them to emulate a predefined MV field. Than we have used this pair of frames to check our system

Fig. 1 From *left* to *right*: the input image acquired by the camera, the luminance estimation and the reflectance signal obtained via (9)

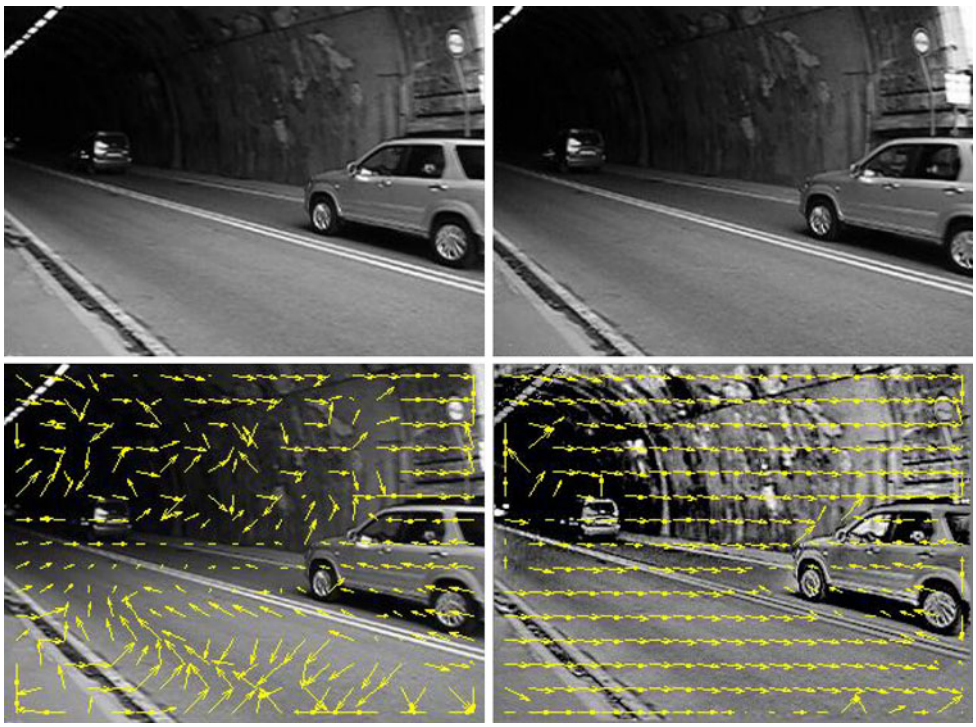

Fig. 2 *Up*: two adjacent frames in the test input sequence. The input sequence suffers for the presence of slight auto-iris compensation and a high dynamic range. *Down left* the MV field obtained without preprocessing and on the right using the proposed algorithm to pre-process the input images

Fig. 3 *Up*: two adjacent frames in the test input sequence. The input images suffer for the presence of shadows both on the street and on the camera lens. *Down left* the motion vector field obtained without preprocessing and on the right using the proposed algorithm to pre-process the input sequence

composed by the cascade of three blocks: the pre-processing Retinex-based algorithm, a quantizer and the full-search motion estimation algorithm.

Using a variable number of bits from 1 to 8 in the quantizer, we have evaluated the mean absolute error between the MV field obtained by the system and the ground truth.

The result depicted in Fig. 4 demonstrates that, even there are no significant advantages in the 4–8 bits range, the usage of the pre-processing algorithm permits to obtain a considerable smaller estimation error particularly when the quantization is very strong. In an extreme case, it is possible to perform the motion estimation, admitting quite a small error, using data of just one bit, i.e., binarizing the

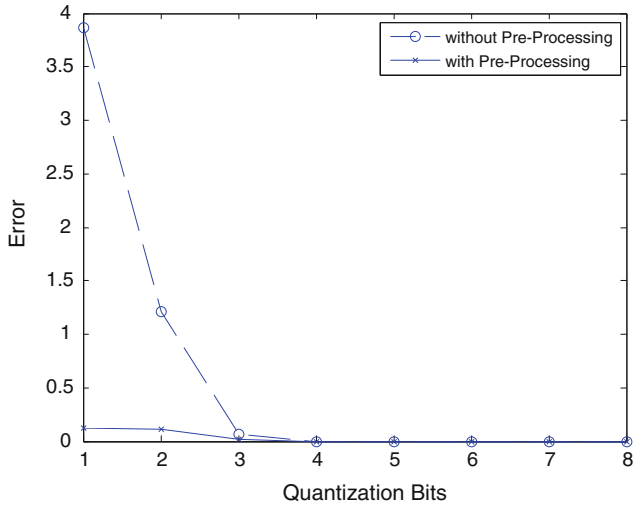

Fig. 4 Mean absolute error (between the MV fields) performed by the ME algorithm on a ground truth test bench for different quantization levels, with (*solid line*) and without (*dashed line*) pre-processing algorithm

input images. Following this strategy [11–16] the ME algorithm which is typically the bottleneck and the most computationally expensive block of the entire system could be realized using very simple operators where for instance the comparators could be substituted by XOR gates.

3.4 Analysis of different me engines to be integrated with the Retinex pre-processor

To test the advantages of the proposed algorithm, we evaluate its benefits also when interfaced to other kind of ME algorithms. In particular, we evaluate the effectiveness of the MV field obtained using three different algorithms: the FS, the TSS, and the adaptive rood pattern search (ARPS) [7], when the input images are, or are not, pre-processed.

To create an objective test bench, it is mandatory to define a reference target, i.e., a ME algorithm able to accomplish good performances even under critical conditions. We have chosen for such a purpose an enhanced FS algorithm, insensitive to the offset of the block [26]. This algorithm works very similarly to a canonical FS-ME but each comparison among the blocks is firstly purged by the offset of the blocks themselves; in such a way the FS algorithm becomes more robust to light variations and the estimated MV field becomes more reliable. It could be noted, however, that this improvement on the canonical FS algorithm, even if quite simple from a conceptual point of view, leads to a significant extra computational effort, equivalent to the one of the entire FS-ME algorithm. In fact the evaluation of the offset for each block requires one extra addition of all the pixel values in the block itself. If no particular strategies are adopted (such as pre-computing the mean values for all the possible image blocks and storing them in a temporary memory, which, however, would significantly increase the complexity in the final realization), the number of addition/subtraction operations to be performed for each block will double.

Moreover, it is also important to underline that such an algorithm, even though its performances are very good under many testing conditions, can however occasionally fail.

Adopting the previously described improved FS algorithm as reference target, we have evaluated the distance between the MV fields estimated by the algorithms under test with and without the pre-processing phase, and the target one. The metric adopted to evaluate the distance between the two MV fields is the mean absolute value of all the difference vectors. The results obtained analyzing a 20-frame input sequence (a couple of sequence frames are depicted in Fig. 3) using four different algorithms, with and without the benefit of the pre-processing algorithm, is depicted in Fig. 5.

It can be noted that the presence of the pre-processing leads in all cases to better performances, which are particularly evident when the presence of critical conditions compromises canonical algorithms performances. Moreover, it can also be noted that the pre-processing together with an adaptive ME algorithm like the ARPS method can significantly outperform the FS without pre-processing. Moreover, the performances of the TSS algorithm, due to the problem of the local minima, still remains quite poor in both cases and its application seem to be not suitable for the proposed goal of the reliability of the MV.

An extra consideration: it can be noticed that in Fig. 5 the ARPS algorithms often performs better than the FS. To motivate this behavior, it must be underlined that the FS does not look for the best MV in the sense of the actual block motion, but in the sense of the most similar candidate block. This behavior can generate large differences especially in critical conditions when luminance variations are present. In such critical cases the ARPS algorithm, which bases the research of the MV candidate using the MV information of the other nearest blocks, produces a more coherent MV field and a better representation of the actual movement. As a consequence, its error with respect to the hypothesized real motion field can be smaller than the one provided by the FS algorithm.

4 Application Specific Instruction-set Processor for real-time integrated video motion estimation and Retinex-like pre-processing

4.1 ASIP design approach and definition of processing operators and machine arithmetic

The real-time hardware implementation of the proposed algorithms for robust motion analysis is based on the

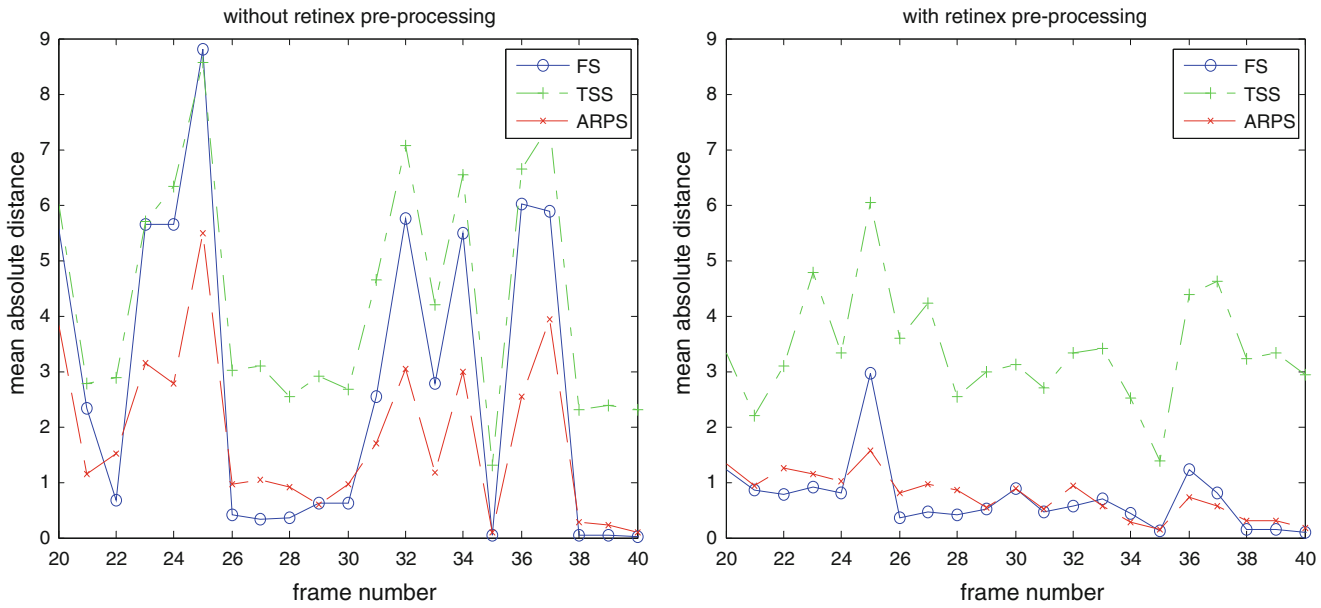

Fig. 5 Performances of three different ME algorithms: full search (*FS*), three step search (*TSS*) and adaptive rood pattern search (*ARPS*) for an image sequence without (*left*) and with (*right*) the pre-processing algorithm

design, supported by languages for processor architecture description such as LISAtek [27, 28], and synthesis on standard-cell CMOS technologies of an ASIP.

As already proved in literature [28–38] for hardware implementation of a specific class of digital signal processing algorithms, including video and image processing as discussed in our previous works [24, 25, 39], ASIPs offer a better trade-off between flexibility and performance versus classic ASICs and DSPs solutions. By exploiting the ASIPs paradigm the designer can achieve flexibility within an application domain, being able to accomplish a group of functionalities using a set of common operations. The most frequently repeated application kernels can be grouped in optimized hardware units, keeping their activation at a software level and hence allowing for different programmable versions of a class of algorithms. While designing ASIPs, it is a designer duty to trade-off algorithmic performance, flexibility and physical criteria like area and power consumption. In the literature ASIPs specific for video motion estimation only or for Retinex only already exist [30, 36, 39]. Hence a straightforward architecture solution, to support in real-time the algorithm presented in Sect. 3, could be simply assembling separated ASIP cores, available as reusable IP macrocells from our previous works [24, 25, 39], in a single system-on-chip through an on-chip communication infrastructure based, as example, on the well known AMBA bus. The limit of such solution is that an unnecessary hardware overhead in terms of area and leakage power (no more negligible in sub-micron CMOS technologies below 100 nm) has to be paid. With respect to our previous ASIP Retinex design [24, 25] in this work we:

1. extend the instruction set to support with a single flexible and more optimized ASIP core both motion estimation and Retinex filtering functions;
2. optimize the hardware resources avoiding computation units, e.g., such those for color conversion and gamma luminance correction, not needed since the algorithm in Sect. 3 refers to monochrome frames only and just uses the reflectance information, see (8) and (9).

Designing a customized instruction-set for integrated motion estimation and Retinex filtering requires a preliminary optimization phase to define the machine arithmetic accuracy and the way to implement the required operators. These operators include, besides a Multiply and Accumulate unit, several nonlinear processing tasks: for motion estimation there are the minimum search (to evaluate the best matching block and its cost function and MV coordinates), the threshold comparison (for early stop search criteria), the cost function evaluation (absolute difference, SAD, or mean square error, MSE). For Retinex processing there is the evaluation of logarithms, division operators and square operators used in (2–9). Starting from the linearized and bit-true algorithmic model the design of an ASIP is then addressed as detailed in next sections. For the design, we used the architecture design language LISAtek from which the VHDL description is automatically generated and then manually refined before synthesis on CMOS standard-cells technology.

To address the above issues the class of algorithms described in Sect. 3 has been modeled through a parametric linearized structural model, with finite arithmetic precision. The linearization of non-linear operators has been realized using some predefined optimization schemes based on piecewise linear and piecewise constant representations. High-order polynomial piecewise approximations are not reported in this paper, since during the exploration phase they resulted in too high implementation complexity while, from the algorithmic point of view, optimal performance can already be achieved with the piecewise linear and piecewise constant approaches. By changing the model parameters a trade-off analysis has been realized to find the most suited arithmetic finite precision and implementation of the non-linear operators. The trade-off is played on the piecewise approach, constant or linear, and on the number of used edges and their distribution along the abscissa axis. Two criteria are used to drive the optimization process: a PSNR-based objective criterion and a subjective one based on visual perception. The target is reducing the implementation complexity of the proposed technique while keeping the same performances shown in Sect. 3.

Figures 6, 7 and 8 show the hardware implementation block diagrams for linear-piecewise operator, for constant piecewise operator and for absolute-difference operator.

In Figs. 6 and 7 first the input sample *In* is compared to proper thresholds in order to identify the correct approximation interval; the results are then used to address a look-up table (LUT) storing the parameters of the piecewise segment: (offset Q, slope H) for the piecewise linear, directly the output for the piecewise constant. In case of the piecewise linear an extra multiplier unit is required to calculate the output $H \times In + Q$. Figure 8 presents the hardware unit [40] dedicated to absolute difference (AD) and accumulation operations (used during motion estimation).

As a results of the analysis the piecewise constant solution leads to a better trade-off between hardware complexity and visual quality when the non-linear transformation to be linearized involves quantities which are not directly observable at the system output, like the coefficients of the RRF for illumination estimation depending on S_h and S_v. In such a case the visual quality using a

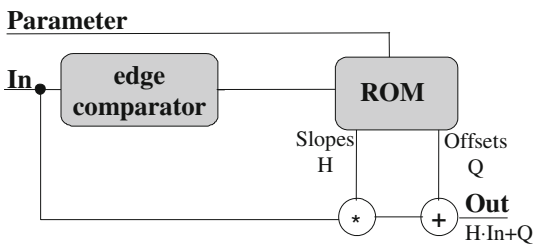

Fig. 6 Implementation block diagrams for piecewise linear approximation of non-linear operators

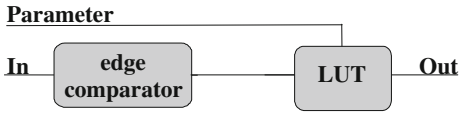

Fig. 7 Implementation block diagrams for piecewise constant approximation of non-linear operators

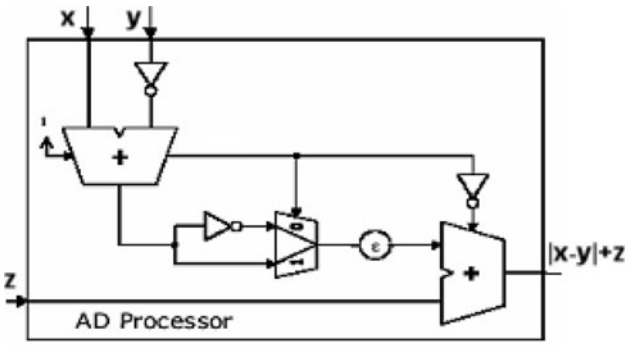

Fig. 8 Combinatorial part of the AD processing element

piecewise constant approximation is the same as using a linear one, but the former is simpler since it is based on a LUT and avoids using multipliers (compare Figs. 6, 7).

On the contrary the piecewise linear approach shows better quality results and is adopted only to approximate the reflectance processing operator in (9). The piecewise linear approach has been also used for the division operation.

To be noted that the edges of the approximating segments for both piecewise linear and piecewise constant operators, determining in Figs. 6 and 7 the thresholds of the edge comparator and the slopes and offset data to be stored in the ROM, have been selected taking into account the statistical distribution of the input samples of the operators under linearization. Several edge distribution laws have been compared to find the best trade-off between circuit complexity and algorithmic accuracy. The latter has been measured with both a PSNR-based objective criterion and a visual perception subjective one. Particularly, in this work, we have considered linear and non-linear (exponential, quadratic, cubic, and fourth order polynomial) distribution laws. Among them the cubic distribution law leads to the best trade-off.

After the linearization of the operator, the number of precision bits for data representation has been specifically tailored to reduce the memory requirement and the ASIP circuit complexity. In the case study, we found that fixed-point data representation for the hardware implementation can ensure unnoticeable algorithmic performance reduction (measured both by PSNR and visual subjective criteria considering several test images as input stimuli) versus the original algorithm in Sect. 3. Particularly 8-bit size can be

used for input pixel representation and 12-bit size can be used for motion estimation and Retinex processing data paths. Particularly for motion estimation cost function, represented on 12 bits, a saturation technique has been implemented: if the cost function (e.g., SAD) exceeds the 12-bit representation it means that the matching block is not the optimal one and hence the SAD result is saturated (it will be discarded during the final SAD minimum search).

Besides the instruction set dedicated to processing tasks, the proposed ASIP will include dedicated resources for other tasks mainly related to memory data transfer management, control and scheduling of the operations to be performed and of the data flow (e.g., threshold comparisons and minimum detection operations for motion estimation, especially if a predictive algorithm with a first coarse block matching analysis followed by a refinement iterative search is adopted). This way different algorithmic configurations of the Retinex pre-processing filter and of the motion estimation (FS, but also adaptive predictive search to reduce the computational complexity while keeping good algorithmic performances) are supported in hardware.

As example, considering the address generation for the data memory, from the specifications of Retinex and motion estimation algorithms it can be noticed that some predetermined patterns are established iterating over the image. The specific pattern to be uploaded from memory depends on the specific algorithm. Thus, a Programmable Address Generation Unit (PAGU) calculating the next address for the data memory by incrementing the pixel pointer is implemented in hardware. This automatic address calculation feature is reflected in the syntax of several instructions by a short extension. Thus, the address update is performed in parallel without the need of wasting cycles just to update the data address.

Another observation is that in conventional loop implementations comparisons and conditional branches create a significant instruction overhead and, even worse, cause pipeline control hazards. They lead to pipeline stalls and flushes. These problems can be avoided by implementing a loop mechanism in hardware. This is possible for loops being executed a predictable number of times. In this case, it is sufficient to have a loop-parameter initialization before entering the loop and to manage the loop jumps by the hardware. This technique is known as zero-overhead loop. With these implementation strategies, the programming is made easier and pipeline stalls and flushes resulting from control hazards can be eliminated.

Summarizing, the ASIP proposed in this work shares part of the set of 42 instructions, detailed in [24], with a previously designed ASIP for Retinex filtering of color images/video sequences in consumer applications. The shared instructions belong to the following groups: non-linear transformations, arithmetical computations, memory accesses, processor initialization, data flow and loop control. With respect to the instruction-set in [24] the current ASIP design, as described above, avoids computational units for space color conversion and gamma luminance correction while introduces new instructions dedicated to ME processing: minimum search to evaluate the best matching block and its cost function and the MV coordinates; threshold comparison for early stop search criteria; cost function evaluation.

4.2 Pipelined architecture

The ASIP proposed for this case study is based on a seven-stage pipeline architecture sketched in Fig. 9. The names assigned to each stage are mnemonical names applying to the presented particular case. Depending on the instruction, different operations can be executed in the same stage. As described hereafter the operation of the first three stages, FE, DC and LD, are independent from the specific operating part of the currently implemented algorithm while the operations realized in the other stages (Exe1, Exe2, Exe3 and WB) depend on which algorithm is actually processed. The selection of the pipeline architecture in Fig. 9 is motivated by the fact that this ASIP is an evolution, targeting both Retinex and ME algorithms, of the previous Retinex-ASIP, we proposed in [24]. As we already detailed in [24] the seven-stage pipeline architecture is needed to implement a *Single instruction non-linear transformations*, i.e., a single instruction able to load an operand from the data memory, to perform the luminance or reflectance transformations in the piecewise linearized form, and to store the result back to the data memory. The ASIP design in this paper and that in [24] share part of their set of instructions, as described in Sect. 4.1; therefore the adoption of the seven-stage pipeline hardware architecture has been confirmed as starting point for the new ASIP development in this work.

Hereafter, we briefly describe the role of each stage:

- *FE* is the fetch stage in which the instruction is fetched from the program memory.

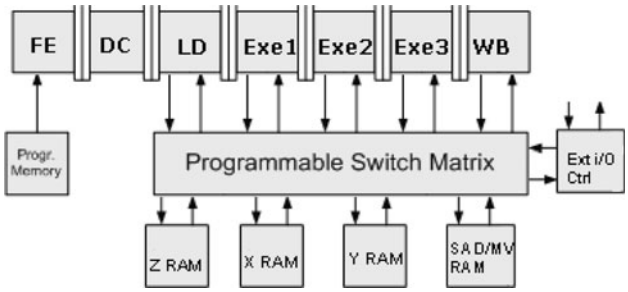

Fig. 9 Pipeline and memory organizations for the ASIP

- *DC* is the decode stage where the instruction is decoded producing the control signals for the operating part.
- *LD* is the load stage in which the operand is loaded from the data memory; since the motion estimation task requires a much higher computation capability than the Retinex one, in case of ME 4 pixels will be processed in parallel (a 32-bit operand is loaded from memory).
- *Exe1* is the comparison stage where, during Retinex processing through piecewise linearized operators, the loaded operand is compared to the edges on the abscissa axis to identify the correct approximation interval; during motion estimation processing instead four pixels (a 32-bit operand) from the previous frame memory are loaded and the absolute differences with the current pixels are calculated (4 absolute difference calculated in parallel using an array of 4 processing elements) and accumulated with previous calculated ones.
- *Exe2* is the stage where, during Retinex processing, the result of the previous comparison is used to address a ROM storing the parameters (offset Q, slope H) of the correct piecewise segment or directly the LUT containing the output result for piecewise constant operators; during motion estimation implementation the same operations of Exe1 are implemented for other four couples of pixels belonging to the current image block and its relevant candidate matching block from the previous frame.
- *Exe3* is the stage in which the fetched ROM parameters are used to calculate the output according to the piecewise segment expression $H \times IN + Q$; such stage is by-passed if a piecewise constant operator has to be implemented. During motion estimation an adder tree adds up the four absolute-difference results coming from Exe2 stage. Moreover, if the SAD relevant to a block matching operation is terminated, it is compared to the previous evaluated SAD_{min} to detect the minimum SAD value and the corresponding MV coordinates. In case a predictive search algorithm is implemented, the evaluated cost function is compared to proper thresholds for early search termination criteria.
- *WB* is the write-back stage in which the output is stored back to memory: a data frame memory if the Retinex operator is the one currently computed and the result is a 8-bit filtered pixel; the SAD/MV memory if the ME operator is the one currently computed and the result is a 12-bit SADmin value plus relevant MV coordinates stored in ten bits [displacement in the range $(-16, +15]$ pixel along both x and y axis).

Overall considering the proposed architecture is able to process for each clock cycle a maximum (without pipeline stalls or flushes) of eight absolute-difference operations at pixel level during motion estimation processing and one complete non-linear operator at pixel level for Retinex processing.

4.3 Memory organization and by-passes

Besides pipeline stage definition, for the ASIP is also important defining the memory organization since multimedia applications are memory dominated. Both the particular memory organization and the data pipelining are important ASIP hardware customizations applying to the case study.

For motion estimation and for the iterative applications of the RRF inside the Retinex, it is necessary to store at least one previous frame. It can be also the case that the video processing is split over several pipeline stages, by the means of frame pipelining, to increase the throughput. In such a case the memory size of a single frame has to be multiplied by the number of used pipeline stages. To reduce the memory amount one way is pursued by reducing the number of precision bits (see Sect. 4.1). Another effective way to reduce memory is to remove the pipelining at a frame level. This solution is based on a re-utilization of the same memory to store the intermediate data concerning the partially processed frames. The main drawback is, of course, the throughput reduction, which is a critical specification item. To improve timing performance while keeping the benefits of a memory organization which minimizes the use of large frame memories, pipeline processing can be re-introduced moving it from the frame level to the pixel level. Therefore, pixel-level pipelining has been implemented in the ASIP architecture proposed in Fig. 9 since it is more efficient in terms of memory usage than frame-level pipelining. This permits the parallel processing of several pixels making the architecture timing efficient as well.

Because of the trade-off between memory resources and data throughput, we decided to use three frame memories: synchronous SRAM named X, Y and Z RAM in Fig. 9.

This solution allowed us to keep a good degree of parallelism, since it is possible, for instance, to perform the operations needed for the Retinex on luminance and reflectance at the same time, without increasing the memory requirements too much. Indeed, at least two frame memories (the X and Y RAM) are needed during motion estimation to store the current frame and the previous frame from which candidate matching blocks are uploaded. The third frame memory (Z RAM) is needed to allow correct interleaving of Retinex and motion estimation processing tasks when realizing the integrated robust motion analysis algorithm proposed in Sect. 3.

When starting the analysis of an acquired input video the first frame I is filtered with Retinex operator (using X and Y RAM for iterative applications of the RRF for luminance estimation) and the final result I^* in stored in the Z RAM. Then the subsequent input frame $I + 1$ is processed using X and Y RAM and storing the filtered frame $I^* + 1$ in Y RAM. After Retinex processing, the motion estimation is implemented using $I^* + 1$, stored in Y RAM, as current reference frame and I^*, stored in Z RAM, as previous frame. The results of the motion analysis are stored in the SAD/MV RAM while concurrently the next frame to be filtered $I + 2$ is pre-fetched in the X RAM. After motion analysis the frame $I + 2$ is filtered using X and Z RAM (Y RAM still stores the frame $I^* + 1$ that will be used for motion analysis while the frame I^* can be deleted since it is no more needed) and the final results $I^* + 2$ stored in Z RAM. Subsequently the motion estimation is implemented using $I^* + 2$, stored in Z RAM, as current reference frame and $I + 1^*$, stored in Y RAM, as previous frame. The results of the motion analysis are stored in the SAD/MV RAM while concurrently the next frame to be filtered $I + 3$ is pre-fetched in the X RAM. The processing flow then continues alternating Retinex and motion estimation phases.

Besides X, Y and Z RAM the memory organization is completed with a program memory from which program instructions are fetched, and a motion estimation results memory where SADmin and MV coordinates are stored for each 16×16 image block of the frame which is currently processed.

In other ASIP architectures proposed in literature, e.g. [24, 25], the connections between pipeline stages and memories were fixed: the data RAM can be accessed in read mode or write mode only from certain stages. Using fixed connections can cause data transfer overheads in case of new algorithmic versions needing direct memory access from other stages. However, designing multi-port RAM concurrently accessible from all execution stages is too expensive in terms of area and power. Such problem is solved in the proposed ASIP architecture in Fig. 9 connecting the memory resources and the pipeline ASIP stages through a configurable interconnect matrix (see programmable switch matrix in Fig. 9). Hence, by proper programming an array of MOS switch through a context configuration memory the desired coupling between execution stages and memories can be customized after post-silicon fabrication.

A problem related to the pipeline architecture proposed in Fig. 9 is data dependencies. The seven-stage pipeline leads to the disadvantage of data hazards: e.g., in the LD stage an instruction (consumer) may read from a shared storage, a general purpose register or a memory location, which is expected to be written by a previous instruction

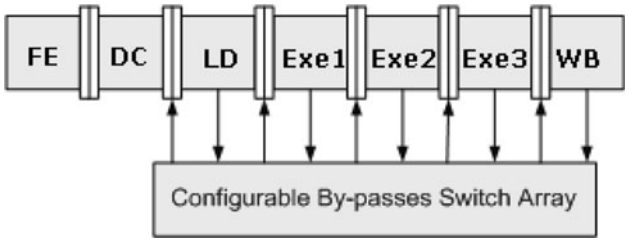

Fig. 10 Implemented by-pass mechanisms

(producer). If the producer instruction has not yet reached the WB stage in which the final result is stored in the shared storage, the consumer instruction will load an outdated value. Data dependencies can be solved efficiently through bypasses which forward data immediately from a pipeline stage back to a previous stage. In previous ASIP versions for Retinex processing proposed in literature [24, 25] fixed bypasses are implemented only from the last two stages to the relevant preceding stages. This approach can limit the efficiency of the ASIP for algorithms needing bypasses from other stages. As reported in Fig. 10 to overcome such problem in the new ASIP architecture proposed in this work there is a configurable interconnect switch matrix allowing for post-fabrication customization of the bypasses between the different execution stages of the pipeline. As discussed above the configuration mechanism is based on writing the desired configuration bit plan in a context configuration memory.

To be noted that when using configurable connections for by-passes, see Fig. 10, and for memory accesses, see Fig. 9, the increased flexibility (i.e., with respect to the proposed algorithm in Sect. 3 the ASIP can be easily updated to support future new algorithmic versions requiring different schemes for by-passes and for direct memory access) is paid with a small overhead in terms of area and increased time propagation delay. However, given the good performance results achieved in Sect. 5 when implementing the architecture in standard-cells CMOS technology, in this work, we preferred increasing the ASIP architecture flexibility adopting the configurable connection schemes in Figs. 9 and 10.

5 CMOS implementation results and performance

After the design of the architecture in Sect. 4 with LISAtek, a parametric VHDL description of the ASIP has been automatically generated and then manually refined for those paths representing a bottleneck in terms of timing. The VHDL data base, configured according to the machine arithmetic sizing reported in Sect. 4.1, has been synthesized with the Synopsys CAD tool in a 65 nm 1.1 V supply voltage nine metal layers CMOS standard-cells technology.

To be noted that for the CMOS technology three library versions were available: beside the standard library nominal voltage threshold (NVT) there are also the high voltage threshold (HVT), optimized for low power consumption, and particularly for low leakage power) and the LVT (low voltage threshold, optimized for high speed). In this work, we used the HVT library version to minimize the power consumption. As detailed hereafter the achieved timing performance with HVT still meet the real-time requirements of automotive scenarios.

The ASIP processing core, synthesized in the above mentioned CMOS technology, has a complexity of roughly 105 kgates plus 15 kB of ROM to implement LUT-based operators. The static power consumption (leakage power) amounts to roughly 25 μW. Such results are very interesting when compared to the alternative solution of assembling together two separate ASIPs, already available in literature, e.g., in [39], one dedicated to Retinex processing and one to motion estimation. This second case would require a circuit complexity of at least 215 kgates plus 20 kB of ROM plus extra resources for communication and coordination of the two ASIP macrocells.

The requirements in terms of program and data memory (X, Y and Z data SRAM plus SAD/MV results SRAM) for our design are reported in Table 1 for different formats which are typical for cameras used in automotive and transport system applications. To be noted that the algorithmic results reported in Sect. 3 referred to SIF format.

The required memory from Table 1 can be easily implemented on-chip, for all considered formats, using the memory blocks provided by the used 65 nm standard-cell technology. Since on-chip RAM storage is used for the program memory, at board-level an external non-volatile programmable ROM is needed from which each time the ASIP chip is powered-on the program to be executed should be uploaded.

The maximum achieved clock frequency for the proposed ASIP in the CMOS 65 nm—HVT technology at 1.1 V is 300 MHz. This allows for a maximum throughput of about 2,400 millions of absolute-difference operations per second if using the ASIP only for motion estimation processing. Such performance enables real-time calculation of motion estimation for all the considered image formats in Table 1 at 30 frame/s, using different search algorithms including the worst case of FS (through the paper monochrome images and 16-pixel search displacement are considered). When using the ASIP only for Retinex processing the computational throughput amounts to roughly 9 million of pixels per second since, in average, using the Retinex algorithm in Sect. 3 roughly 30 clock cycles are needed for a single pixel filtering. Hence the ASIP can be used to process in 1 s large still images or in real-time VGA videos up to 30 frames/s.

Table 1 RAM needed for the ASIP to process various formats

	QCIF (176 × 144)	SIF (320 × 240)	VGA (640 × 480)
Program memory	1 kB		
X, Y, Z SRAM	25 kB each	75 kB each	300 kB each
SAD/MV RAM	0.3 kB	0.8 kB	3.2 kB
Total RAM	75.3 kB	225.8 kB	903.2 kB

As already discussed in Sect. 4, when implementing the integrated algorithm (Retinex plus motion analysis) described in Sect. 3 using the proposed ASIP, the Retinex pre-processing phase and the motion estimation phase can not occur in parallel. The two processing tasks should be time interleaved and in average the ASIP is used half-time for Retinex processing and half-time for motion estimation. Hence the integrated algorithm in Sect. 3 using the proposed ASIP can be implemented in real-time up to 30 frames/s for QCIF, SIF and CIF formats and up to 15 frames/s for VGA formats. The achieved performances match the requirements of typical vision-based system in automotive applications or for intelligent transport systems. The dynamic energy cost of the ASIP processing core (in the 65 nm CMOS technology) when running the algorithm in Sect. 3 amounts to roughly 12 nJ/pixel.

Finally, the parametric VHDL architecture of the ASIP has been re-synthesized considering, for the hardware units which implement motion estimation specific instruction, input samples quantized on 2 bits instead of 8 bits. The other hardware units (dedicated to Retinex filtering tasks or to general activities for I/0 interfacing, memory management, instruction and data flow scheduling) have been left untouched with respect to the previous discussed ASIP architecture. Synthesizing this new ASIP processor permits analyzing the complexity of the quantized-algorithm discussed in Sect. 3.3. The synthesis results on the same 65 nm CMOS technology prove that a lower circuit complexity is obtained, roughly 95 kgates for the logic, together with a small speed improvement being the maximum clock frequency roughly 320 MHz. Using the quantized motion estimation also the memory requirements in Table 1 can be reduced since only 2 out of 3 frame memories are required while the size of the third can be reduced by a factor 8-bit/2-bit = 4.

6 Conclusions

The problem of robust motion analysis of images acquired in real road scenarios for automotive or intelligent transport system applications is addressed in the paper. A novel technique based on integrated Retinex-like pre-processing algorithm with block matching video motion estimator is

presented allowing for many benefits versus state-of-art solutions: the entire system is more robust; the estimated motion vectors are more reliable and less dependent on critical ambient conditions like shadows or flashes; the proposed algorithm may allow to perform motion estimation using very few bits, still maintaining good performances. Real-time implementation is achieved by design a novel ASIP hardware architecture. With respect to a straightforward solution based on assembling different macrocells specific for motion estimation and Retinex preprocessing, the proposed ASIP architecture exploits the integration of the two techniques to optimize both the processing core and the memory organization. Synthesized in a 65 nm CMOS standard-cells technology, with 1.1 V supply and low-leakage HVT library version, the ASIP ensures real-time processing for video formats up to VGA with a bounded circuit complexity of 105 kgates and 15 kB of ROM for the processing core with a static power consumption of 25 μW and a dynamic energy cost of roughly 12 nJ/pixel. The on-chip memory size is in the order of hundreds of kilobytes depending on the supported image format. The achieved results prove the suitability of the proposed technique for real-time and motion analysis in automotive and intelligent transport systems applications.

Acknowledgments This research was partially supported by grants of the University of Trieste and of the Regione Friuli-Venezia Giulia within the "Eladin 2" project.

References

1. Yamada, K., Soga, M.: A compact integrated visual motion sensor for ITS applications. IEEE Trans. Intell. Transp. Syst. **4**(1), 35–42 (2003)
2. Gillner, W.: Motion based vehicle detection on motorways. IEEE Int. Veh. Symp. pp. 483–487 (1995)
3. McCall, J.: Video-based lane estimation and tracking for driver assistance: survey, system, evaluation. IEEE Trans. Intell. Transp. Syst. **7**(1), 20–37 (2006)
4. Soga, M., Kato, T., Ohta, M., Ninomiya, Y.: Pedestrian detection with stereo vision. ICDE Workshops 2005
5. He, Z., Qin, Z., Wen, H.: Video-based measure of traffic volume parameter. In: IEEE International Conference on Automation and Logistics, pp. 421–425 (2007)
6. Boltz, S., Wolsztynski, E., Debreuve, E., Thierry, E., Barlaud, M., Pronzato, L.: A minimum-entropy procedure for robust motion estimation. In: IEEE ICIP 2006, pp. 1249–1252 (2006)
7. Barjatya, A.: Block matching algorithms for motion estimation. Matlab central: http://www.mathworks.com/matlabcentral/fileexchange/8761 (2005)
8. Cheung, H.-K., Siu, W.-C., Feng, D., Wang, Z.: Retinex based motion estimation for sequences with brightness variations and its application to H.264. In: IEEE international conference on acoustics, speech and signal processing, 2008 (ICASSP 2008), pp. 1161–1164 (2008)
9. Cheung, H.-K., Siu, W.-C., Feng, D., Wan, Z.: Constrained one-bit transform for Retinex based motion estimation for sequences with brightness variations. In: International Conference on Neural Networks and Signal Processing, 2008, pp. 682–685 (2008)
10. Cheung, H.-K., Siu, W.-C., Feng, D., Wang, Z.: Windowing technique for the DCT based Retinex algorithm to handle videos with brightness variations coded using the H.264. In: IEEE ICIP 2008, pp. 2860–2863 (2008)
11. Urhan, O., Erturk, S.: Constrained one-bit transform for low complexity block motion estimation. IEEE Trans. Circ. Syst. Video Technol. **17**(4), 478–482 (2007)
12. Lee, S., Kim, J.-M., Chae, S.-I.: New motion estimation algorithm using adaptively quantized low bit-resolution image and its VLSI architecture for MPEG2 video encoding. IEEE Trans. Circ. Syst. Video Technol. **8**(6), 734–744 (1998)
13. Feng, J., Lo, K.-T., Mehrpour, H., Karbowiak, A.E.: Adaptive block matching motion estimation algorithm using bit-plane matching. IEEE ICIP 2005 **3**, 496–499 (1995)
14. Erturk, A., Erturk, S.: Two-bit transform for binary block motion estimation. IEEE Trans. Circ. Syst. Video Technol. **15**(7), 938–946 (2005)
15. Celebi, A., Urhan, O., Hamzaoglu, I., Erturk, S.: Efficient hardware implementations of low bit depth motion estimation algorithms. IEEE Signal Process. Lett. **6**(6), 513–516 (2009)
16. Erturk, S.: Multiplication-free one-bit transform for low-complexity block-based motion estimation. IEEE Signal Process. Lett. **14**(2), 109–112 (2007)
17. Land, E.H., McCann, J.: Lightness and Retinex theory. J. Opt. Soc. Am. **61**(1), 1–11 (1971)
18. Funt, B., Ciurea, F., McCann, J.: Retinex in Matlab. In: Proceedings of IS&T/SID 8th Color Imaging Conference, pp. 112–121 (2000)
19. Jobson, D.J., Rahman, Z., Woodell, G.A.: Properties and performance of a center/surround Retinex. IEEE Trans. Image Process. **6**(3), 451–462 (1997)
20. Orsini, G., Ramponi, G., Carrai, P., Di Federico, R.: A modified Retinex for image contrast enhancement and dynamics control. In: IEEE ICIP 2003, Barcelona, Spain (2003)
21. Marsi, S., Impoco, G., Ukovich, A., Ramponi, G., Carrato, S.: Using a recursive rational filter to enhance color images. IEEE Trans. Instrum. Meas. **57**, 1230–1236 (2008)
22. Ramponi, G.: Polynomial and rational operators for image processing and analysis. In: Mitra, S.K., Sicuranza, G.L. (eds.) Nonlinear Image Processing. Academic Press, New York (2000)
23. Jain, A.K.: Fundamentals of Digital Image Processing. Prentice-Hall International, Englewood Cliffs, NJ (1989)
24. Saponara, S., Fanucci, L., Ramponi, G., Marsi, S.: Algorithmic and architectural design for real-time and power-efficient Retinex image/video processing. J. Real Time Image Process. **1**(4), 267–283 (2007)
25. Saponara, S., Fanucci, L., Ramponi, G., Marsi, S., Kammler, D., Witte, E.: Application-specific instruction-set processor for Retinex-like image and video processing. IEEE Trans. Circ. Syst II **54**(7), 596–600 (2007)
26. Gilge, M.: Motion estimation by scene adaptive block matching (SABM) and illumination correction. In: SPIE Conference on Image Processing Algorithms and Techniques, pp. 355–366 (1990)
27. Schliebusch, O., Chattopadhyay, A., Witte, E.M., Kammler, D., Ascheid, G., Leupers, R., Meyr, H.: Optimization techniques for ADL-driven RTL processor synthesis. In: Proceedings of IEEE Workshop on Rapid Prototyping Systems, Montreal, pp. 165–171 (2005)
28. Schliebusch, O., Chattopadhyay, A., Kammler, D., Ascheid, G., Leupers, R., Meyr, H., Kogel, T.: A framework for automated and optimized ASIP implementation supporting multiple hardware description languages. IEEE ASP-DAC **1**, 280–285 (2005)

29. Dinoi, L., Martini, R., Masera, G., Quaglio, F., Vacca, F.: ASIP design for partially structured LDPC codes. Electron. Lett. **42**(18), 49–50 (2006)
30. Momcilovic, S. et al.: Application specific instruction set processor for adaptive video motion estimation. In: Proceedings of IEEE Euromicro DSD, pp 160–167 (2006)
31. Bajot, Y., Mehrez, H.: Customizable DSP architecture for ASIP core design. In: Proceedings of ISCAS'01, pp. 302–305 (2001)
32. Kappen, G., Noll, T.: Application specific instruction processor based implementation of a GNSS receiver on an FPGA. In: IEEE DATE'06, vol. 2, pp. 1–6 (2006)
33. Lee, J., Moon, J., Heo, K., Sunwoo, M., Oh, S., Kim, I.: Implementation of application-specific DSP for OFDM systems. In: Proceedings of IEEE International Conference on Circuits and Systems (ISCAS), pp. 665–668 (2004)
34. Yue, H., Lai, M.-C., Dai, K., Wang, Z.-Y., Design of a configurable embedded processor architecture for DSP functions. In: Proceedings of IEEE ICPADS'05, pp. 27-31 (2005)
35. Chattopadhyay, A., Ahmed, W., Karuri, K., Kammler, D., Leupers, R., Ascheid, G., Meyr, H.: Design space exploration of partially reconfigurable embedded processors. In: IEEE DATE'07, pp. 1–6 (2007)
36. Peters, H., Sethuraman, R., Beric, A., Meuwissen, P., Balakrishnan, S., Pinto, C., Kruijtzer, W., Ernst, F., Alkadi, G., van Meerbergen, J., de Haan, G.: Application specific instruction-set processor template for motion estimation in video applications. IEEE Trans. Circ. aSyst. Video Technol **15**, 508–527 (2005)
37. Vogt, T., When, N.: A reconfigurable application specific instruction set processor for Viterbi and log-map decoding. In: IEEE Workshop on Signal Proceedings Systems Design and Implementation, pp. 142–147 (2006)
38. Goossens, G., Lanneer, D., Geurts, W., Van Praet, J.: Design of ASIPs in multi-processor SoCs using the chess/checkers retargetable tool suite. In: IEEE International Symposium on System-on-Chip, 2006, pp. 1–4
39. Saponara, S., Casula, M., Fanucci, L.: ASIP-based reconfigurable architectures for power-efficient and real-time image/video processing. J. Real Time Image Process. **3**(3), 201–216 (2008)
40. Fanucci, L., Saponara, S., Bertini, L.: A parametric VLSI architecture for video motion estimation. Integration VLSI J. **31**(1), 79–100 (2001)

Author Biographies

Stefano Marsi was born in Trieste, Italy, in 1963. He received the Dr. Eng degree in electronic engineering (summa cum laude) in 1990 and the Ph.D. degree in 1994. Since 1995, he has held the position of researcher in the Department of Electronics at the University of Trieste where is the teacher of courses in electronic field. His research interests include non-linear operators for image and video processing and their realization through application specific electronics circuits. He is author or co-author of more than 50 papers in international journals, proceedings of international conferences or contributions in books.

Sergio Saponara got the Laurea degree, cum laude, and the Ph.D. in Electronic Engineering from the University of Pisa in 1999 and 2003, respectively. In 2002, he was with IMEC, Leuven (B), as Marie Curie Research Fellow. Since 2001, he collaborates with Consorzio Pisa Ricerche in Pisa. He is senior researcher at University of Pisa in the field of electronic circuits and systems for telecom, multimedia, space and automotive applications. He holds the chair of electronic systems for automotive and automation at the Faculty of Engineering. He co-authored more than 100 scientific publications and holds four patents. Sergio Saponara is also research associate of CNIT and INFN and served as guest editor of special issues on international journals and as program committee member of international conferences.