FPGA-based detection of SIFT interest keypoints

Article *in* Machine Vision and Applications · February 2013

DOI: 10.1007/s00138-012-0430-8

CITATIONS
20

READS
451

4 authors:

Leonardo Chang
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
24 PUBLICATIONS **85** CITATIONS
SEE PROFILE

José Hernández-Palancar
Advanced Technologies Applications Center, (CENATAV), Cuba
70 PUBLICATIONS **167** CITATIONS
SEE PROFILE

Luis Enrique Sucar
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
357 PUBLICATIONS **2,599** CITATIONS
SEE PROFILE

Miguel Arias-Estrada
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
142 PUBLICATIONS **528** CITATIONS
SEE PROFILE

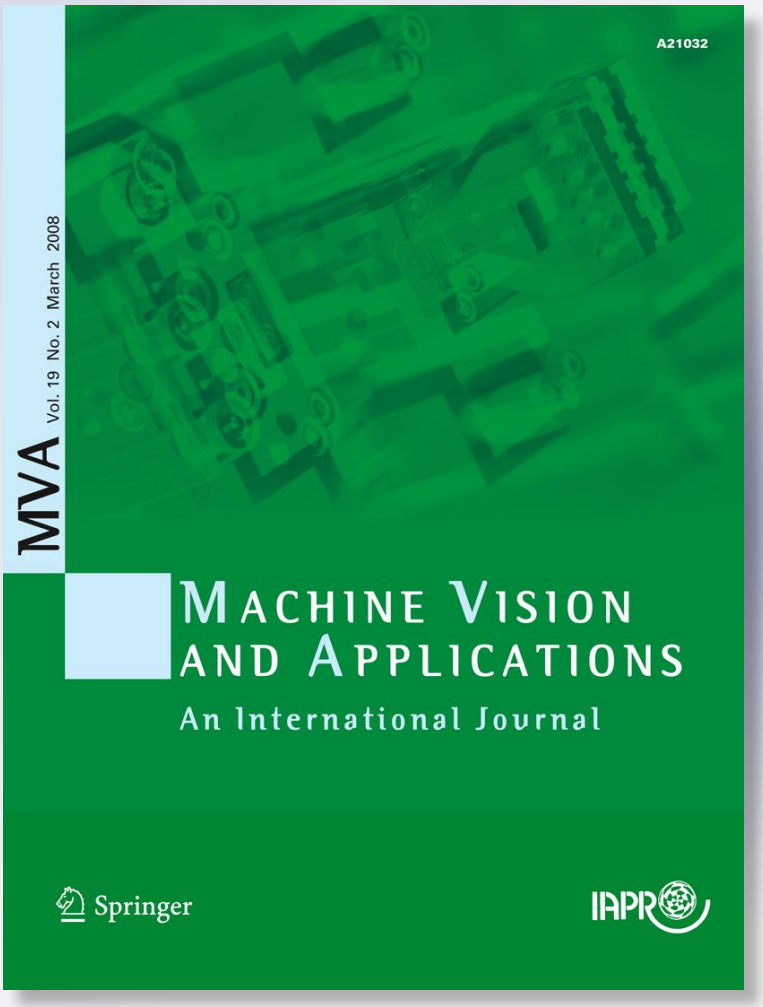
FPGA-based detection of SIFT interest keypoints

Leonardo Chang, José Hernández-Palancar, L. Enrique Sucar & Miguel Arias-Estrada

Machine Vision and Applications

ISSN 0932-8092

Machine Vision and Applications
DOI 10.1007/s00138-012-0430-8



Springer

Machine Vision and Applications
DOI 10.1007/s00138-012-0430-8

ORIGINAL PAPER

FPGA-based detection of SIFT interest keypoints

Leonardo Chang · José Hernández-Palancar · L. Enrique Sucar · Miguel Arias-Estrada

Received: 11 April 2011 / Revised: 21 December 2011 / Accepted: 23 April 2012

Abstract The use of local features in images has become very popular due to its promising results. They have shown significant benefits in a variety of applications such as object recognition, image retrieval, robot navigation, panorama stitching, and others. SIFT is one of the local features methods that have shown better results. Among its main disadvantages is its high computational cost. In order to speedup this algorithm, this work proposes the design and implementation of an efficient hardware architecture based on FPGAs for SIFT interest point detection In order to take full advantage of the parallelism in this algorithm and to minimize the device area occupied by its implementation in hardware, part of the algorithm was reformulated. The main contribution of the hardware architecture proposed in this paper and the main difference with the rest of the architectures reported in the literature is that as the number of octaves to be processed is increased, the amount of occupied device area remains almost constant. The evaluations and experiments to the architecture support this contribution, as well as accuracy, repeatability, and distinctiveness of the results. Experiments also showed device area occupation and time constraints of the hardware implementation. The architecture presented in this paper is able to detect interest points in an image of 320 × 240 in 11 ms, which represents a speedup of 250× with respect to a software implementation.

Keywords Local features · SIFT · Keypoint detection · Hardware architecture · FPGA

L. Chang (✉) · J. Hernández-Palancar
Advanced Technologies Application Center,
7a #21812 e/218 y 222, Siboney, Playa, CP 12220 Havana, Cuba
e-mail: lchang@ccc.inaoep.mx; lchang@cenatav.co.cu

J. Hernández-Palancar
e-mail: jpalancar@cenatav.co.cu

L. E. Sucar · M. Arias-Estrada
National Institute for Astrophysics, Optics and Electronics,
Luis Enrique Erro No. 1, Sta María Tonantzintla,
CP 72840 Puebla, Mexico
e-mail: esucar@inaoep.mx

M. Arias-Estrada
e-mail: ariasmo@inaoep.mx

1 Introduction

In Computer Vision, it is necessary to extract image features that can be used in applications such as object recognition, image retrieval, robot navigation, panorama stitching, face recognition, and others. These features should be invariant to image variations such as translation, rotation, scale, viewpoint, and illumination. The feature extraction process also needs to be repetitive and precise, so that the same features are extracted from different images containing the same object, as well as distinctive, that is to say, that the different features can be distinguished from each other.

In the past decade, significant progress was achieved in this direction with the development of local invariant features. One of the most popular and widely used local features method that has shown good results in this area is the Scale Invariant Feature Transform (SIFT) method proposed by Lowe [13]. The features extracted by SIFT are largely invariant to scale, rotation, illumination changes, noise, and small changes in viewpoint. The idea of this method is to first identify significant points in the image and to obtain a discriminant description of these points from its surroundings, which is then used for comparison between these descriptors using a similarity measure.

One of the main disadvantages of the SIFT algorithm is its high computational cost. This is the result of complex iterative processes to obtain invariance to the aforementioned

Published online: 30 May 2012

changes and transformations. For an image of 1,024 × 768 pixels, a software implementation [24] of the algorithm takes about 3 s to extract an average of 1,200 characteristics in a PC (CPU P4 3.0 GHz, 2 GB RAM).

There are several scenarios and applications that require features to be extracted and compared in real time (approximately 30 frames per second) and even on high-resolution images (more than 2 megapixels). Currently, very few systems running on personal computers achieve such processing results, and those who reach that speed, process low-resolution images or reduce the number of octaves and scales in the scale-space, compromising the robustness of the algorithm. Therefore, an implementation of this algorithm that achieves real-time processing with high repeatability and distinctiveness rates is desired.

A technique that has been widely used in recent years to accelerate computational tasks is the use of Field-Programmable Gate Arrays (FPGAs). These are revolutionary devices that combine the benefits of hardware and software. These devices can implement circuits, providing great advantages in energy, area, and performance compared with software. They can be reconfigured in a simple and low-cost manner to implement a wide range of tasks.

In this paper, to speedup the extraction of SIFT features, we propose a reformulation of the most computationally expensive phase of this algorithm: the detection of interest keypoints. Based on this reformulation, we propose a parallel algorithm and a hardware architecture for this stage of the SIFT method.

The main contribution of the architecture and the parallel algorithm proposed here is that, while increasing the number of octaves to be processed, the amount of occupied device area will remain almost constant, only increasing the number of memory blocks needed to store the new octaves and the logic needed to control the interleaving of more octaves. This is possible because all octaves for the same scale, regardless of the amount, will be processed in parallel in the same convolution block. This is relevant as the trend in computer vision is to work with larger images, and the number of octaves is a function of the size of the image. Therefore, for higher resolution images (and thus a greater number of octaves), the hardware logic required to process these higher resolution images will be the same. This contribution was supported by the experiments to architecture, showing quantitatively the benefits introduced with the interleaving of octaves processing.

The rest of the paper is organized as follows: In Sect. 2 the SIFT algorithm is described and its interest points detection stage is detailed. Section 3 discusses the works presented in literature to speedup SIFT using FPGA. The proposed reformulations and parallel algorithm aimed to obtain the maximum performance of a hardware implementation are presented in Sect. 4. The hardware architecture that implements the algorithm introduced in Sect. 4 is explained in Sect. 5. The tests to the proposed hardware architecture and the main results are discussed in Sect. 6. Finally, Sect. 7 concludes the paper, and future work is presented in Sect. 8.

2 Scale invariant feature transform

Methods based on comparisons of entire images or windows within them are suitable for learning and describing the global structure of objects, but cannot deal with partial occlusion problems, sudden changes in pose or viewpoint, or with non-rigid objects.

Significant advances have been accomplished in solving these problems with the development of local invariant features. The use of these features allows us to find local structures that are present in different views of the image. It also provides a description of these structures that is largely invariant to image transformations such as translation, rotation, scale, illumination, and viewpoint. A study and comparison of some local feature extraction methods is presented in [22].

The purpose of local features is to provide a representation that allows us to find correspondences between images efficiently and effectively. To satisfy this objective, the feature extractor must meet two important aspects:

- The feature extraction process must be repeatable and accurate, so that the same features of an object are extracted from different images containing that object.
- The features should be distinctive, so that extracted features can be distinguished from each other.

In turn, a sufficient number of features are required that cover the entire object so that it can be recognized even under partial occlusion.

SIFT, proposed by Lowe [13], is one of the most popular local features methods. Its descriptor has shown better results than other local descriptors [14]. This method tries to identify structures that are similar in different views of a scene and describe them by a vector which is independent of image size and orientation.

2.1 SIFT algorithm profiling

In order to achieve its invariance to scale changes and rotation, and as a result of complex and iterative processes, the SIFT feature extraction method is an expensive computational task.

Lowe divided his method in four major computational stages:

1. Scale-space extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint description

Table 1 SIFT algorithm profiling

Stage	Time (ms)	Percentage
(1) Scale-space extrema detection	1,391	44.83
(2) Keypoint localization	97	3.13
(3) Orientation assignment	341	10.99
(4) Keypoint description	1,274	41.05
(*) Whole algorithm	3,103	100.0

Table 1 shows execution times for each stage of the SIFT algorithm. These times were obtained for an image of size 1,024 × 768 pixels. We used the software implementation provided in [24]. The timing was acquired on a PC with an Intel P4 processor at 3.0 GHz and 2 GB of RAM.

As could be seen in Table 1 total running time was above 3 s. The scale-space extrema detection stage was the most expensive occupying nearly 45 % of the total processing. The high computational cost of this stage is due to the large number of convolutions that are produced to generate the Difference of Gaussians (DoG) scale-space, resulting in a large number of multiplication-accumulation (MAC) operations of floating-point numbers. The number of MAC operations to be performed for an $M \times N$ sized image to generate its DoG scale-space with O octaves and S scales is given by

$$\omega = \sum_{i=0}^{O-1} \frac{MN}{4^i} k^2 S,$$

where k is the Gaussian convolution kernel width.

Also, in this stage there are a large number of comparisons to find local extrema in the DoG scale-space which are marked as candidate keypoints. The number of comparisons at this stage is roughly given by

$$\varpi = \sum_{i=0}^{O-1} 26 \cdot \frac{MN}{4^i}(S-2), \tag{1}$$

For example, for the configuration used to obtain the above profiling ($M = 1{,}024$, $N = 768$, $O = 4$, $S = 6$ and $k = 7$) the number of MAC operations for the generation of the DoG scale-space is 307 077 120, and the number of comparisons for local extrema detection is 108 625 920.

The keypoint description stage proved to be the second largest in terms of computational cost, with more than 40 % of the total processing. At this stage of the algorithm, for each keypoint, a descriptor is generated from the gradient direction and magnitude of its neighbors. The calculation of the gradient orientation involves trigonometric operations, which are the most computationally expensive operations in the descriptor generation phase. In hardware, to achieve a result per clock cycle, this type of operation requires a large amount of device area. There are other solutions that use less silicon area, but take several clock cycles [23].

Scale-space extrema detection and keypoint description stages have similar computational costs, but the former has greater potential for parallelism and hardware acceleration. For these reasons, to obtain the highest possible acceleration of the SIFT algorithm by speeding up one of its parts, we focused on the scale-space extrema detection stage.

2.2 Scale-space extrema detection

This work presents an algorithm reformulation and a hardware architecture for the scale-space extrema detection phase of SIFT. This section describes in detail this stage of the algorithm and some of its theoretical foundations.

The scale-space extrema detection stage searches through all scales and image locations to find potential interest points that are invariant to scale and orientation. For this, the image is convolved with Gaussian filters at different scales and then differences between adjacent blurred images are obtained. Finally, the local maxima and minima in the difference of Gaussians (DoG) at different scales are marked as interest points.

For a given image $I(x, y)$, the SIFT detector is constructed from its Gaussian scale-space, $L(x, y, \sigma)$, that is built from the convolution of $I(x, y)$ with a variable-scale Gaussian:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y),$$

where $*$ is the convolution operator in x and y, and $G(x, y, \sigma)$ is the Gaussian kernel defined by

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2}.$$

The Gaussian scale space is created by generating a series of smoothed images at discrete values of σ. Thus, the σ domain is quantised in logarithmic steps arranged in O octaves, where each octave is further subdivided in S sub-levels. The value of σ at a given octave o and sub-level s is given by

$$\sigma(o, s) = \sigma_0 2^{o+s/S}, \quad o \in [0, \ldots, O-1], \quad s \in [0, \ldots, S-1],$$

where σ_0 is the base scale level, e.g., $\sigma_0 = 1.6$. At each successive octave the data are spatially down-sampled by a factor of two.

To efficiently detect stable keypoint locations in scale space, Lowe proposed to use extrema in the DoG scale-space, $D(x, y, \sigma)$, computed from the difference of adjacent scales:

$$D(x, y, \sigma(o, s)) = L(x, y, \sigma(o, s+1)) - L(x, y, \sigma(o, s)).$$

In order to detect the local maxima and minima of $D(x, y, \sigma)$, each pixel in the DoG images is compared with

its eight neighbors at the same image, plus the nine corresponding neighbors at adjacent scales. If the pixel value is bigger or smaller than all these neighbors, it is selected as an interest point.

3 Related work

In recent years, as a result of the popularity of SIFT as a local features method, and because of its high computational cost which makes it not viable for many real-time applications, several researchers have been trying to obtain faster implementations of this algorithm. Some researchers have focused on the use of Graphics Processing Units (GPUs). Examples of such works are [10,12,20,21]. Some other works in the literature have also addressed the speeding up of SIFT by using approximations or modifications in software. The most significant examples are [2,8,11]. Also, due to the widespread use and positive results of FPGAs as a means to speedup various computing tasks, researchers have begun to focus on developing systems based on FPGAs for real-time extraction of SIFT features. The main works that use these devices to accelerate SIFT are [3,6,15,17–19]. In this section we only discuss each of the last papers, highlighting their advantages and disadvantages, and analyzing the type of hardware architecture proposed in each of them, as they are the most relevant publications for the purpose of this paper.

The first work reported in the literature in the field of scale and orientation invariant feature extraction based on FPGAs was the work of Se et al. [19]. In their work, to speedup SIFT with respect to software implementations, the authors presented an FPGA implementation of the algorithm using fixed point arithmetic. This implementation was developed based on a software implementation employing floating point representation. The authors also mentioned that several of the routines of the software version were modified to make more efficient their hardware implementation. In order to implement most of the algorithm they used Xilinx System Generator. The authors suggest that using low-level hardware description languages, such as VHDL or Verilog, would be very costly in terms of development time. However, VHDL was used to implement low-level processes such as Direct Memory Access (DMA) and other memory access routines. In this study they used a Xilinx Virtex II FPGA. The SIFT execution time for a 640×480 image was reduced to 60 ms compared with 600 ms required by a Pentium III processor at 700 MHz. Their paper only provided the above details; there is not any kind of information about the modifications to the algorithm, and architecture specifications.

In [15], Pettersson and Petersson presented a partial implementation of SIFT for online stereo calibration. They implemented some of the most expensive parts of SIFT: the generation of DoG scale-space and Sobel filtering. These parts of the algorithm were implemented in a Xilinx Virtex II FPGA and the rest of the algorithm was implemented in software running on a personal computer. The authors propose a pipeline architecture where convolution blocks are cascaded to reduce the errors introduced by having a very small kernel compared with its standard deviation, and to be able to use a kernel of fixed size. For obtaining each scale-image it is used a different convolution block. For the convolution they use the separability property of the Gaussian kernel, and multiplications are replaced by using a Look Up Table (LUT); how to do this is described in [1,7]. This technique, despite replacing the multiplication operations, is a compromise between accuracy and size of the LUT, because it depends on the width of the convolution kernel and the number of bits used to represent it. The authors state that their systems work at 60 Hz and reduce the feature extraction time between 50 and 70 %, but no information about the resolution of the input image is provided. Besides there is no details on the use of FPGA device area; neither is there any analysis on the replacement of multiplications or any other information on the architecture that affects the accuracy of the results.

Another FPGA-based partial implementation of the SIFT is presented in [6]. In this work, Chati et al. present a hardware/software co-design to detect SIFT keypoints, implementing in hardware the parts with large degree of parallelism. They propose to use a wide array or sliding window to produce all scales at the same time; however, this is only mentioned and they do not provide any information about the operation of this method. In their paper, Chati et al. exposed some modifications to the algorithm for operation in hardware, but they did not provide details of the system architecture, nor mention details about the use of device resources, silicon area occupation, or other analysis. The device used was a Xilinx Virtex II Pro FPGA, where the system can process images of size 320×240 pixels in 0.8 ms.

The most complete FPGA implementation of SIFT reported to date in the literature is the work of Bonato et al. [3,4]. Their implementation uses a hardware/software co-design strategy; except the generation of descriptors, which is executed on a NIOS-II software processor, the remaining stages of SIFT are implemented in hardware. This architecture consists of three hardware blocks, one for the generation of DoG scale-space, one for the calculation of the orientation and magnitude, and one for the location of keypoints. The block for DoG scale-space generation receives the input image from the camera and the result is sent to the other two hardware blocks. In addition, this architecture has a software block that handles the generation of descriptors for each keypoint. The authors suggest that the generation of descriptors is developed in software as the type of calculation performed at this stage is more feasible to be conducted by a software processor; also it is easier to implement in soft-

ware than in hardware and gives greater flexibility to modify the descriptor according to the final application. The implementation of the DoG scale-space generation block considers the properties of separability and symmetry of the Gaussian kernel. In addition, they save four multipliers by normalizing the convolution kernel so that it always takes values of 0 or 1 on its first and last positions, avoiding the multiplication at these points. This brings the disadvantage of being forced to work with fixed point or floating point values, because for certain values of σ if these results are normalized in this way and then rounded to integers, all elements will have the same value. The proposed system implements 18 blocks of convolution with Gaussians, one for each scale-image, under a configuration of three octaves and six scales. Another modification in the architecture to save area of the device, is that they represent the DoG images with a 5-bit unsigned representation. Using an unsigned format affects the amount of points detected, which is reduced by about half, since only local maxima points are considered, not taking into account the minima. According to the authors, this decrease in the number of points is not considered a problem for their application to Simultaneous Localization and Mapping (SLAM) where only a few dozen of these are necessary, but this decrease in the number of points could affect other applications. This system, implemented in an Altera Stratix II FPGA with a NIOS-II soft processor running at 100 MHz, requires 33 ms to extract the SIFT features in an image of 320 $\times$ 240 pixels, where the architecture bottleneck is the generation of descriptors held in the NIOS-II.

In [18], Qiu et al. present an architecture for the generation of the DoG scale-space. This work outperforms [3] and [17] in terms of the use of device resources. This system manages to generate the DoG scale-space for input images of size 320 $\times$ 240 pixels in 12 ms. For this, they exploit the separability property of the Gaussian kernel, making the separable convolution as [17]. In addition, it uses the associative property of convolution, where the result of a convolution can be equivalent to two successive convolutions, and the sum of the squares of the radii of the convolution kernels of the latter is equal to the square of the radius of the first ($R_0^2 = r_1^2 + r_2^2$). This allows them to split one convolution in two, but using smaller kernels. According to the authors, the advantages of using this technique is given by the possibility of reusing intermediate results, saving hardware resources and simplifications provided by the order in which they perform the convolutions. Theoretically, this gives them a saving of up to 17.8 % of the cost of hardware resources. In this architecture, the authors propose to use only five convolution blocks, in which, after seven iterations, the DoG scale-space for five octaves and six scales ($O = 5, S = 6$) is generated. This scheme has the disadvantage that, despite using only five blocks of convolution (which implies a saving in the use of device area), seven iterations must be completed to obtain the whole DoG scale-space. The authors achieved not only improvements in the FPGA resources occupation with respect to [3,4], they also mentioned improvements in processing time, but comparing their architecture with the whole system in [3,4] and not just with the DoG scale-space generation stage of the algorithm, which in [3,4] is more efficient in time than the architecture proposed in the work of Qiu et al.

In our work we present a hardware architecture in order to speedup the detection of interest points (i.e. scale-space extrema) of the SIFT algorithm. The main difference between the architecture proposed in our work with earlier architectures reported in the literature, lies in a more efficient use of FPGA resources by interleaving the processing of octaves, while obtaining a result every two clock cycles, implying a considerable speedup over existing software implementations and many of the hardware implementations discussed in this section. Furthermore, the architecture presented in our work achieves higher rates of FPGA resources saving as the number of processed octaves is increased. This implies a great advantage since the number of octaves depends on the size of the image, and the trend in computer vision is to work increasingly with higher resolution images.

4 A parallel algorithm for scale-space extrema detection

When performing a particular computational task it is common to have several methods or algorithms. The final selection is usually given by the application and the hardware device to be use. Usually, the optimal algorithm for FPGA differs from the optimal algorithm for a general purpose processor or a sequential computer.

Although the specifications and configuration of FPGA systems looks like software programs in high-level languages, they specify hardware and not software. A reformulation of the algorithm in software can often mean a substantial improvement in the performance of the hardware due to the fact that a specific computational technique that is good in software does not necessarily have to be good in hardware [9]. Hardware provides flexibility to create optimal computational structures that best undertakes a given task as well as to exploit low level parallelism.

This section describes the proposed parallel algorithm for the scale-space extrema detection. This algorithm is a reformulation of the algorithm presented by Lowe [13] for this purpose. This algorithm is aimed at obtaining maximum performance in a hardware implementation of this stage of the algorithm. These reformulations are primarily focused on taking full advantage of parallelism in this process, while trying to minimize the device area occupied.

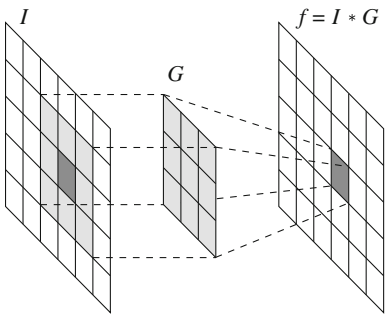


Fig. 1 In 2D convolution, the result of a pixel depends only on a neighborhood of the same size of the convolution window around the pixel in the input image. In this figure for a convolution window G of size 3×3 the result depends only on a region of equal size in the input image I

4.1 General considerations of the algorithm

In order to obtain an algorithm that allows a more efficient use of FPGA resources, we took into account the potential for exploitation of data parallelism, the separability property of Gaussian kernel, and the interleaving in the processing of octaves. This section details these elements that form the basis of our proposed reformulation for the scale-space extrema detection algorithm.

4.1.1 Exploiting data parallelism

Let I be a two-dimensional image and let G be a convolution mask of odd size $k \times k$, then the convolution of I and G is given by

$$f(x, y) = \sum_{-i}^{i} \sum_{-j}^{j} I(i, j)G(x - i, y - j), \tag{2}$$

where $i, j = \lfloor \frac{k}{2} \rfloor$.

As can be seen in Eq. 2, for the calculation of $f(x_1, y_1)$ only a neighborhood of size $k \times k$ of center (x_1, y_1) is necessary. This is also shown graphically in Fig. 1. Similarly, to determine if a point is a point of interest only a neighborhood of size 3×3 is needed in the DoG image and in the adjacent images in the DoG scale-space.

Previously mentioned characteristics of the 2D convolution and of the scale-space extrema detection provide a high potential for data parallelism, specifically the type Single Process, Multiple Data (SPMD). As an example of using the SPMD parallelism in this task we can divide an image into P partitions with an overlap of $k - 1$ lines and process all partitions simultaneously by using P different processors. This implies an improvement in processing time of P times, but also an increase in the use of the device area by the same factor. Therefore, the right balance between desired speedup and device area must be found depending on the application.

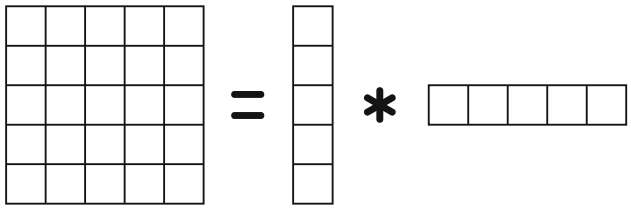


Fig. 2 A matrix of $M \times N$ is separable if it can be decomposed into two matrices $M \times 1$ and $1 \times N$

4.1.2 Exploiting the separability property of the Gaussian kernel

A technique that has been widely used in image processing to reduce the computational complexity of the 2D Gaussian filtering is the exploitation of the separability property of the Gaussian kernel [16]. A 2D filter is separable if it can be divided into two 1D signals: a vertical and a horizontal projection (see Fig. 2). The Gaussian filter can be separated as follows:

$$G(x, y, \sigma) = h(x, \sigma) * v(y, \sigma),$$

where

$$h(x, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}, \text{ and } v(y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-y^2/2\sigma^2}.$$

In addition, the convolution associative property holds:

$$I(x, y) * \big(h(x, \sigma) * v(y, \sigma)\big) = \big(I(x, y) * h(x, \sigma)\big) * v(y, \sigma).$$

Therefore, the 2D image convolution with a Gaussian filter can be carried out by first convolving the image with $h(x, \sigma)$ in the horizontal direction and then with $v(y, \sigma)$ in the vertical direction or vice versa. A 1D convolution to obtain an output value require k MAC (multiplication–accumulation) operations compared with k^2 MAC operations required by the 2D variant. Therefore, the computational advantage of the separable convolution versus non-separable is $k^2/2k$. Having a convolution window of size 7×7, the use of this technique would represent a reduction in the number of MAC operations by a factor of 49/14 = 3.5, which could represent a reduction of up to 3.5 times in the use of device area for these operations.

4.1.3 Octaves interleaving for spatial pyramid processing

After processing each octave, the image is sub-scaled by a factor of two, taking every second pixel in each row and column, i.e. $I_o(x, y) = I_{o-1}(2x, 2y)$. After scaling an image in half, the total number of pixels is reduced by four. In hardware, to reduce the amount of data, its sampling rate is reduced by the same factor. If after processing every octave the amount of data is reduced by a factor of four, the sampling

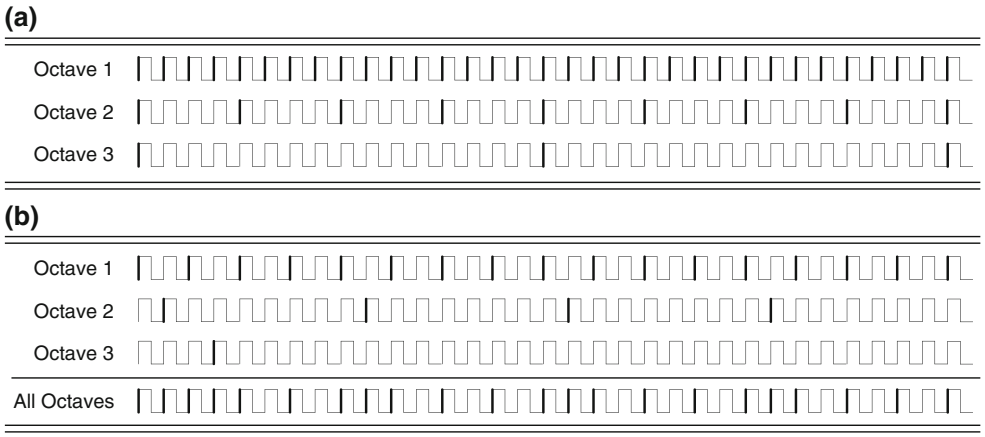


Fig. 3 The rising edges that are not marked in bold in (**a**) indicate the amount of processing time that is not used to obtain a new result. In (**b**) a way to take advantage of these times is shown, where the sampling period of each octave is doubled, making possible to interleave the processing of all the octaves in a single convolution block

period τ for an octave o is given by

$$\tau(o) = \tau_0 4^o, \tag{3}$$

where τ_0 is the sampling period of the first octave. Therefore, after sub-scaling, there is a large percentage of idle processing time with respect to the processing time of the first octave. This large amount of idle processing time is a result of the high sampling period in the last octave due to the small size of the images with respect to the original one. The idle processing time for a system of O octaves is given by

$$\hat{i} = \sum_{o=0}^{O-1} [\tau(o) - 1],$$

and can be identified in Fig. 3a as the rising edges not marked in bold in each of the octaves.

The main contribution of this paper is a scheme for spatial pyramid processing that takes advantage of these periods of inactivity, enabling the calculation of the O octaves of a scale in a single convolution block, no matter how big O is.

The general idea of this approach is to interleave the processing of the O octaves in only one processor. In order to do that, the sampling period of every octave is doubled, aiming to make place in the first octave processor for the calculation of the rest of the octaves. Figure 3b shows this idea.

We claim that using this technique, regardless of the number of octaves, all the octaves for a specific scale could be processed in a single processor (with the required latency), which involve a great system scalability and saving of hardware resources.

Proposition 1 *Let the first octave occupy every odd clock cycle, every other octave k (where $k = 0$ refers to the second octave) will occupy the cycles defined by the following sequence:*

$$s_k : a_k, a_k + \tau_1 \cdot 4^k, a_k + 2(\tau_1 \cdot 4^k), \ldots, a_k + n(\tau_1 \cdot 4^k), \ldots \tag{4}$$

where a_k is the first processing cycle of the octave k and $\tau_1 = 8$ is the sampling period of the second octave.

This interleaving order ensures that two or more octaves will never request the same processing clock cycle, allowing to interleave an infinite number of octaves. Moreover, letting a_k be the first unused cycle, an optimal interleaving order its obtained.

Proof. by Mathematical Induction:
BASIS: The case $k = 0$:

$$s_0 : 2, 10, 18, \ldots \quad a_0 = 2, x \equiv 2(8), \forall x \in s_0$$

is trivially satisfied because every clock cycle in s_0 is even and every clock cycle occupied by the first octave is odd.

INDUCTION STEP: Consider any $k > 0$. Assume the induction hypothesis that any two or more octaves except the last one will never request the same processing clock cycle (i.e. $s_0 \bigcap s_1 \bigcap \ldots \bigcap s_{k-1} = \emptyset$):

$$\begin{aligned}
&s_0 : a_0, a_0 + 8 \cdot 4^0, a_0 + 2(8 \cdot 4^0), \ldots, a_0 + n(8 \cdot 4^0), \ldots \\
&s_1 : a_1, a_1 + 8 \cdot 4^1, a_1 + 2(8 \cdot 4^1), \ldots, a_1 + n(8 \cdot 4^1), \ldots \\
&\vdots \\
&s_{k-1} : a_{k-1}, a_{k-1} + 8 \cdot 4^{k-1}, a_{k-1} + 2(8 \cdot 4^{k-1}), \ldots, \\
&a_{k-1} + n(8 \cdot 4^{k-1}); \ldots
\end{aligned}$$

then, the following congruences are satisfied:

$$
\begin{aligned}
&x_0 \equiv a_0(8), \quad \forall x_0 \in s_0\\
&x_1 \equiv a_1(8 \cdot 4^1), \quad \forall x_1 \in s_1\\
&\quad\vdots\\
&x_{k-1} \equiv a_{k-1}(8 \cdot 4^{k-1}), \quad \forall x_{k-1} \in s_{k-1}
\end{aligned} \tag{5}
$$

Assume that $\exists a_k$ such that it is not in any sequence $s_i, -1 \leq i \leq k-1$; then a_k do not satisfy any of the previous congruences in Eq. 5 and since $8 \cdot 4^k \equiv 0(8), 8 \cdot 4^k \equiv 0(8{\cdot}4^1), \ldots, 8{\cdot}4^k \equiv 0(8{\cdot}4^{k-1})$ by properties of congruences we have

$$
\begin{aligned}
&a_k + 8 \cdot 4^k \not\equiv a_0(8), a_k + 2(8 \cdot 4^k) \not\equiv a_0(8), \ldots, a_k + n(8 \cdot 4^k)\\
&\not\equiv a_0(8) a_k + 8 \cdot 4^k\\
&\not\equiv a_1(8 \cdot 4^1), a_k + 2(8 \cdot 4^k)\\
&\not\equiv a_1(8 \cdot 4^1), \ldots, a_k + n(8 \cdot 4^k) \not\equiv a_1(8 \cdot 4^1)\\
&\qquad\vdots\\
&a_k + 8 \cdot 4^k \not\equiv a_{k-1}(8 \cdot 4^{k-1}),\\
&a_k + 2(8 \cdot 4^k) \not\equiv a_{k-1}(8 \cdot 4^{k-1}), \ldots,\\
&a_k + n(8 \cdot 4^k) \not\equiv a_{k-1}(8 \cdot 4^{k-1})
\end{aligned}
$$

Then, based on the assumption that $\exists a_k$ such that it is not in any sequence $s_i, -1 \leq i \leq k-1$ we have proved that $s_0 \bigcap s_1 \bigcap \ldots \bigcap s_k = \emptyset$. Now we have to prove that there is always a possibility to find an a_k that is not present in any of the previous sequences.

The total number of clock cycles occupied by an infinite number of octaves is given by

$$
\begin{aligned}
L &= \frac{|U|}{2} + \frac{|U|}{2 \cdot 4} + \frac{|U|}{2 \cdot 4^2} + \cdots + \frac{|U|}{2 \cdot 4^k} + \cdots\\
L &= |U| \left(\frac{1}{2} + \frac{1}{2} \left(\frac{1}{4} \right) + \frac{1}{2} \left(\frac{1}{4} \right)^2 + \cdots + \frac{1}{2} \left(\frac{1}{4} \right)^k + \cdots \right)
\end{aligned}
$$

which have the form of the infinite geometric series $ar^0 + ar^1 + ar^2, \ldots, +ar^k + \cdots$ which converges to $\frac{a}{1-r}$ if and only if $|r| < 1$, since $a = \frac{1}{2}$ and $r = \frac{1}{4}$,

$$
L = \frac{2}{3}|U|.
$$

As the number of occupied clock cycles is less than the number of available ones ($L < |U|$) there will always exist a clock cycle a_k that is not occupied. This complete the proof of the induction step and thus of the proposition.

Further details about the hardware architecture that implements this idea are provided in Sect. 5.1.

4.2 Local extrema detection

As detailed in Sect. 2.2, the scale-space is constructed by generating a series of images blurred at discrete values of σ, where its domain is divided into logarithmic intervals organized in O octaves and where each octave is then divided in S sub-levels. Therefore, to obtain a result at a given location in the image in a certain scale it is necessary to obtain the value of that same location in the previous scale, and so on. Since a result of convolution only depends on a small region, all convolutions are performed concurrently, existing a latency, in the input data with respect to previous scale, relatively small compared with the size of the image. Similarly, the differences between adjacent scales to form the DoG scale-space are performed concurrently at the same time the DoG scale-space is being obtained.

In order to detect local extrema in the DoG scale-space, each pixel in the DoG images is compared with its eight neighbors in the same image, plus the corresponding nine neighbors in the adjacent scales. This implies that the same neighborhood of 3×3 on a certain scale is used three times, while processing its scale and while processing the two adjacent scales (see the total number of comparisons in Eq. 1). An efficient and equivalent way to obtain local maxima and minima that allows to reuse partial results is described below.

For every adjacent images is obtained the minimum and maximum point to point

$$
\begin{aligned}
\mathrm{Min}_1(x, y, o, s) &= \min\left(D(x, y, o, s), D(x, y, o, s+1)\right),\\
\mathrm{Max}_1(x, y, o, s) &= \max\left(D(x, y, o, s), D(x, y, o, s+1)\right).
\end{aligned}
$$

Then the process is repeated on the images obtained in the previous step:

$$
\begin{aligned}
&\mathrm{Min}_2(x, y, o, s)\\
&\quad = \min\left(\mathrm{Min}_1(x, y, o, s), \mathrm{Min}_1(x, y, o, s+1)\right),\\
&\mathrm{Max}_2(x, y, o, s)\\
&\quad = \max\left(\mathrm{Max}_1(x, y, o, s), \mathrm{Max}_1(x, y, o, s+1)\right).
\end{aligned}
$$

With this procedure it is possible to obtain images representing the minimum and maximum values over three adjacent images. To check whether a pixel is a point of interest it is necessary to prove that it is a local maximum or a minimum of $Min_2(x, y, o, s)$ or $Max_2(x, y, o, s)$, respectively. Figure 4 shows a diagram for this procedure. In addition, it should be checked that its value is equal to the corresponding pixel in the DoG, and despite being a local extrema it is not equal to its counterpart in any of the adjacent scales. For this to $Min_2(x, y, o, s)$ and $Max_2(x, y, o, s)$ a flag $\beta(x, y, o, s)$ is added to indicate these phenomena.

The total number of comparisons with the proposed local extrema detection method is defined in every octave by the $S-1$ image comparisons for the calculation of the first-order extrema, plus the $S-2$ image comparisons for the second-order extrema, plus the eight comparisons of every pixel in the second-order extrema images against its neighbors and the one needed to check the β flag:

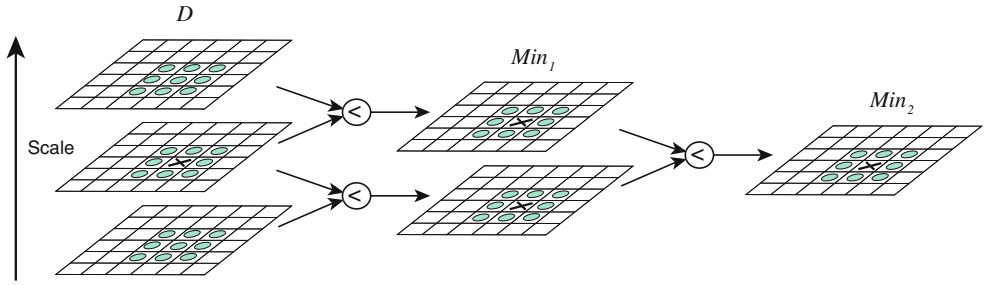


Fig. 4 A pixel (marked with X in D) is selected as a point of interest if it is a local minimum in a 3×3 neighborhood in Min_2 (marked with *circles*). Min_2 is the second order minimum between adjacent scales in the DoG scale-space. Similarly this figure applies to the maximum

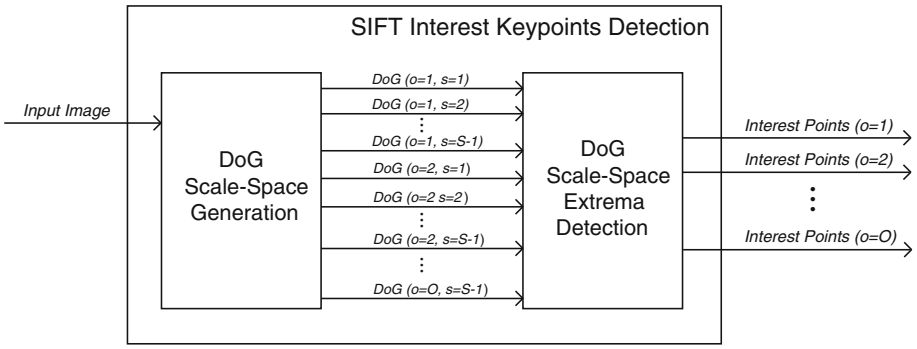


Fig. 5 The proposed architecture consists of two main parts: one for the generation of the DoG scale-space and the other for the detection of extrema in this space. The first block receives the image, generating the DoG scale-space, which serves as input to the second block, which extracts the points of interest

$$\varpi' = \sum_{i=0}^{O-1} \frac{MN}{4^i} [(S-1) + (S-2) + 9(S-2)],$$
$$= \sum_{i=0}^{O-1} 11 \cdot \frac{MN}{4^i} (S - 1.9). \quad (6)$$

Comparing the total number of comparison operations of the proposed method (Eq. 6) with the comparisons needed by the classical method (Eq. 1), a decrease by at least a factor of two is appreciated.

5 Proposed hardware architecture for scale-space extrema detection

In the previous section we proposed a reformulation for the scale-space extrema detection phase of the SIFT algorithm presented by Lowe [13]. This reformulation tries to maximize the advantage of the parallelism of this algorithm and to minimize the device area occupied by a hardware implementation. In this section we propose a hardware architecture for the scale-space extrema detection stage of the SIFT method. The architecture presented here implements the parallel algorithm proposed in the previous section.

The proposed architecture uses the elements discussed in Sect. 4, namely the exploitation of data parallelism, the exploitation of the separability property of the Gaussian kernel, and the octaves processing interleaving. The utilization of these elements contributes to a better use of the device area since they provide an efficient way to perform this process.

This section describes each of the parts that integrates the architecture, which are also illustrated with diagrams, indicating their relation with the parallel algorithm proposed in the previous section.

For the detection of scale-space extrema, the architecture is divided into two parts: (i) generation of DoG scale-space and (ii) detection of local extrema in this space (see Fig. 5). The input image is processed by the DoG scale-space generation block, which returns $O \cdot (S-1)$ images that form the DoG scale-space. These images are given to the local extrema detection block that determines which image locations are considered as points of interest.

5.1 DoG scale-space generation

In the architectures proposed in [15] and [3], to generate the DoG scale-space, the authors use one convolution block for each convolution operation that is carried out and divide the processing by octaves, so it takes $O \cdot S$ convolution blocks. In the architecture presented here, we propose to use only S convolution blocks for the $O \cdot S$ convolutions, dividing the processing by scales while keeping the same system performance. This is achieved by interleaving the octaves processing as detailed in Sect. 4.1.3.

A block diagram for the generation of DoG scale-space is shown in Fig. 6. This diagram shows a system of four octaves and five scales ($O = 4, S = 5$). This architecture can also be generalized to any configuration of these parameters.

The proposed architecture is mainly composed of Scale Calculation Blocks (SCB). A single SCB block performs O

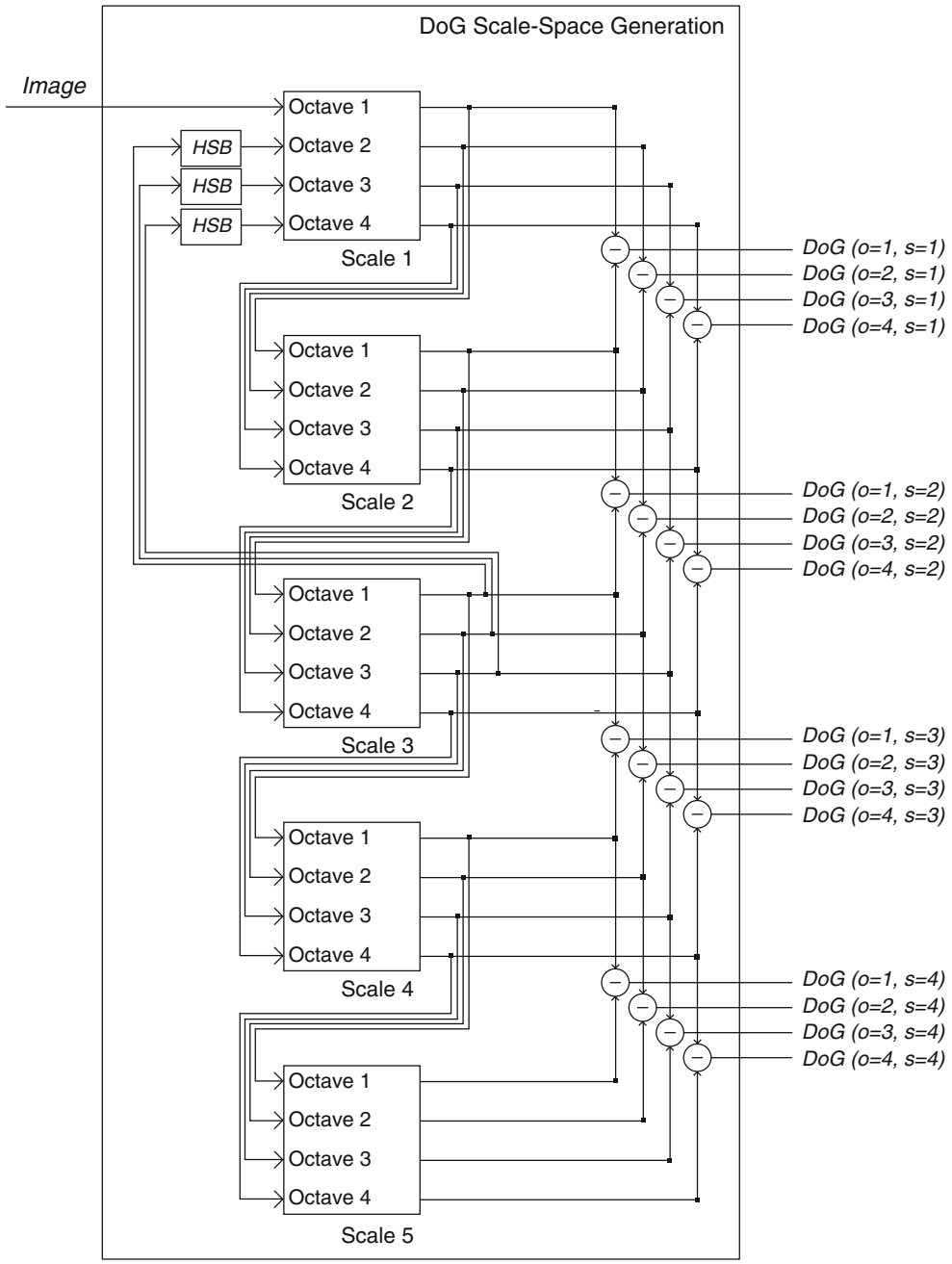


Fig. 6 High-level diagram of the architecture for DoG scale-space generation. This diagram shows the cascade connections between the scale processor blocks, where each of these blocks processes O octaves. As output of this block DoG scale-space is obtained which serves as input to the local extrema detection block

Gaussian filtering operations for a given scale, following the interleaving procedure described in Sect. 4.1.3. Therefore, each SCB block has O input ports and O output ports, one for each octave, respectively, where the sampling period for each octave is defined by Eq. 3. SCB blocks are cascaded to use a convolution kernel of fixed size and thus avoid the convolutions with large kernels. This cascading can be seen in Fig. 6.

A SCB block, for Gaussian filtering, takes advantage of the separability property of Gaussian kernel as described in Sect. 4.1.2. Taking advantage of that property, this block performs filtering first in the horizontal direction and then in the vertical, which can be seen in Fig. 7.

The internal organization of horizontal filtering block is detailed in Fig. 8. Each input signal is shifted through $k - 1$ registers, where k is the number of coefficients of the 1D

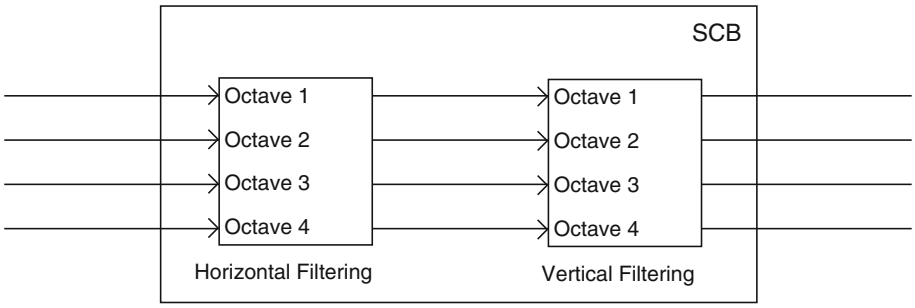


Fig. 7 2D convolution is performed by two consecutive 1D convolution, first passing through a horizontal filter and then through a vertical one

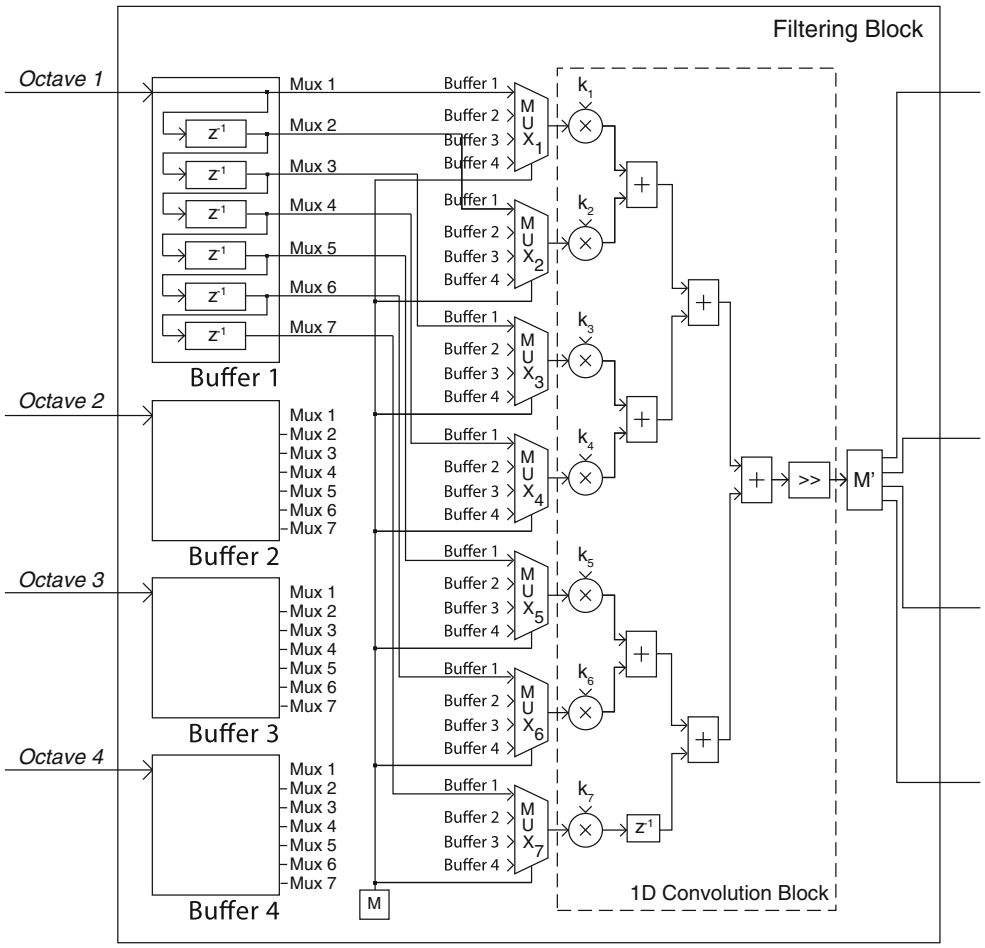


Fig. 8 Internal structure of a horizontal filtering block. It can be seen how all the octaves are processed in the same block, sharing the same hardware elements for convolution. The operating logic for interleaving the processing is given by the block *M* and by the multiplexers to which it controls

convolution kernel. The k signals corresponding to the O octaves are multiplexed with the aim of controlling the processing order of octaves and achieve the desired interleaving. The operating logic of the multiplexers in an instant t is determined by block M, which implements the interleaving order defined in Proposition 1.

The structure of vertical filtering block is the same as the horizontal, with the difference that each buffer stores the last k lines of the image instead of the last k pixels. To store these values a RAM block is used to store each line. Therefore, this part of the design will use $k-1$ blocks of RAM for each octave in each SCB block; hence the amount of RAM blocks used to generate the DoG scale-space is given by

$$\#\text{RAM\_blocks} = (k-1) \cdot O \cdot S.$$

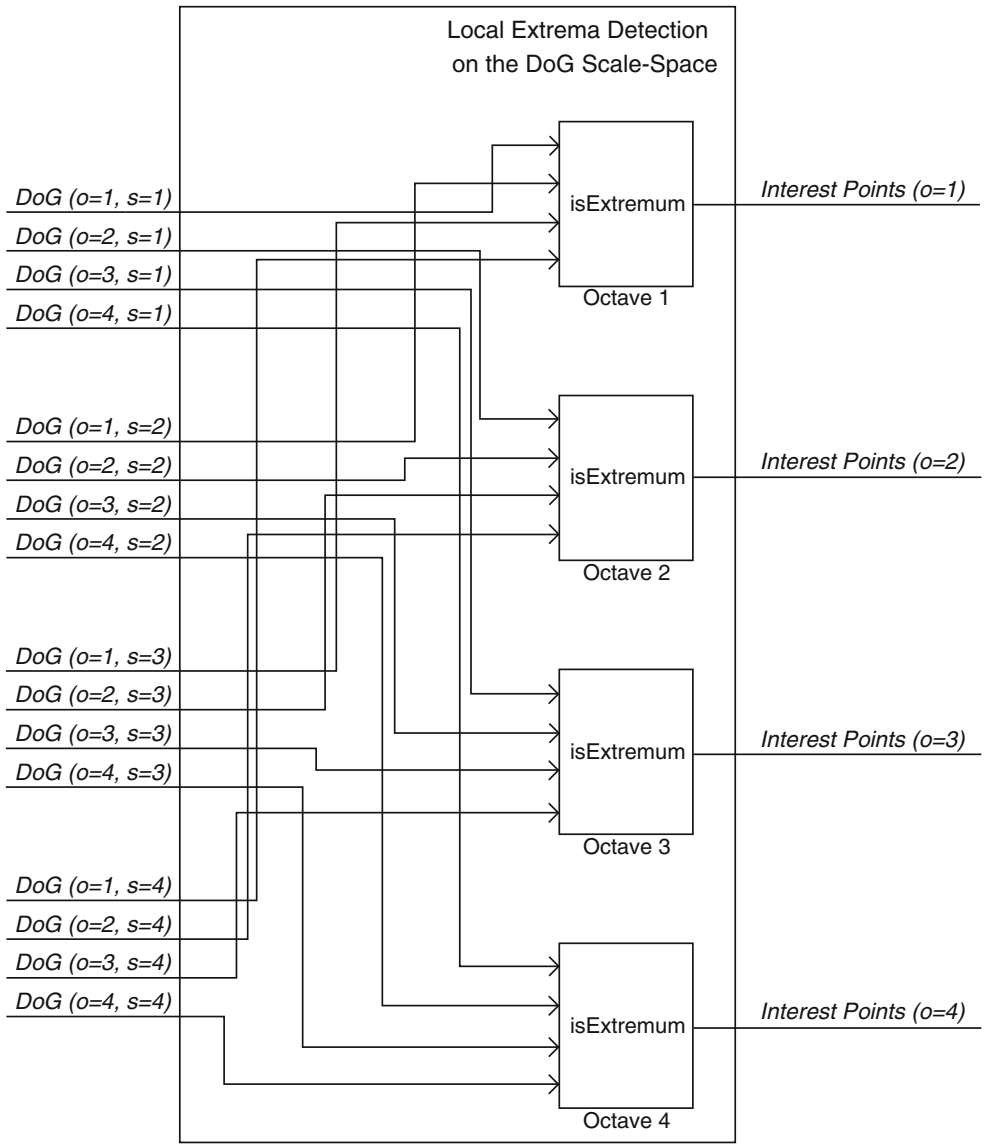


Fig. 9 High-level diagram of the architecture for local extrema detection in the DoG scale-space. Each block *isExtremum* receives all the DoG images of an octave, for which, this block determines the local extrema, i.e. points of interest

A 1D convolution block uses k multipliers and $k-1$ adders, number that we call $r(k)$. Then, the amount of multipliers and adders resources used by the architecture for building the scale-space is given by

$$\#\text{multipliers\_adders} = 2 \cdot r(k) \cdot S.$$

As it can be seen, this quantity only depends on the size of the convolution kernel and the number of scales, and it is independent of the number of octaves.

The HSB block in Fig. 6 performs image subscaling. To this end, an addressable shift register and a counter is used.

In order to replace the use of fixed-point values using integers, the coefficients of the convolution kernel are multiplied by a constant. Then, the filtered result is divided by this same constant. Preferably, this constant must be a power of two, to replace the division operation by a simple bit shift operation.

5.2 Local extrema detection on the DoG scale-space

The processing block that detects local extrema receives as input the DoG scale-space. This block implements the algorithm for this purpose stated in Sect. 4.2. A high-level diagram of this block for a system with a DoG scale-space of four octaves and four scales ($O = 4, S = 4$) is shown in Fig. 9. This architecture can also be generalized for any configuration of these parameters.

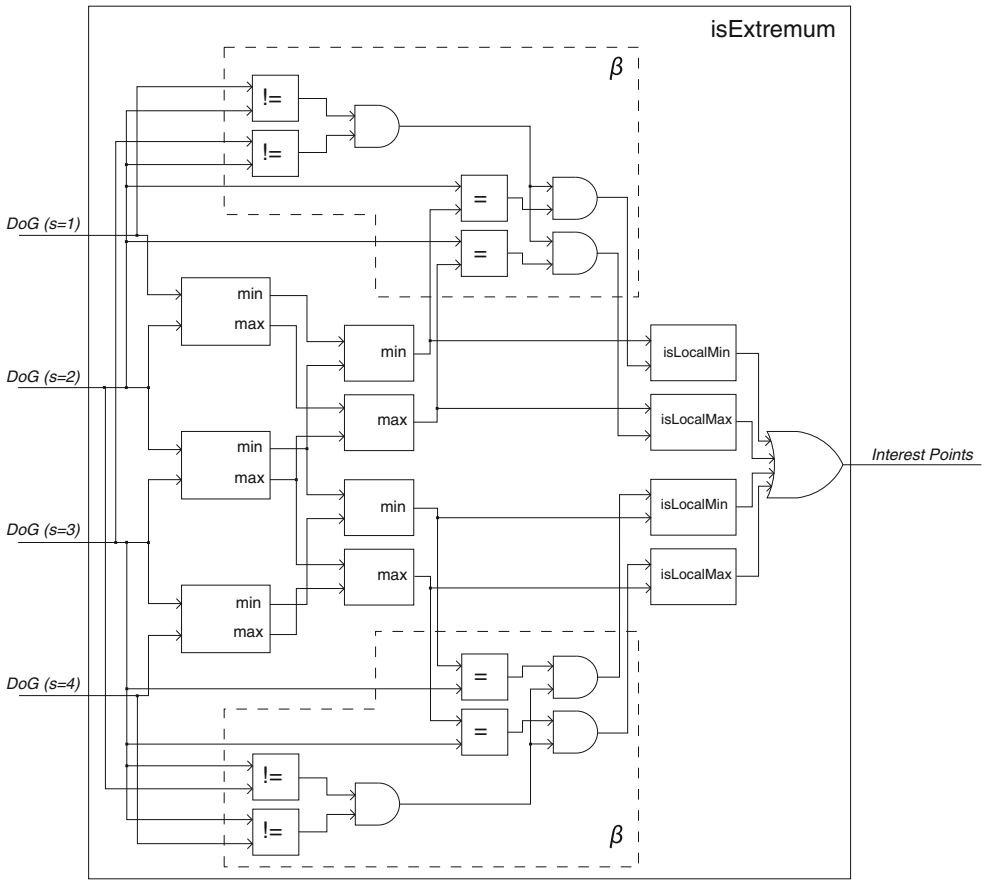


Fig. 10 Internal structure of *isExtremum* block. First, the block obtains the maxima and minima for every three adjacent images, this is done in two stages in order to reuse intermediate results. The blocks enclosed in *dashed lines* implement the β flag, which serves as input to the *isLocalMin* and *isLocalMax* blocks which determine the points of interest

For each octave, all the DoG images are passed to a *isExtremum* block, which determines which are the points to be considered of interest. The output of this block is a 1-bit vector indicating for each point if it is regarded as an interest point or not. The internal structure of a *isExtremum* block is detailed in Fig. 10.

As explained in Sect. 4.1.1, the procedure designed to detect local extrema aims to reuse the intermediate calculations by more than one scale, resulting in device resources saving. As can be seen in Fig. 10, this process was divided into two stages where minimum and maximum values in the images in common are reutilized. The blocks enclosed in dashed lines implement the β flag, which indicates whether each minimum or maximum value is equal to its corresponding pixel in the DoG and if it is not equal to its counterpart in any of the adjacent scales. *isLocalMin* and *isLocalMax* blocks determine whether each pixel is a local extremum in a neighborhood of 3×3, taking into account also the value of the β flag. If a point is an extremum at any scale it is considered as an interest point, so one OR gate is used before the output.

6 Experimental results

This section details and analyzes the experiments conducted on the proposed architecture for the scale-space extrema detection phase of the SIFT method. The evaluation of the architecture focused on measuring the reliability and accuracy of the results obtained and how they affect the repeatability and distinctiveness of the extracted SIFT features. The efficiency in the occupation of device area and the speedup obtained with respect to a software implementation is also analyzed. In addition, we compare our results with other related architectures reported in the literature.

In order to verify the accuracy and reliability of our proposed architecture a hybrid system was implemented where scale-space extrema detection stage is performed by our proposed architecture. The remaining stages of the algorithm are performed by Vedaldi [24] software implementation available online. Figure 11 shows an schematic of this hybrid system.

The proposed architecture was modeled and simulated using Xilinx System Generator 10.1 + Simulink. As shown

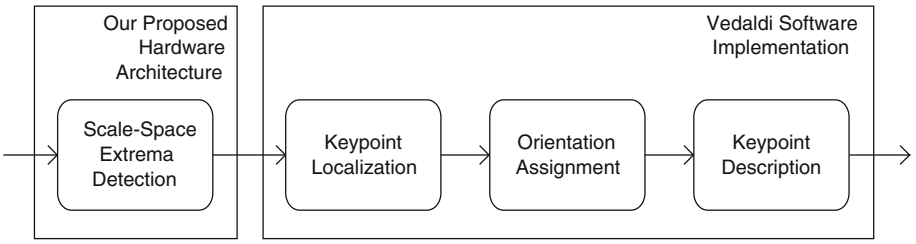


Fig. 11 Experimentation platform. The detection of the SIFT interest points stage is performed by our proposed architecture. The results are passed to a software implementation that executes the rest of the stages

in Fig. 11, the results of the first stage of the SIFT obtained from the simulation of the architecture are passed to the Matlab workspace, where the software implementation takes the values necessary to perform the remaining stages of the algorithm. The comparison between results obtained by this hybrid implementation and a software implementation [24] will provide us with a basis for determining the quality of the results produced by our architecture.

6.1 Accuracy evaluation

To evaluate the accuracy of our proposed hardware architecture we compared the results obtained in a set of 38 images for the hybrid implementation with the results obtained by the software implementation of Vedaldi. Test images were taken from Krystian Mikolajczyk website.[1] These images were captured with the aim of testing local feature extraction methods and were used in [14] to compare state-of-the-art local features methods. The experiments and analysis presented here only focus on the generation of the DoG scale-space, since for the detection of local extrema in this space the results were identical to the software implementation.

As evaluation measure we used the Mean Square Error (MSE). The MSE quantifies the difference between an obtained result and its expected or true value. It measures the average of the square of the error, where the error is the amount by which the result differs from the true value. In this paper the MSE is used to measure the difference between the values obtained by the proposed hardware architecture and a software implementation. The MSE is defined by Eq. 7.

$$\mathrm{MSE} = \frac{\sum_{M,N} [I_{sw}(m,n) - I_{hw}(m,n)]^2}{MN} \tag{7}$$

where $I_{sw}(m,n)$ and $I_{hw}(m,n)$ are the intensity values of the pixel (m,n) in the images of size $M \times N$ generated by the hybrid and the software implementations, respectively.

For these tests we use the scale-spaces generated for a configuration of six octaves and five scales ($O = 6, S = 5$). Smaller values of MSE indicate that the results generated by our architecture are more similar to those obtained by the software implementation (i.e. higher degree of accuracy). The results of this test are summarized in Fig. 12.

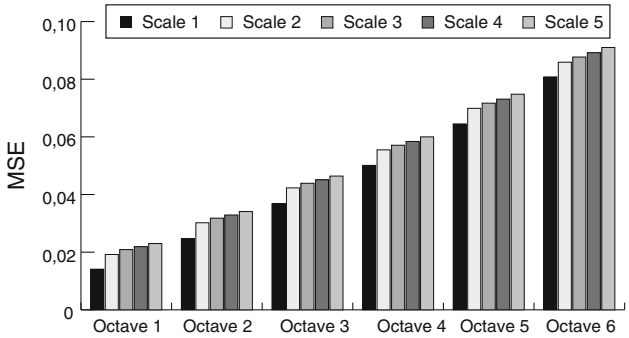


Fig. 12 MSE values for each octave and scale. The rounding and aproximation errors committed in the convolution process are propagated in the order of dependence between the images in scale-space and hence the MSE increase in that order

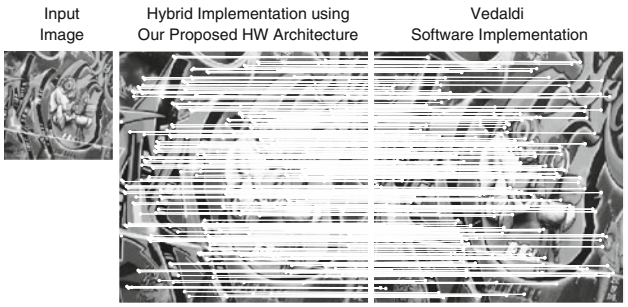


Fig. 13 SIFT features are extracted from input image using both implementations independently. Then, based on their keypoint correspondences we calculate the error in detection

Each bar in Fig. 12 represents the average MSE of the 38 test images in a specific octave and scale. In this figure, we can see how the error within an octave is increasing at every scale, as well as increases at every octave. This is due, as explained in Sect. 2.2, that each scale-image depends on the former scale-image, and the first image in each octave depends on the penultimate scale-image in the previous octave. Therefore, the error in the calculation of each image is propagated to the next. These dependencies between the images can be seen in Fig. 6.

The initial error, which is then increased over each scale and octave is caused by approximations to the Gaussian

[1] http://lear.inrialpes.fr/people/Mikolajczyk/.

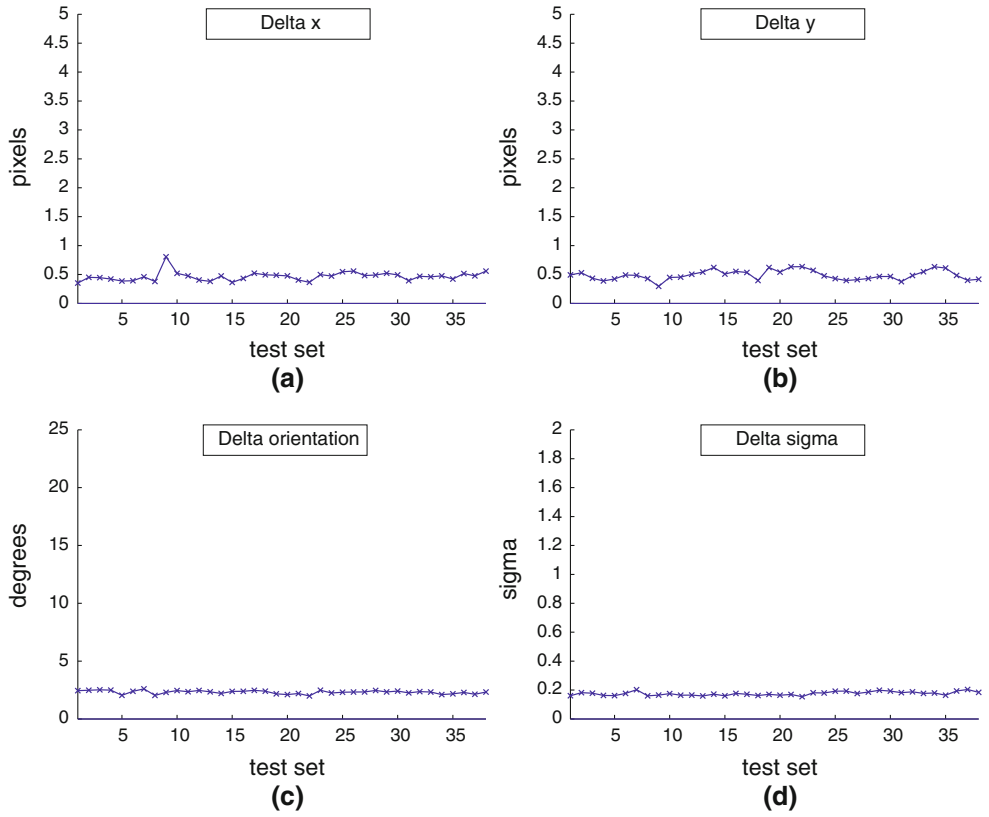


Fig. 14 Errors in the extraction of keypoints using our architecture with respect to software implementation. **a**, **b** shows the localization errors, **c** shows errors in orientation, and **d** errors in scale

convolution kernel to substitute arithmetic operations on float numbers with operations with integer values. This process is detailed in Sect. 5.1.

The MSE provides a quantitative measure of how much the rounding affects the scale-space obtained, but provides no information about how much these approximations affect the output of the algorithm, which is the detection of interest points. With this objective in mind, we extracted SIFT keypoints using the hybrid and software implementations described above from the same set of images. For each image, correspondences between the keypoints extracted by both implementations were found. An example of this is shown in Fig. 13.

Figure 14 shows the average variation in each of the 38 images for the keypoints obtained with both implementations. We measured variations in the coordinates, scale, and orientation of the gradient of the keypoints detected. It can be seen that variations in terms of coordinates of keypoints identified for these images did not exceed one pixel on average, although major changes were of four pixels. For gradient orientation each point variation was also small; the average variation was smaller than 2.5°, and the largest variations were of 10.0°. The average variation of σ was 0.2, which also represents a small difference.

6.2 Repeatability and distinctiveness evaluation

The previous section presented results showing the accuracy of the proposed architecture from a more theoretical perspective focusing on the error in the scale-space generation and keypoints detection. This section follows an experimentation more focused on the use of these features in an application. In a real application, we need points to be detected with great accuracy, but we also need repetitive and distinctive keypoints, that is, the same point can be detected in different views of the scene or object and that it can be differentiated from the others.

To this end, we checked the correspondences between SIFT keypoints detected by our architecture in different images of the same scene. Figure 15 shows examples of images used to evaluate the repeatability and distinctiveness of the proposed architecture. We evaluated four different changes in image conditions: changes in viewpoint (Fig. 15a), changes in scale and rotation (Fig. 15b), different JPEG compressions (Fig. 15c), and image blurring (Fig. 15d). In images with viewpoint changes the camera position varies from a frontal to a lateral position with a deviation of 60°. Images with scale and rotation changes were obtained by varying the camera tilt and optical zoom. Different JPEG

Fig. 15 Test set. In **a** changes in viewpoint, **b** changes in scale and rotation, **c** variations in JPEG compression, and **d** variations of blur. For each of these subsets the first image is taken as reference image

compressions were obtained with a standard software by modifying the parameter of image quality. Blurred images were obtained by varying the focus of the camera. These images were also obtained from the website of Krystian Mikolajczyk[2]; they were captured specifically aiming to test and compare local descriptor through a similar experimentation.

To measure keypoints repeatability and distinctiveness we use the matches rate. This is calculated as the ratio between the number of correct matches between two images and the smaller number of detected points in this pair of images:

$$\text{matches\_rate}(I, I') = \frac{\#\text{correct\_matches}(I, I')}{\min(\#\text{keypoints(I)}, \#\text{keypoints}(I'))}.$$

It is desired that the proposed architecture presents high matches rate values but also a high number of matches.

The results of these tests are shown in Fig. 16. The measure was calculated for each of the above variations between a reference image (first image in each column of Fig. 15) and the rest of the images in the subset. An ideal response would be a horizontal line at 100 %.

As can be seen in Fig. 16, for the first three variations, matches rates of the proposed architecture are smaller in comparison with software implementation, especially in the images of minor variations, with more similar

[2] http://lear.inrialpes.fr/people/Mikolajczyk/.

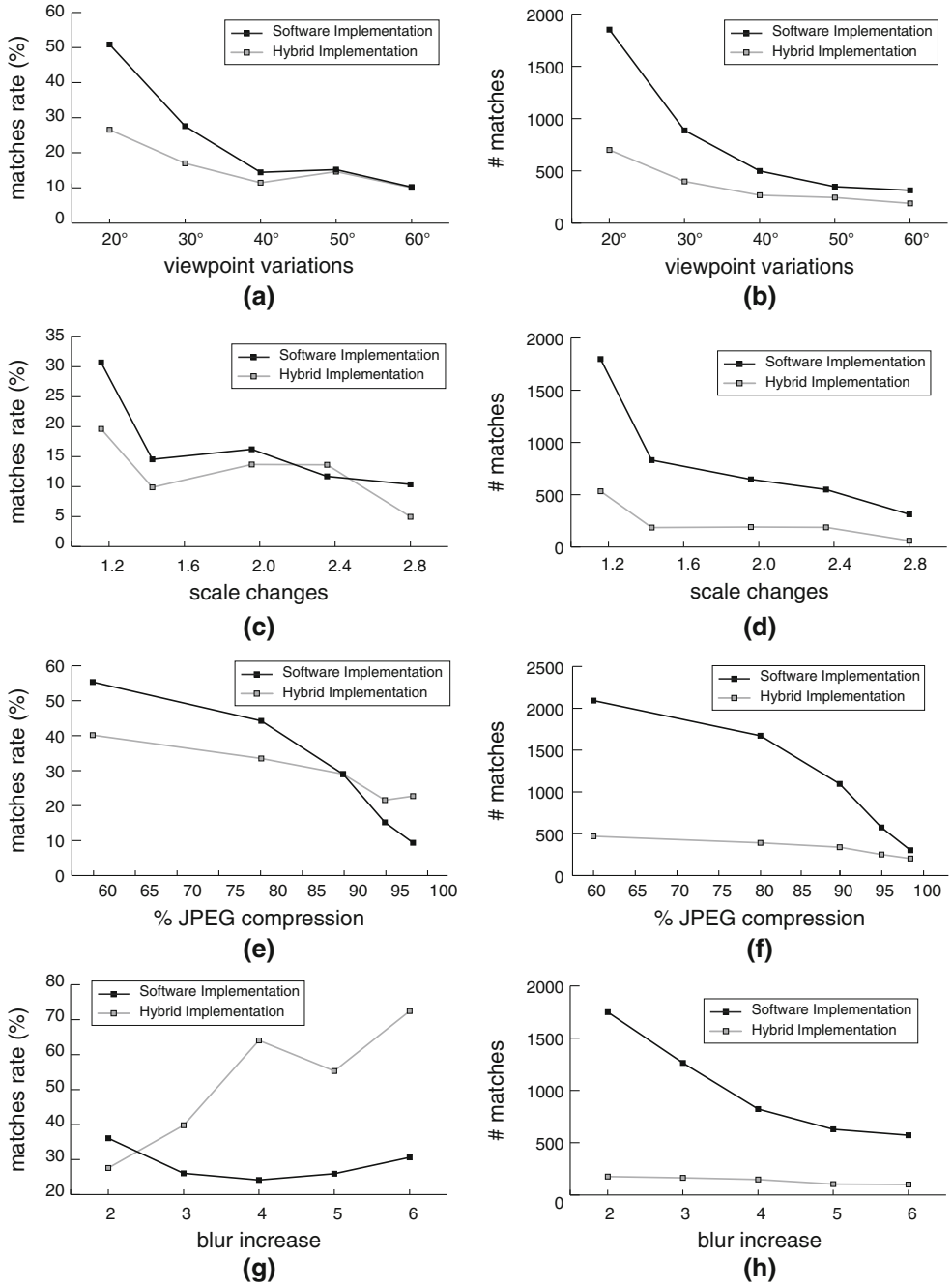


Fig. 16 Matches rates and number of matches for each subset of variations. As can be seen in **a**, **c**, and **e** matches rates of the proposed architecture are a little smaller when compared with the software implementation, particularly in the minor variations images, with more similar rates in images with greater variations. In **g**, the opposite situation is evidenced, induced in spite of the number of matches is smaller, the matches found were the most repetitive. As can be seen in **b, d, f**, and **h** the number of matches found was always more than 200, which represents a good number of extracted features for many applications. The differences in matches rates and matches count obtained by the hybrid implementation with respect to software implementation, show the impact on the repeatability and distinctiveness caused by errors discussed in the previous section. However, the fall of these values was not very drastic

rates in images with greater variations. The number of matches found was always more than 200, which represents a good number of extracted features for many applications.

The lines that describe the results obtained by the proposed architecture in all cases have a smaller slope than those obtained for the software implementation, indicating that despite their lower match rates in the images with minor

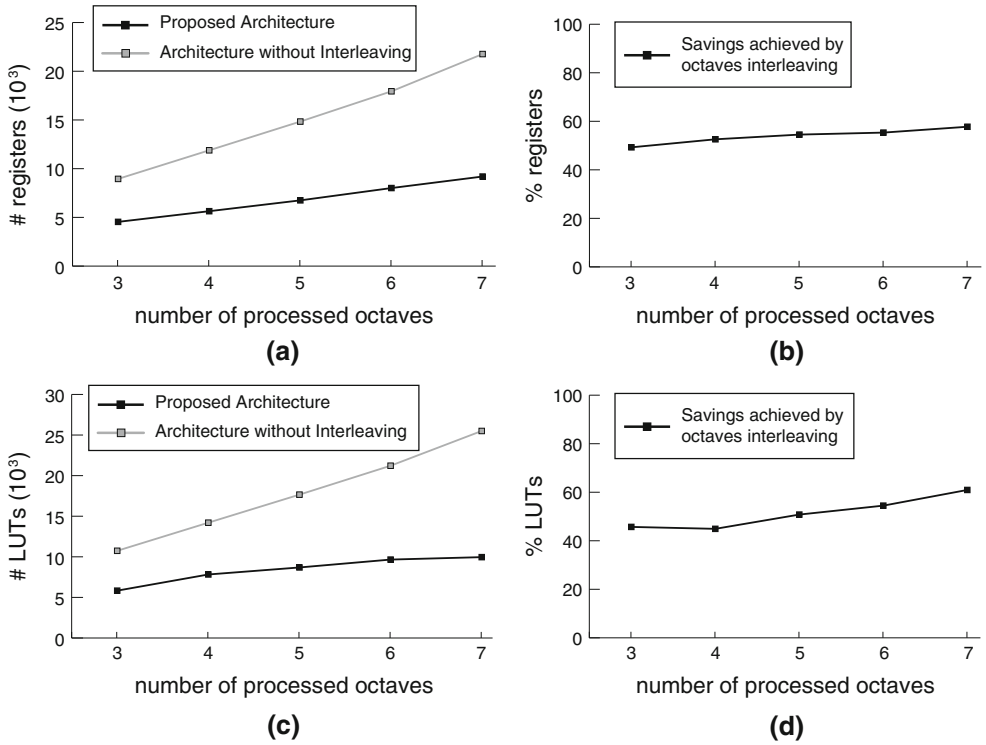


Fig. 17 Advantage in device area achieved by interleaving the octaves processing. **a**, **c** The differences between the proposed architecture using this technique and ignoring it. In **b** and **d** percentages of savings in the use of hardware resources

variations, the results were more stable over all images. Moreover, for the blurred images set the results showed a positive slope. The differences in match rates and match count obtained by the hybrid implementation with respect to software implementation show the impact on the repeatability and distinctiveness caused by errors discussed in the previous section. However, the fall of these values was not very drastic.

6.3 Evaluation of efficiency in FPGA resources occupation

In Sect. 4 it is stated that by introducing octave processing interleaving, for the same scale, all octaves could be calculated in the same processing unit; therefore, a great advantage in the consumption of the device area would be achieved. Later, in Sect. 5 the architecture that implements this idea is proposed and then the savings in FPGA resources that it implies are better evidenced. In this section we present several tests to validate the mentioned contribution. With this aim, the proposed architecture was redesigned having only one significant change, i.e. the removal of the interleaving of octaves processing. In this new design a filtering block is added for each image convolution at each scale and each octave. The resulting design is very similar to that proposed by Bonato et al. in [3]. Its high-level structure remains the same as the one shown in Fig. 6.

To demonstrate the benefits of octaves processing interleaving, we obtained several implementations of both architectures for different configurations of its parameters, where the number of octaves varied between three and seven ($O = [3, 7]$), and the number of scales ($S = 5$) and the dimensions of the image ($M = 512, N = 512$) were kept constant. Then, these implementations were synthesized for a Xilinx Virtex II Pro device (XC2-VP30-5FF1152), to obtain the amount of device resources occupied by each of these architectures for different number of octaves, and to obtain a quantitative measure of the advantage in terms of device area, achieved by interleaving the octaves processing.

Figure 17 summarizes these comparisons. In Fig. 17a, c, the number of registers and LUTs occupied by each of the architectures is shown. Also, the reduction of FPGA resources introduced by the octaves processing interleaving can be noticed; moreover, its lower growth trend can be appreciated and the line that describes it has a lower slope. Figures 17b and d show the percentage of saved registers and LUTs provided by the use of this technique. These values are almost all above 50 % and with a noticeable tendency to increase while increasing the number of octaves being processed.

Table 2 Hardware synthesis results of the proposed architecture for a configuration of $M = 320$, $N = 240$, $O = 3$, $S = 6$, $k = 7$ using a Xilinx Virtex II Pro (XC2VP30-5FF1152)

Logic utilization	Used	Available	Utilization
Number of Slice Flip Flops	5,676	27,392	20
Number of 4 input LUTs	5,554	27,392	20
Logic Distribution			
Number of occupied slices	4,393	13,696	32
Number of Slices containing only related logic	4,393	4,393	100
Number of Slices containing unrelated logic	0	4,393	0
Total Number of 4 input LUTs	6,699	27,392	24
Number used as logic	5,154		
Number used as a route-thru	1,145		
Number used as Shift registers	400		
Number of bonded IOBs	153	644	23
Number of RAMB16s	108	136	79
Number of BUFGMUXs	1	16	6

Table 3 Comparison with related works

Comparison parameters	Proposed architecture	Bonato et al. [3]	Qiu et al. [17]	Qiu et al. [18]
Image Size	QVGA	QVGA	VGA	QVGA
Max. clock frequency MHz	145.122	149.0	82.0	95.0
Throughput (Mpixels/seg)	72.6	149.0	5.1	15.3
Speed	900 fps	1,940 fps	16 fps	81 fps
Registers	5,676	7,256	6,333	6,120
LUTs	5,554	15,137	5,825	5,011

6.4 Comparison with related architectures

This section compares the results obtained by our proposed architecture with related works of Bonato et al. [3], and Qiu et al. [17,18]. To this end, the proposed architecture was synthesized in a Xilinx Virtex II Pro (XC2VP30-5FF1152) with a configuration as close to that of those works ($M = 320$, $N = 240$, $O = 3$, $S = 6$, $k = 7$). The synthesis results for these settings are summarized in Table 2.

After this process of synthesis was also determined that the implementation could operate at a maximum frequency of 145.122 MHz. Therefore, since the architecture returns a result every two clock cycles, our system is able to process 72.6 millions of pixels per second. With the achieved throughput it is possible to process high-definition video (1080×1280 pixels) at a 50 frames per second (fps) rate.

Table 3 compares these results with those obtained by related architectures (discussed in Sect. 3) reported in the literature for the detection of SIFT interest keypoints.

As can be seen in Table 3, the maximum frequency at which the system could work is higher than the rest of these works, except for the work of Bonato et al., which have very similar maximum frequencies. This maximum frequency, combined with the fact that our architecture returns a result every two clock cycles, allows us to have a processing speed of 900 fps, well above Qui et al. architectures [17,18]. The work of Bonato et al. for this stage of the algorithm returns a result per clock cycle, so this part working separately can achieve twice the speed of our system, although its general architecture has a restriction of 30 fps, which is introduced by another stage of the algorithm. The architecture proposed in this paper achieves half the throughput of the work of Bonato et al.; this is because to perform the octaves processing interleaving it is necessary to reduce by two the sample rate. However, we sacrifice half of throughput to obtain an advantage in device area of almost three times, which is the critical factor. Our work exceeds several times the throughput provided by Qiu et al. works. The proposed architecture also consumes less silicon area than the other architectures, except the number of LUTs compared with the work of [18] where the difference is very small. In addition, as discussed in other sections, the greater the number of octaves processed, the smaller the increased rate in the use of the device area of our architecture. Therefore, for a larger number of octaves,

Table 4 Comparison with other known implementations on software and GPU

Comparison parameters	Proposed architecture	SIFT OpenCV 2.3.1 [5]	SURF OpenCV 2.3.1 [5]	SiftGPU [25]
Used Hardware	FPGA Xilinx Virtex II Pro	Macbook Intel 2.4 GHz Core 2 Duo, 4 Gb RAM	Macbook Intel 2.4 GHz Core 2 Duo, 4 Gb RAM	8800 GTX 768 Mb GPU
Image size	QVGA	QVGA	QVGA	QVGA
Speed	900 fps	8 fps	19 fps	153 fps

the savings of hardware resources achieved by our architecture will be much bigger.

6.5 Comparison with other known implementations

This section compares the results obtained by our proposed architecture implemented in a Xilinx Virtex II Pro (XC2VP30-5FF1152) with other software and GPU-based well-known implementations. We compare our implementation against the implementations of SIFT and SURF in the latest version of OpenCV (2.3.1) [5] and against the GPU-based implementation SiftGPU [25]. The comparison results are shown in Table 4. As could be seen in Table 4 our results also outperform these implementations.

7 Conclusions

In this paper we proposed a hardware architecture for the detection of SIFT interest points. In order to take full advantage of the parallelism of this stage of the algorithm and to minimize the device area occupied by its implementation in hardware, part of the algorithm was reformulated. Given the characteristics of the algorithm we took into account the potential for exploitation of data parallelism. To decrease the amount of multiplication-accumulation operations and thanks to the separability property of Gaussian kernel we used the separable convolution. Also, we introduced the octaves processing interleaving, which allowed us to perform all convolution operations for a given scale in a single processing unit.

The main contribution of this architecture and the algorithm that it implements is that as the number of octaves to be processed is increased, the amount of occupied device area remains almost constant. This phenomenon is due to the fact that all octaves for the same scale—no matter how many—will be processed in the same convolution block.

The experiments and evaluations to the architecture, as first target, checked how similar the results were compared with a software implementation. Low error rates in the generation of Gaussian scale-space were reported, as well as average errors lower than a pixel on the location of interest points. Also, a series of tests to verify the variation in repeatability and distinctiveness of SIFT features detected by our architecture were conducted. We took into account several variations in the images as viewpoint, rotation and scale, JPEG compression and blur. The differences in matches rates were small, detecting a sufficient number of features correspondences between images. A series of tests that showed quantitatively the benefits introduced by interleaving the octaves processing were also carried out, resulting in savings in the use of device area above 50 % with an increasing tendency while more octaves are being processed. Finally, we compared the results obtained by our proposed architecture with other architectures reported in the literature for the detection of SIFT interest points. The proposed architecture showed best indicators of time and efficiency of device area use than the rest in almost all parameters. The architecture presented in this work is able to detect SIFT interest points in an image at a rate of one pixel every two clock cycles. Implemented in a Xilinx Virtex II Pro FPGA, with a configuration of three octaves and six scales, and a clock restriction of 145 MHz, an image of 320 $\times$ 240 is processed in 1.1 ms (900 fps), which represents a speedup of 250x (two orders of magnitude) with respect to Vedaldi software implementation.

8 Future work

Based on the results obtained in this paper, some ideas arise that can be followed as future work. First, to implement in hardware the remaining stages of SIFT, in particular the descriptors generation phase which is the second largest stage in terms of computational cost, and to thus obtain a greater speedup of the algorithm in general. Also, it is worth to explore hardware acceleration of other SIFT variations, since their algorithmic conception was designed with the aim of speeding up this algorithm, either by approximations or by substituting operations with equivalents of lower computational cost.

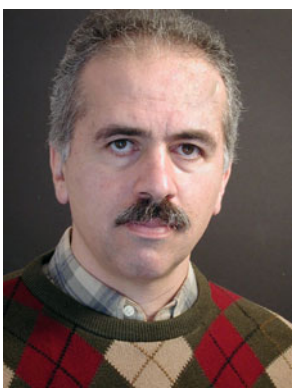
Acknowledgments This work was supported in part by CONACYT grant No. 103878. L. Chang was supported in part by CONACYT scholarship No. 240251.

References

1. Atmel: 3x3 convolver with run-time reconfigurable vector multiplier in Atmel at6000 FPGAs. http://atmel.com/dyn/resources/proddocuments/DOC0764.PDF (1999)
2. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (SURF). Comput. Vis. Image Underst. **110**(3), 346–359 (2008). doi:10.1016/j.cviu.2007.09.014
3. Bonato, V., Marques, E., Constantinides, G.A.: A parallel hardware architecture for scale and rotation invariant feature detection. IEEE Trans Circuits Syst Video Technol **18**(12), 1703–1712 (2008). doi:10.1109/TCSVT.2008.2004936
4. Bonato, V., Marques, E., Constantinides, G.A.: A parallel hardware architecture for image feature detection. In: ARC'08: Proceedings of the 4th international workshop on reconfigurable computing, pp. 137–148. Springer, Berlin (2008)
5. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
6. Chati, H.D., Muhlbauer, F., Braun, T., Bobda, C., Berns, K.: Hardware/software co-design of a key point detector on FPGA. In: FCCM'07: proceedings of the 15th annual IEEE symposium on field-programmable custom computing machines, pp. 355–356. IEEE Computer Society, Washington (2007). doi:10.1109/FCCM.2007.36
7. Evans, J.: Efficient FIR filter architectures suitable for FPGA implementation (1994). http://citeseer.ist.psu.edu/evans94efficient.html
8. Grabner, M., Grabner, H., Bischof, H.: Fast Approximated SIFT. In: Narayanan, P.J., Nayar, S.K., Shum, H.Y. (eds.) ACCV (1), Lecture Notes in Computer Science, vol. 3851, pp. 918–927. Springer, Berlin (2006)
9. Herbordt, M.C., VanCourt, T., Gu, Y., Sukhwani, B., Conti, A., Model, J., DiSabello, D.: Achieving High Performance with FPGA-Based Computing. Computer **40**(3), 50–57 (2007). doi:10.1109/MC.2007.79
10. Heymann, S., Frhlich, B., Medien, F., Müller, K., Wiegand, T.: SIFT implementation and optimization for general-purpose GPU. In: In WSCG07 (2007)
11. Ke, Y., Sukthankar, R.: PCA-SIFT: a more distinctive representation for local image descriptors. In: 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 506–513 (2004). doi:10.1109/CVPR.2004.1315206
12. Lalonde, M., Byrns, D., Gagnon, L., Teasdale, N., Laurendeau, D.: Real-time eye blink detection with GPU-based SIFT tracking. In: CRV'07: Proceedings of the Fourth Canadian Conference on Computer and Robot Vision, pp. 481–487. IEEE Computer Society, Washington (2007). doi:10.1109/CRV.2007.54
13. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004). doi:10.1023/B:VISI.0000029664.99615.94
14. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **27**(10), 1615–1630 (2005). doi:10.1109/TPAMI.2005.188
15. Pettersson, N., Petersson, L.: Online stereo calibration using FPGAs. In: Proceedings of IEEE Intelligent vehicles symposium, 2005, pp. 55–60 (2005)
16. Pratt, W.K., Adams, J.E.: Digital image processing, 4th edn. J. Electron. Imaging **16**(2), 029901 (2007)
17. Qiu, J., Huang, T., Ikenaga, T.: 1D-based 2D Gaussian Convolution Unit Based Hardware Accelerator for Gaussian & DoG Pyramid Construction in SIFT. In: Proceedings of the IEICE General Conference **2009**(2), 178 (2009-03-04). http://ci.nii.ac.jp/naid/110007095923/en/
18. Qiu, J., Huang, T., Ikenaga, T.: A 7-Round Parallel Hardware-Saving Accelerator for Gaussian and DoG Pyramid Construction Part of SIFT. In: ACCV09, pp. III: 75–84 (2010)
19. Se, S., Kong Ng, H., Jasiobedzki, P., jing Moyung, T.: Vision based modeling and localization for planetary exploration rovers. In: 55th International Astronautical Congress 2004 (2004)
20. Sinha, S., Frahm, J.M., Pollefeys, M., Genc, Y.: Feature tracking and matching in video using programmable graphics hardware. Mach. Vis. Appl. doi:10.1007/s00138-007-0105-z
21. Sinha, S., Frahm, J.m., Pollefeys, M., Genc, Y.: GPU-based Video Feature Tracking and Matching. Tech. rep. (2006). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.107.3260
22. Tuytelaars, T., Mikolajczyk, K.: Local invariant feature detectors: a survey. Now Publishers Inc., Hanover (2008)
23. Vadlamani, S., Mahmoud, W.: Comparison of CORDIC algorithm implementations on FPGA families. In: System Theory, 2002. Proceedings of the Thirty Fourth Southeastern Symposium on, pp. 192–196 (2002)
24. Vedaldi, A.: An open implementation of the SIFT detector and descriptor. Tech. Rep. 070012, UCLA CSD (2007)
25. Wu, C.: SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). http://cs.unc.edu/~ccwu/siftgpu (2007)

Author Biographies

Leonardo Chang is a researcher in the Systems Engineering Department at the Advanced Technologies Application Center (CENATAV), Cuba. He received his B.Eng. in computer engineering from the CUJAE University, Cuba in 2007. In 2010, he obtained his M.Sc. degree in computational sciences at the National Institute of Astrophysics, Optics and Electronics (INAOE) of Mexico, where he is currently a Ph.D. student in computational sciences. He is member of the Cuban Society of Mathematics and Computation and member of the Cuban Association for Pattern Recognition since 2007. L. Chang has focused his researches on the design of efficient hardware architectures for FPGAs and massively parallel implementations on GPUs for computer vision algorithms, but his interests also cover other areas of computer vision such as object categorization in images and face recognition.



José Hernández-Palancar graduated in computer sciences of Havana University in 1990. He completed his doctoral thesis in 1997 in the same institution. Currently, Dr. Palancar works in the Advanced Technologies Application Center (CENATAV) since 2003, where he is a senior researcher and deputy director for applied research, before he was the head of the Data Mining Department. In CENATAV, his research interests focus on parallel processing applied to data mining and biometrics.

L. Enrique Sucar has a Ph.D. in computing from Imperial College, London; an M.Sc. in electrical engineering from Stanford University; and a B.Sc. in electronics and communications engineering from ITESM. He is currently Director of Research at the National Institute for Astrophysics, Optics and Electronics, Puebla, Mexico. He has been an invited professor at the University of British Columbia, Canada; Imperial College, London; and INRIA, France. He has more than 150 publications and has directed 15 Ph.D. theses. Dr. Sucar is a member of the National Research System, the Mexican Science Academy, and senior member of the IEEE. He has served as president of the Mexican AI Society, has been member of the Advisory Board of IJCAI, and is Associate Editor of the journal Computación y Sistema. His main research interests are in graphical models and probabilistic reasoning, and their applications in computer vision, robotics and biomedicine.

Miguel Arias-Estrada obtained his B.Eng. in communications and electronics, and his M.Eng. in digital sytems at the FIMEE (University of Guanajuato) in Salamanca, Gto. in 1990 and 1992, respectively. In 1998, he obtained his Ph.D. degree at the Computer Vision and Systems Laboratory of Université Laval (Quebec city, Canada). He was a professor-researcher at the Computer and Systems Laboratory at Laval University where he worked on the development of a smart vision camera. Since 1998, he is with the Computer Science Department of INAOE (National Institute of Astrophysics, Optics and Electronics, Puebla, Mexico) where he continues his research on FPGA architectures for computer vision. His interests are computer vision, FPGA and GPU algorithm acceleration for 3D and machine vision.