International Journal of Electronics
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713599654

Local image phase, energy and orientation extraction using FPGAs

J. Díaz [a]; E. Ros [a]; S. Mota [b]; R. Carrillo [a]
[a] Department of Computer Architecture and Technology, University of Granada, Granada, Spain [b] Department of Computer Science and Numerical Analysis, University of Cordoba, Cordoba, Spain

To cite this Article Díaz, J., Ros, E., Mota, S. and Carrillo, R.'Local image phase, energy and orientation extraction using FPGAs', International Journal of Electronics, 95: 7, 743 — 760
To link to this Article: DOI: 10.1080/00207210801941200
URL: http://dx.doi.org/10.1080/00207210801941200

International Journal of Electronics
Vol. 95, No. 7, July 2008, 743–760

Local image phase, energy and orientation extraction using FPGAs

J. Díaz[a]*, E. Ros[a], S. Mota[b] and R. Carrillo[a]

[a]*Department of Computer Architecture and Technology, University of Granada, Granada, Spain;*
[b]*Department of Computer Science and Numerical Analysis, University of Cordoba, Cordoba, Spain*

(*Received 31 May 2007; final version received 19 September 2007*)

Local features such as phase, energy and orientation have been widely used in the literature for multiple computer vision applications such as edge detection, texture analysis, motion estimation, depth computation, etc. They can be computed using wavelet image decomposing based on Quadrature filters tuned at different scales and orientations. Although this represents a very powerful image analysis technique, its feasibility for real-time domain applications is still beyond current computer capabilities due to the high demand of computational resources. In this article, we present an architecture developed for real-time processing of these features using reconfigurable hardware. Although, to the best of our knowledge, previous implementations of Gabor or steerable filters on chip already exist, none of them provides energy, phase, and orientation information at the same time. We show how to implement quadrature filters based on Gaussian derivatives. We develop a hardware friendly technique to interpolate between a multi-oriented filter bank to compute features at their main orientation which has not been used before. We finally describe the system resource requirements, performance and some system illustrative results.

Keywords: real-time image processing; quadrature filters; local image phase; orientation and energy; fixed point arithmetic

1. Introduction

In this contribution, we have focused on three basic image properties, phase, energy and orientation, which provide significant cues in the process of image understanding. They have been extensively used on computer vision (Krüger and Wörgötter 2002; Krüger and Felsberg 2004) and, as we will see, they can be obtained from the same set of image processing operations.

Local image orientation encodes the geometric information of the signal whereas the phase can be used to differentiate between diverse image structures independently of their orientation differences. Because of the inherent ambiguity of orientation (two possible directions), we have used the double angle representation proposed by Granlund (1978).

Energy (or equivalently its square root, the magnitude) represents the information about the local luminance and contrast and therefore, is a luminance-depending feature. This is used in our system as a valuable parameter on the estimation of confidence parameters for our features.

*Corresponding author. Email: jdiaz@atc.ugr.es

ISSN 0020-7217 print/ISSN 1362-3060 online
© 2008 Taylor & Francis
DOI: 10.1080/00207210801941200
http://www.informaworld.com

Phase information, as is stated in the literature (Fleet and Jepson 1993), is more stable against image changes on contrast, scale, or orientation than luminance based information. It can be used to interpret the kind of contrast transition at its maximum (Kovesi 1999). It has been applied to numerous applications, especially for motion (Fleet 1992; Gautama and Van Hulle 2002), and stereo processing (Fleet, Jepson and Jenkin 1991; Solari, Sabatini and Bisio 2001; Díaz, Ros, Carrillo and Prieto 2007b).

The three image features are strongly interrelated. The estimation of the local phase and the local energy is an important step in many signal and image processing tasks, but this requires a second crucial task which is the estimation of the local orientation. In most cases, energy and phase are computed using a set of filters with some predefined orientation. Each complex filter is composed by an odd and an even component, where one is the Hilbert transform of the other. As local phase and energy information are intrinsically one-dimensional (1D) features, the preferred orientation is necessary for its computation unless some spherical filter is used (see Felsberg and Sommer 2001).

If these features are not computed at their corresponding orientation, our estimation will be suboptimal and will not reflect accurate values. This makes it necessary to properly cover the orientation space in order to obtain accurate estimations for these features. Furthermore, taking that under consideration, if the signal presented at an image position is not 1D as it happens in corners, junctions, or textures, these features can be multi-valued and then, complex analysis is required. In this contribution, we assume that we have 1D image signals and we address the implementation of a real-time architecture for computing these three features.

Previous works have been reported in the literature which implement real-time Quadrature filter computations which is the first processing stage of our system and the one which has the main processing power requirements. The real-time implementation of the multioriented decomposition on standard processors is still a challenge due to the large number of convolution operations required. Using Gabor filters, one 2D convolution for each orientation is required and at least seven filters are required for the steerable filters approach (for second order Gaussian derivatives). There are some preliminary works as Bernardino and Santos-Victor (2005) for fast software computation that achieve 5 fps for one orientation and multiscale analysis with an image resolution of 128×128 pixels on a P4 2.66 GHz processor. Using Matlab software and speeding up the convolution operations using the FFTW (Frigo and Johnson 2007), our source code runs at 1.2 fps of images of resolution 512×512 pixels on a Intel Dual Core 6600, 2.4 GHz processor. Even if a C/C++ implementation could significantly reduce the computation time, we are still far away from the real-time requirements of 25 fps.

The high computational load has motivated the development of customised chips to achieve the real-time. One important constraint of these implementations is that it uses fixed arithmetic in the computations. Another limitation on the systems developed in the literature is that only some features (or filter outputs) are provided for the systems.

For instance, Kubota and Alford (1997) present an interesting VLSI implementation of steerable filters for orientation computation. Unfortunately, the chip does not produce local phase information. Voß and Mertsching (2001) develop a Gabor filtering bank using reconfigurable hardware with 12 different orientations and able to compute 12 fps with images of resolution 256×256 pixels. They compare floating point arithmetic and fixed point arithmetic, concluding that, with proper intermediate data scaling, final system accuracy could be almost indiscernible between the fixed and the floating point versions. This result confirms our analysis presented in Díaz, Ros, Mota and Carrillo (2007c) and summarised in Section 3.

The system developed by Bouganis, Cheung, Ng and Bharath (2004) uses steerable filters to produce image wavelet decomposition. The hardware implementation works on one image pyramid and produces the Quadrature filter outputs at four orientations. Bit-width analysis is also presented. This implementation is quite similar to that of our first processing stage (convolution operations and linear filter combinations) but we have gone beyond that and combined this information into the chip. This reduces the communication bandwidth requirements.

In Darabiha, Maclean and Rose (2006), a local phase-based stereo algorithm is implemented on reconfigurable hardware. This system is a good example of application of the phase information to solve the image correspondence problem. Though input stages are also based on steerable filters, they only compute two different orientations and phase is not provided to the local path orientation but computed at two predefined orientations.

Finally, Cheung, Leong, Tsang and Shi (2006) present an FPGA implementation for Gabor filter computation using CNNs. The CNN template can be reprogrammed, but only one orientation is computed each time. Furthermore, no orientation, energy, or phase is directly provided by the chip because only binary cell output is provided. This makes the system inoperative for target applications such as stereo or motion image estimation, local contrast measure, or local image orientation estimation.

Our contribution contains relevant differences with the previous works. First of all, to the best of our knowledge, this is the first real-time system able to produce at the same time the energy, phase and orientation. Previous contributions are only able to produce one or two of the previous features, but as shown in Sabatini et al. (2007), a complete harmonic representation is required to properly analyse the input images. We have gone beyond the preliminary stages which are mainly convolution operations and have taken advantage of the shared operations required for these three features to reduce the hardware resources and improve features accuracy. Furthermore, we properly combine the different modalities to compute phase at their corresponding orientation using the Haglund's (1992) technique. This provides the proper local phase value instead of producing estimations at approximated orientations, which is usually done by most of the authors (Voß and Mertsching 2001; Cheung et al. 2006; Darabiha et al. 2006). From the hardware architecture point of view, a highly parallel design technique has been used to exploit the inherent FPGAs parallelism capabilities to achieve a computing performance of more than 56 Mpps (megapixels per second). Our architecture is able to provide one feature estimation per clock cycle which seems to not be the case for the four referenced architectures (not clear from their papers) and overcame the computing performance of the previous systems of the literature.

This article is structured as follows. In section 2, we summarise the required computer vision theory. In section 3, we analyse the degradation due to fixed-point arithmetic utilisation and section 4 describes the superscalar and long-datapath pipelined architecture developed. Finally, section 5 describes our main conclusions and future work.

2. Pixel-wise image processing model for estimation of orientation, energy and phase

Local image features can be extracted using Quadrature filters. A Quadrature filter is a complex filter that allows decomposing each output as phase and magnitude. If we note $\boldsymbol{h}(x,\mathrm{k})$ for a complex filter tuned to a spatial frequency f_0 along the x axis then:

$$h(x;f_0) = c(x;f_0) + js(x;f_0) \tag{1}$$

where c and s, respectively represent the odd and even components of the filters, fulfilling the condition that the odd and even filters are Hilbert transforms of each other. The convolution with the image $I(x)$ is expressed in Equation (2):

$$I \times h = \int I(\xi)h(x - \xi; k_0)d\xi = C(x) + jS(x) = \rho(x)e^{j\phi(x)} \tag{2}$$

where $\rho(x)$ denotes its *amplitude* (that we will also note as *magnitude*, and *energy* when we refer to its square value), $\phi(x)$ is the phase and the components $C(x)$ and $S(x)$ represent respectively the responses of the odd and even filters. Quadrature filters are usually computed using bandpass filters such as Gabor or Gaussian derivatives which analyse the image at discrete orientations. Furthermore, they fit well on FPGA devices because they are based on convolutions.

Phase information is intrinsically 1D and therefore, only can be computed for image areas which can be locally approximated as lines or edges. This leads to the necessity of local estimation of the main orientation. An extensive analysis of the available theories can be found in (Díaz 2006a). As a conclusion, orientation needs to be computed to properly estimate phase. This inherent relation between features motivates the development of a specific purpose architecture to compute the three of them into the same system.

2.1. Quadrature filter computation using Gaussian derivatives

Widely used in the literature, (Freeman and Adelson 1991; Mota, Ros, Díaz, Ortigosa and Prieto 2005), Gaussian derivatives allow the computation of any particular orientation based on a basic set of separable kernels. This property is usually referenced as stereability (Freeman and Adelson 1991). The kernels have to be properly weighted to get the desired oriented kernel. Quadrature filters are computed using the Hilbert transform of the Gaussian derivative as described in (Freeman and Adelson 1991). A key factor for its design is the derivative order, because the tuning frequency and bandwidth depend on this parameter. Figure 1 illustrates the effect of using different derivative orders and its relation with the Gabor filters.

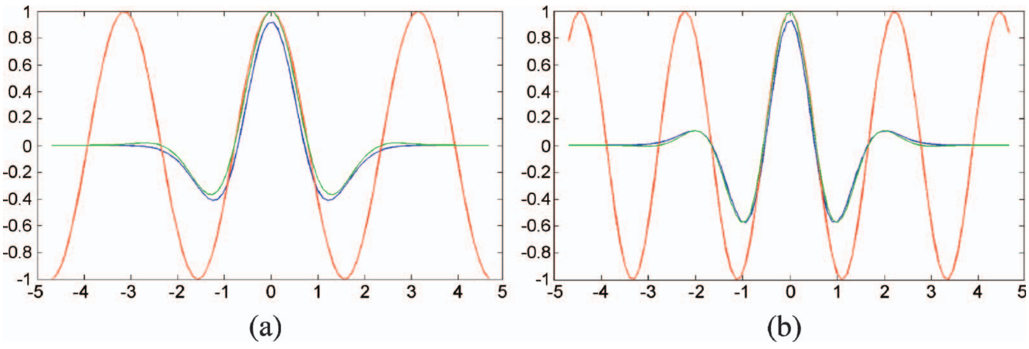


Figure 1. Comparison of Gabor (green), Gaussian derivatives (blue), and cosine (red) functions tuned to the same peak frequency. (a) Gaussian derivative of order 2 is close to Gabor filters but the difference is not negligible. (b) Gaussian derivative of order 4. This time, the similarity is higher and the filter is very close to the Gabor approach. Note that the number of waves increases according to the Gaussian derivative order, corresponding to higher orientation selectivity. These images are available in colour online.

Concerning the constraints of real-time embedded systems we will focus on second order Gaussian derivatives because they represent a good trade-off between accuracy and resource consumption. In our architecture, we use nine tap kernels with peak frequency of $f_0 = 0.21$ pixels^{-1} and bandwidth $\beta = 0.1$ pixel $\cdot$ s^{-1} as utilised on Freeman and Adelson (1991). This derivative order is able to provide high accuracy at a reasonable system cost.

Kernel equations of the Second order Gaussian derivatives G_{xx}, G_{xy} and G_{yy} and their Hilbert transforms H_{xx}, H_{xy}, H_{yx} and H_{yy} are presented on Equations (3) and (4). Note that one of the advantages of Gaussian derivatives (compared with Gabor filters) is that they are computed from a set of 2D separable convolutions, which significantly reduces the required resources. The Gaussian derivative equations are:

$$G_{xx} = 0.9213 \cdot \left(2x^2 - 1\right) \cdot e^{-\left(x^2+y^2\right)} \tag{3a}$$

$$G_{xy} = 1.843 \cdot x \cdot y \cdot \left(2x^2 - 1\right) \cdot e^{-\left(x^2+y^2\right)} \tag{3b}$$

$$G_{xx} = 0.9213 \cdot \left(2y^2 - 1\right) \cdot e^{-\left(x^2+y^2\right)} \tag{3c}$$

and for their Hilbert Transform:

$$H_{xx} = 0.9780 \cdot \left(-2.254x + x^3\right) \cdot e^{-\left(x^2+y^2\right)} \tag{4a}$$

$$H_{xy} = 0.9780 \cdot \left(-0.7515 + x^2\right) \cdot (y) \cdot e^{-\left(x^2+y^2\right)} \tag{4b}$$

$$H_{yx} = 0.9780 \cdot \left(-0.7515 + y^2\right) \cdot (x) \cdot e^{-\left(x^2+y^2\right)} \tag{4c}$$

$$H_{yy} = 0.9780 \cdot \left(-2.254y + y^3\right) \cdot e^{-\left(x^2+y^2\right)} \tag{4d}$$

After convolving the image with these filters, we steer them to estimate the Quadrature filters at some predefined orientations. In our system, we compute the Quadrature filter output at eight different orientations which properly sample the orientation space (further details in Díaz 2006a). The filters steering is implemented following Equation (5) where θ stands for the orientation.

$$\begin{aligned} c_\theta &= \cos^2(\theta)G_{xx} - \cos(\theta)\sin(\theta)G_{xy} + \sin^2(\theta)G_{yy} \\ s_\theta &= \cos^3(\theta)H_{xx} - 3\cos^2(\theta)\sin(\theta)H_{xy} - 3\cos(\theta)\sin^2(\theta)H_{yx} + \sin^3(\theta)H_{yy} \end{aligned} \tag{5}$$

The computation stages described in this section can be summarised as follows:

(1) Second order Gaussian derivatives G_{xx}, G_{xy} and G_{yy} and their Hilbert transforms H_{xx}, H_{xy}, H_{yx} and H_{yy} computation, using separable convolutions.
(2) Linear filter steering at eight orientations using Equation (5).

After these two stages, we have eight complex numbers for each pixel corresponding to the different orientations and even-odd filter outputs pairs.

2.2. *Hardware friendly features interpolation from a set of oriented filters*

If we consider eight oriented filters, it is likely that the local orientation of some features does not fit this discrete number of orientations. Under this circumstance, we need to interpolate the feature values computed from this set of outputs.

In this article, we will focus in the tensor-based Haglund's approach (Haglund 1992). Based on a local tensor that projects the different orientations, information can be computed as described in Equations (6)–(8). In these equations, E stands for energy (magnitude square), j for the complex unit and i for the orientation.

$$E_{\text{local}} = \frac{\sum_i E_i}{N} \tag{6}$$

$$\theta_{\text{local}} = \frac{1}{2}\arg\left(\sum_i \frac{4}{3}\sqrt{c_i^2 + s_i^2}\exp\{j2\theta_i\}\right) \tag{7}$$

$$\begin{gathered} \phi_{\text{local}} = a\tan\left({}^{s}\!/_{c}\right) \\ c = \sum_i c_i \cos^2(\theta_i - \theta_{\text{local}}) \quad s = \sum_i s_i \bullet \sin(\cos(\theta_i - \theta_{\text{local}})) \bullet \cos^2(\theta_i - \theta_{\text{local}}) \end{gathered} \tag{8}$$

There are three main drawbacks or problems to overcome in order to arrive at an affordable high performance hardware system:

(1) *Magnitude/energy for orientation computation*. Equation (7) requires a square root operation which is expensive to compute on reconfigurable hardware.
(2) *Image features dependencies*. Equation (7) uses the local energy extracted at each orientation, also computed on Equation (7). Equation (8) uses the orientation computed at Equation (7). This data dependency enlarges the pipeline datapaths and is very demanding in terms of resources.
(3) *Arctan function computation*. It can be computed using *Look-Up-Tables* LUT based methods or the CORDIC approach (Volder 1959).

We have carried out several modifications to overcome these problems. Problem I is solved by changing from magnitude to energy in Equation (7). We have also measured the difference between the original model and the modified version, considering this modification as a noisy signal. The SNR achieved using this modification is 57.4 dB, which is large enough to ensure a very high quality results. According to the previous discussion, the modified Equation (7) is presented in Equation (9).

$$\theta_{\text{local}} = \frac{1}{2}\arg\left(\sum_i \frac{4}{3}(c_i^2 + s_i^2)\exp\{j2\theta_i\}\right) \tag{9}$$

Problem II is related with the image feature dependencies. This point needs to take into consideration two aspects. First, local energy is used for mean energy computation as well as for orientation estimation. Because local energy only requires odd and even filters squaring and addition, this is not a critical dependency, because it can be solved just using a delay buffer. Second, the dependency between orientation and phase estimator circuits is more relevant. Orientation estimation involves the *arctan* function which usually requires a deep pipeline for high performance. The 8 odd and 8 even filter outputs require a buffering technique to make these data available when orientation is ready. Furthermore, the filter output should be accessible in parallel to maximise the circuits performance, which requires a large number of memory resources or registers. This requires a large amount of resources and should be avoided. Following the indications of Haglund (1992), Equation (8) can be simplified as described on Equation (10). The differences between the two choices, as indicated by Haglund (1992), are negligible and in the case of a 1D signal, there will be no difference at all between the two methods. We will use the new technique because it is more hardware friendly since it does not suffer from feature dependencies.

$$\begin{aligned} c &= \sum_i c_i \\ s &= \sum_i s_i \\ P_{\text{local}} &= \arctan\left({}^{s}\!/_{c}\right) \end{aligned} \tag{10}$$

Finally, relative to the *arctan* function implementation (presented as problem III) the main goal is to optimise accuracy *vs.* resource consumption trade-off. The techniques used for this issue are briefly explained in Section 4.

3. Model accuracy degradation due to fixed point arithmetic

The effect of the arithmetic representation and bit-width quantisation on the embedded system design has a critical impact in terms of resource consumption and system performance. Current standard processors use floating point representation, which has very large data range and precision, but digital circuits usually use fixed point representation with a very restricted number of bits. This is necessary because reconfigurable devices still cannot afford floating point arithmetic when implementing long-datapath pipelined computing architectures that may have a large number of processing elements. The quantisation effects should be properly studied to ensure that the data dynamic range is properly adjusted and the bit quantisation noise is kept at a small value. Therefore, this has motivated the bit-widths analysis of the system implemented by other authors (Voß and Mertsching 2001; Bouganis et al. 2004; Darabiha et al. 2006).

We have made a detailed analysis using the tool *MCode for DSP Analyzer* described in Díaz (2006a). The error metric for evaluating the system degradation has been the signal to quantization noise ratio (SQNR). The system here developed is intended to be used in the framework of the European project DRIVSCO (DRIVSCO 2007). It will be part of a hybrid system (hardware–software) for computing low-level vision cues (motion, stereo, object descriptors, etc.). In this context, hardware quality constrains the global accuracy baseline, thus this motivates the design of a high accuracy system. From these requirements, it is quite difficult to indicate an SQNR quality threshold (moreover, if we consider that this value is different for each image depending on its structure). Therefore, we have experimentally determined which values of SQNR produce quantised feature outputs which are almost indiscernible by visual inspection of the floating point versions. From that and as affordable values, we conclude that SQNR should be above 40 dB for energy, 25 dB for orientation, and 20 dB for phase.

In our system, we use multiple data word-lengths to tune each stage accuracy demands. The main designing decisions are outlined in structuring the model in the following stages:

(1) Convolution kernel weights of the G_{xx}, G_{xy}, G_{yy}, H_{xx}, H_{xy}, H_{yx}, and H_{yy}.
(2) Convolution output results for this filter bank.
(3) Oriented quadrature filters after steering operations.
(4) Trigonometric functions (sin, cos and their powers or multiplications) that are used at the steering and interpolation stages.

(5) Main non linear operations: division and *arctan*.
(6) Estimated goal features: local energy, phase and orientation.

The data outputs of each stage have been properly scaled between the different stages [see Díaz et al. (2007c) for details]. The first two stages are the ones that constrain most the system accuracy. Therefore, we have parameterised our analysis according to the bit-width of these stages. In Figure 2(a,b), we present our analysis for a test bank of six images.

In our final system implementation, we have taken 13 bits for the kernels (according to results of Figure 2(b) right image) and 11 for the convolution outputs. Trigonometric functions require only 9 bits. Arctan function uses 24 bits. Finally, because orientation and phase are angles and therefore their dynamic ranges are well defined (0 to π for orientation and $-\pi$ to π for phase), we have used 9 bits to represent their values. For the energy, the dynamic range depends on the convolution output bit-widths. We have used 22 bits to achieve satisfactory results. Note that the differences between floating point and fixed point representation of Figure 2(c,d), although numerically significant, are not visible on the qualitative evaluation. This validates the previous choices as good alternatives for hardware implementation.

The final SQNR values for the different images with the bit-widths decisions commented above are summarised in the Table 1.

4. System architecture details

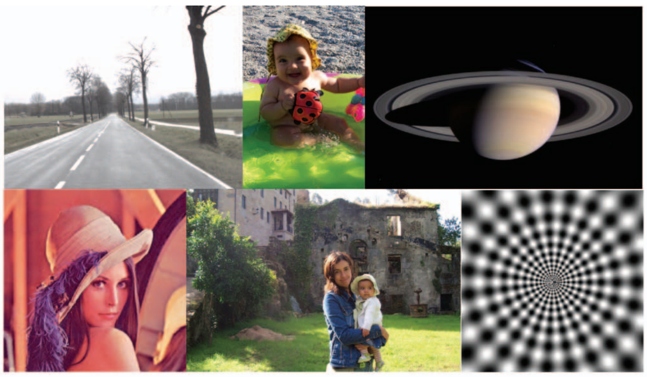
The Equations (7), (9) and (10) have been used for designing our real-time architecture with reconfigurable hardware. This is a technology that is being used in very different research fields, such as mobile robotics (Bonato, Fernández and Marques 2006), network intrusion detection (Clark, Ulmer and Schimmel 2006), cryptography (Kerins, Marnane and Popovici 2006), etc. and that has demonstrated a high performance for image processing applications (Díaz et al. 2006b; Díaz et al. 2007a,b).

The final system is a long-datapath pipelined architecture whose parallelism grows along the stages to keep our throughput goal of one pixel output for clock cycle. This is a key factor to achieve high performance in our system. Architectures based on resources (processing units) sharing techniques require more than one clock cycle for each feature

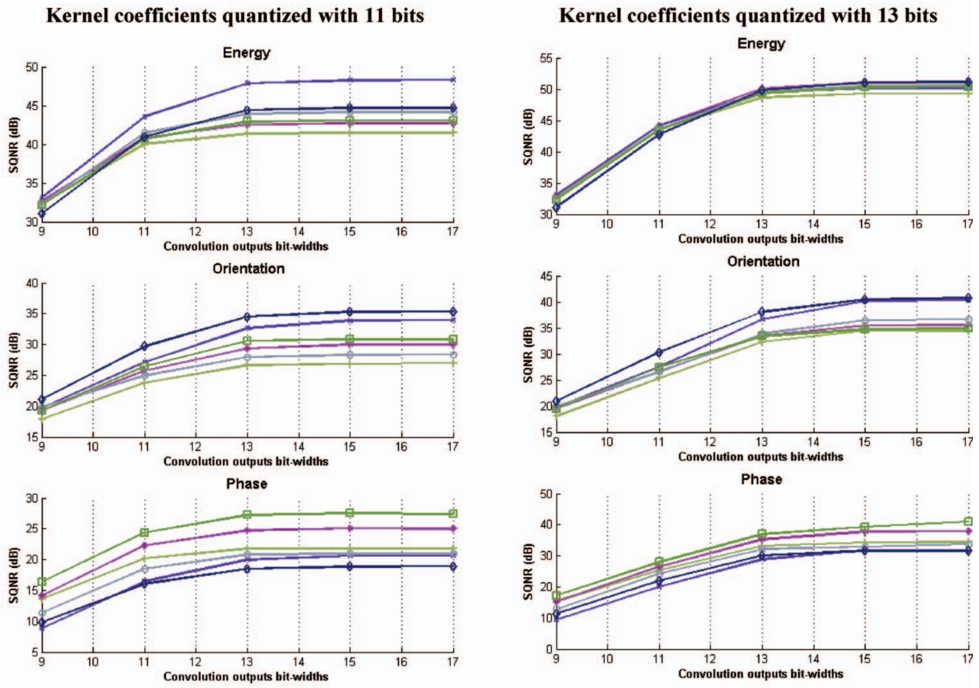
→

Figure 2. Image feature computation with a bank of six images and several data representations and bit-widths. Results obtained without any confidence threshold. (a) Input images chose for the test. (b) SQNR (dB) evolution for different bit-width configurations. The *y*-axis represents the SQNR values, and *x*-axis, the convolution output bit-widths. Figure (b) left and right uses convolution kernel coefficients quantised respectively with 11 and 13 bits. They represent the results obtained for each of the six figures of (a), computed from left to right top to bottom. Six different curves corresponding to each input image are presented at each plot using a marker, respectively the plus sign, asterisk, cross, circles, squares and diamond. Note that although the SQNR is different for each figure, the line trends are the same in all cases. The accuracy differences are produced by the differences in the structure of the images. (c) Image primitives computed using software with double precision floating point representation for the last right-bottom image. This is a synthetic image with multiple orientation and spatial scales that help to illustrate the degradation caused by our bit-width decisions. (d) Image primitives for the synthetic image computed using fixed point data representation with the final system bit-width choices. Energy colourmap encodes contrast variation using warm colours to represent high contrast areas. The orientation direction is encoded on the frame of the picture and phase information uses a colourmap from orange to blue to encode $\pm\ \pi$ values. These images are available in colour online.

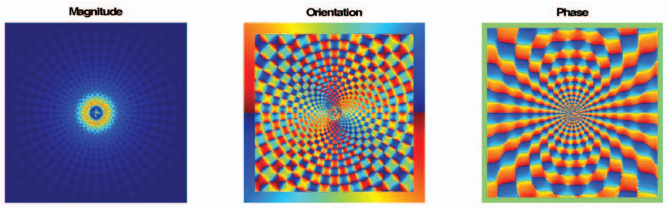
(a)



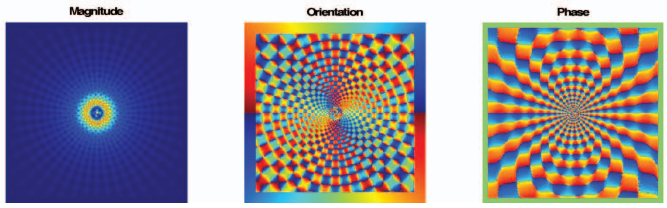
(b)


Kernel coefficients quantized with 11 bits
Kernel coefficients quantized with 13 bits
Energy
Orientation
Phase
SQNR (dB)
Convolution outputs bit-widths

(c)


Magnitude
Orientation
Phase

(d)


Magnitude
Orientation
Phase

estimation. This allows to reduce hardware resource utilisation but forces us to work at high clock frequency values to achieve the same performance. This is not always feasible or could be hard to be put into practice (very fine-grain operation pipelining could be required). Therefore, we decide to implement a one pixel output for clock cycle architecture which is able to achieve high performance even working at low clock frequency values.

In the design process, data dependencies have been eliminated to simplify the processing and also to allow independent processing datapaths [see Equations (9) and (10)]. The main circuit stages are described on Figure 3, each of them is finely pipelined to

Table 1. Energy, orientation and phase SQNR for each test image of Figure 2 computed with fixed point arithmetic.

	SQNR						
	Figure 1	Figure 2	Figure 3	Figure 4	Figure 5	Figure 6	SQNR (mean)
Energy	43.57	44.1	44.11	43.59	43.42	42.73	43.57
Orientation	25.34	26.73	27.47	26.52	27.41	30.27	27.29
Phase	25.33	26.41	20.05	24.26	28.00	22.00	24.34

Mean SQNR values are indicated as global quality reference.

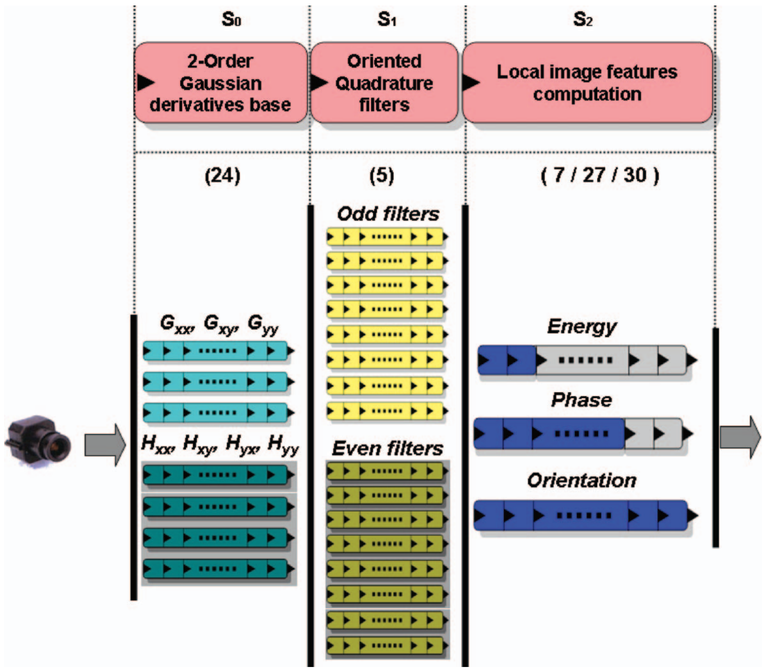


Figure 3. Image feature processing core. Coarse pipeline stages are represented at the top, and long-datapath scalar units, at the bottom. The number of parallel datapaths increases according to the algorithm structure. The whole system has more than 59 pipelined stages (without counting other interfacing hardware controllers such as memory or video input/output interfaces). This allows to compute the three image features at one estimation per clock cycle. The number of substages for each coarse-pipeline stage is indicated in brackets in the upper part of the figure.

achieve our throughput goal. Note that the coarse parallelism level of the circuit is very high (for example, stage S_1 requires 16 parallel paths, eight for the orientation and 2 filter types, odd and even each). Stage S_0 is basically composed of seven separable convolvers corresponding to the convolutions with G_{xx}, G_{xy}, G_{yy}, H_{xx}, H_{xy}, H_{yx} and H_{yy} kernels. The separability property of the convolution operation allows computing first the convolution by rows and after that, by columns. This configuration requires only 8 double port memories shared by all the convolution modules. They are arranged storing one image row at each memory to allow parallel data access at each column. This stage S_0 is finely pipelined in 24 microstages as indicated in brackets in the upper part of Figure 3.

Stage S_1 is used to compute the oriented Quadrature filters. This is achieved by multiplying each output of stage S_0 by some interpolation weights as shown on Equation (5). These coefficients are stored using distributed registers for parallel access. In a sequential approach, they could be stored using embedded memory but the required data throughput makes necessary a full parallel access to all, invalidating therefore this memory-based configuration.

Stage S_2 computes the image features from this set of odd and even complex filters. Figure 3 shows the three different computations which calculate the local features of orientation, phase and energy. Note that the last stage S_2 has different latencies (7, 27 and 30) for each feature (Energy, phase, and orientation). Nevertheless, because we need all of them at the same time, we use delay buffers to synchronise the outputs. The three independent datapaths of stage 3 are described in section 4.1.

4.1. Highly parallel circuits for features interpolation between different orientation-based estimations

Stage S_2 of Figure 3 shows three different datapaths which are described in detail in this section. Because our target system requires to generate a feature estimate per clock cycle, we have focussed on a highly parallel implementation of these circuits.

In Figure 4, the local energy computing architecture which implements the Equation (6) is presented. The system computed the local energy at each orientation and then, using an adder tree, it calculates the total energy value and its mean value dividing it by 8.

In Figure 5, we estimate local orientation. The system takes the local energy estimated at each orientation from the previous described stage. They are weighted using the trigonometric weights described in Equation (9) and finally, the *arctan* function is computed using a CORDIC-based circuit (see a comparison between LUT or CORDIC methods on section 4.2) to get the orientation estimation.

Finally, Figure 6 schematically illustrates the architecture that implements Equation (10) to compute the local phase. This is composed by two adder trees and an arctan computation module.

Note that, due to the different computations executed by each processing unit and the constraint of feature estimation per clock cycle, this leads to different pipeline lengths which motivates the inclusion of the synchronisation buffers shown in Figure 3.

4.2. Analysis of the resources utilisation and performance

For the sake of hardware feasibility, we should take into account the hardware resources to achieve a good trade-off between resource consumption and system accuracy. In section 3 and in Díaz et al. (2007c) we explore the different data bit-width choices and their impact on the final system accuracy. These results are presented in section 3. In addition, we have analysed the resource consumption for different bit-width system configurations. This is graphically presented in the hardware resource *vs*. bit-width evolution shown in Figure 7. We have used as input parameters the convolution output bit-width and scaled the other

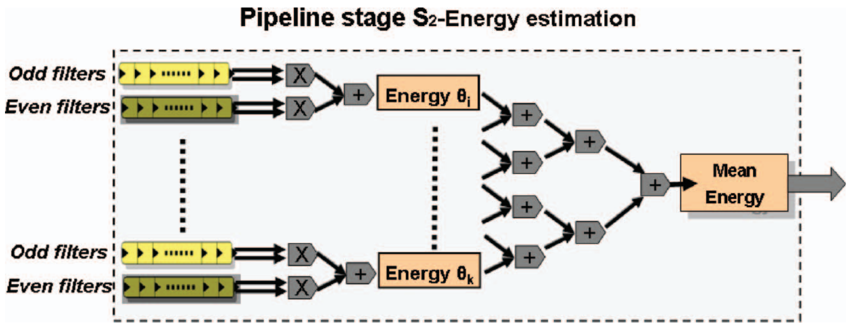


Figure 4. Image features core, stage S_2, Energy computation. This stage corresponds to the implementation of Equation (6). The mean energy is computed using a binary adder pipelined tree. Normalisation (dividing by 8) is computed by shifting operations. All these operations are computed using seven fine pipeline stages but a memory buffer is connected to its output for synchronisation with other image features.

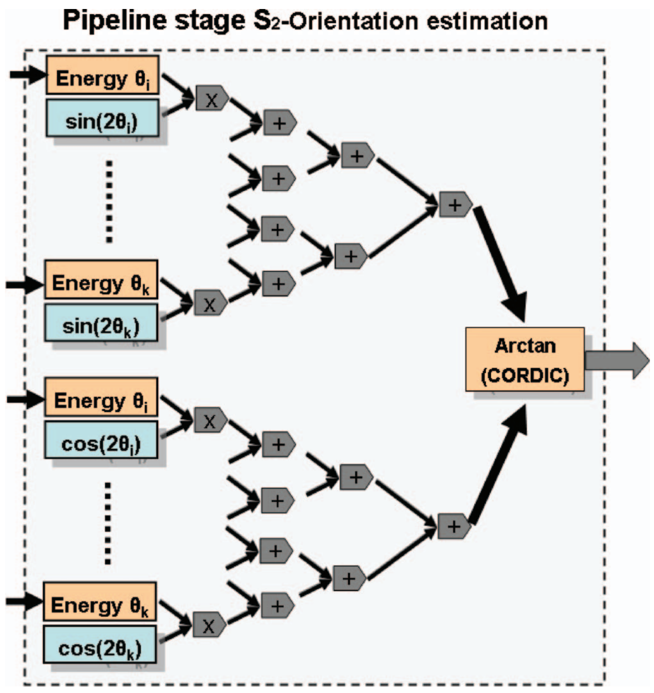


Figure 5. Image features core, stage S_2, orientation computation. This stage corresponds to the implementation of Equation (9). The local energy computed at each orientation on the energy path is the input of this stage. Each oriented energy is multiplied by its own orientation weight and then, it is added using a binary adder's pipelined tree. Final orientation angle is computed using a CORDIC core customised for our application. All these operations are computed using 30 pipeline stages and this is the largest path of stage S_2.

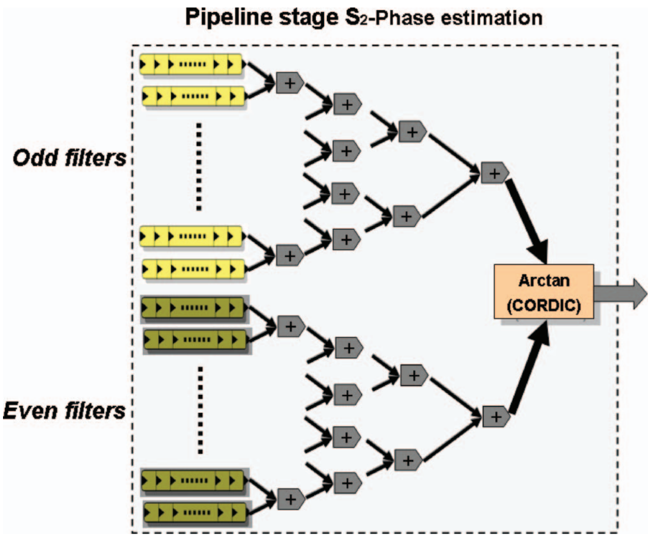


Figure 6. Image features core, stage S_2, phase computation. This stage corresponds to the implementation of Equation (10). The sum of odd and even filters is computed using a binary adder's pipelined tree. Phase angle is computed using a CORDIC core customised for our application. All these operations are computed using 27 pipelines stages, and thus, we need to use a memory buffer connected to its output for synchronisation purposes with other image features.

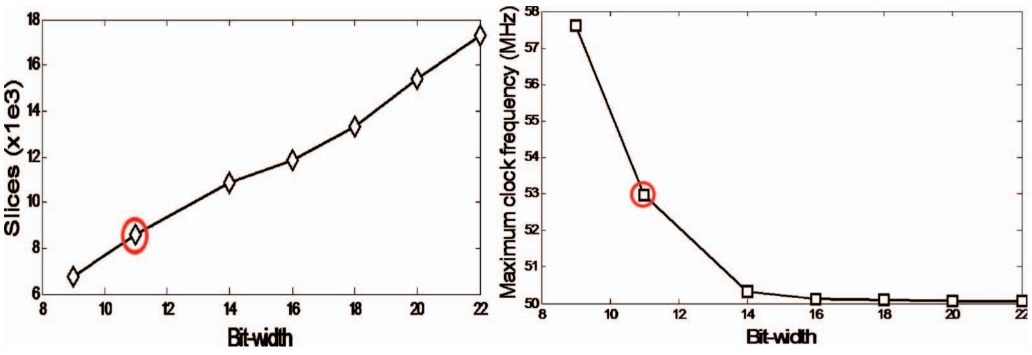


Figure 7. Hardware resource consumption and system clock frequency *vs*. convolution output bit-width. We can address the different configuration studies because the design technique allows to automatically scan data bit-width and to scale changes (just by adjusting some predefined parameters). We have marked with a circle our design choice which corresponds to the configuration described in Section 3.

variable to keep the accuracy. Note that the trade-off depends on the proposed architecture and design parameters. The number of slices grows approximately in an exponential way. This is mainly caused by the non-linear stages whose expansion does not follow a linear tendency. The clock frequency of the system decreases down to a minimum value of (approximately) 50 MHz. This stable minimum can be explained based on the synthesis tool properties. Our design uses the *retiming* capabilities of the DK synthesiser (RC300 2008). This allows redistributing combinational logic across the register's paths to increase the circuit speed. When the bit-width grows, we increase the number of pipeline stages to compensate this effect. For small bit-widths, *retiming* does not produce any effects but, for larger values, *retiming* is able to reduce the combinational paths, increasing the maximum system clock frequency up to this value. Nevertheless, as the frequency change is less than 14% of the peak value, we consider that this is not of significant relevance on the synthesis process and therefore, it can be biased by the parameters of the optimisation tools.

Based on the analysis in Díaz et al. (2007c), summarised on section 3, and the results of Figure 7, we have chosen the bit-widths described in section 3, which are 13 bits for the kernels coefficients and 11 for the convolution outputs. The trigonometric LUT values use 9 bits. Intermediate values are properly scaled to keep the data accuracy. The final orientation and phase results use 9 bits and 22 bits for the energy. In Section 4.1, we commented that *arctan* function (required at the orientation and phase circuits) has been implemented using the CORDIC technique. This is supported by the following results.

We have tested a customised implementation based on LUTs and the CORDIC core taken from the *Coregenerator tool* integrated in the *ISE Foundation of Xilinx* (Xilinx 2007). Our results are summarised in Table 2. Although LUT approach requires fewer resources, it has a lower precision (we use an 8 Ksample LUT, using the symmetries of the arctan function, this circuit has an equivalent data input precision of 15 bits). LUT table extension requires increasing the memory decoding logic which actually affects the speed of the whole circuit (only 42 MHz). This logic can be also pipelined to increase the clock rate but the logic required makes the approach based on the CORDIC core more suitable for our implementation. LUT approach can be improved by samples interpolation or by techniques as presented in (Sasao, Nagayama and Butler 2007) which could be used to reduce table size and logic. We have not explored these alternatives because the CORDIC technique fulfills our requirements, but this could be done as one of the future system improvements.

The whole architecture has been implemented on the RC300 board (see RC300 2008). This prototyping board is provided with a Virtex II XC2V6000-4 Xilinx FPGA as processing element, including also video input/output circuits and user interfaces/ communication buses. The final required hardware consumption and performance measures of our implementation are shown in Table 3.

Detailed information about the image features architecture and the coarse pipeline stages are shown in Table 4. Note that the sum of the partial stages is not the total number of resources. This can be easily explained based on the synthesis tools. The DK synthesiser (RC300 2008) does a flat expansion before addressing a global optimisation and synthesis. For the whole system, it is more likely that the synthesiser engine can share some resources and reduce hardware consumption. This also explains why the whole system has a lower

Table 2. *Arctan* function implementation approaches.

Method	Slices	EMBs	Multipliers	$f_{\text{clk\_max}}$ (MHz)
CORDIC core	1020	0	0	118
LUT-Arctan	841	5	0	42

CORDIC core uses data input with 21 bits. The LUT uses 8 Kwords to sample the *arctan* function. Decision logic allows to extend the range from the sampled interval [0, $\pi/2$] to the whole circumference.

Table 3. Complete system resources required for the local image features computing circuit.

Slices (%)	EMBS (%)	Embedded multipliers (%)	Mpps	Image resolution	Fps
9135 (27)	8 (5)	65 (45)	56.5	1000 × 1000	56.5

The circuits have been implemented on the RC300 prototyping board (RC300 2008). The only computing element is the Xilinx FPGA Virtex II XC2V6000-4. The system includes the image features processing unit, memory management unit, camera Frame-grabber, VGA signal output generation and user configuration interface. Results obtained using the Xilinx ISE Foundation software (Xilinx 2007). Mpps, mega-pixels per second at the maximum system processing clock frequency; EMBS, embedded memory blocks.

Table 4. Partial system resources required on a Virtex II XC2V6000-4 for the coarse pipeline stages described for this circuit.

	Circuit stage	Slices (%)	EMBS (%)	Embedded multipliers (%)	f_{clk} (MHz)
S_0	Gaussian base convolutions	4170 (12)	8 (5)	50 (34)	85
S_1	Oriented quadrature filters	1057 (3)	0	0	69
S_2	Features Energy, Phase, and Orientation computation	2963 (8)	0	6 (4)	89
	Whole processing core	7627 (22)	8 (5)	65 (45)	58.8

Results obtained using the Xilinx ISE Foundation software (Xilinx 2007). EMBS, embedded memory blocks.

clock frequency than the slowest subsystem (stage S_1 runs at 69 MHz and the whole core only at 58 MHz). By modular code stages synthesis, this effect can be avoided. Furthermore, more pipelining could be used to increase the system performance and produce a very fine-grain pipelined architecture. Our system is fine-grain in the sense that it uses one clock cycle per operation but some arithmetic operations have long combinational paths which reduce the global system clock. Because of the fact that the current performance already fulfils our requirements, we have left out these optimisations, but further code modularisation and very fine-grain arithmetic unit pipelining could be done if required.

We have not explained yet why the number of multipliers grows on the complete system with respect to the sum of the multipliers used at each substage. We have used automatic inference of multipliers on the system. As the number of multiplications by constants is high, it is difficult to manually determine the time when it is better to use an embedded multiplier or to compute it using FPGA logic. Synthesis tools define cost functions that are able to evaluate this based on technological parameters for each multiplication, which is the best option. We have compared the manual and the automatic inference of multipliers and, though differences are not large, automatic inference usually achieves higher clock rates than manual generation with slightly less logic. This automatic inference changes depending on the circuit that is being synthesised and produces the differences presented in Table 4.

The final qualitative results for a test image are shown in Figure 8. Though there are significant differences in the bit-width between the software and hardware data, the high SQNR guarantees high accuracy with imperceptible degradation.

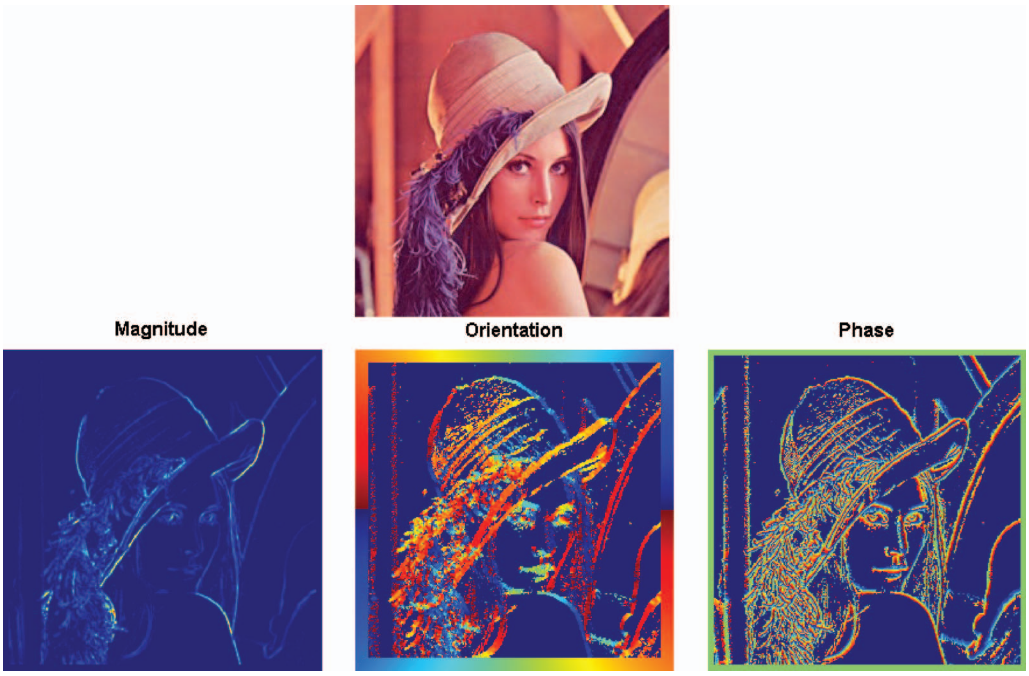


Figure 8. System results for the known image of Lena (top image). The three bottom pictures present the computed features for this image using the same colour encoding used in Figure 2. Note that, although restricted fixed-point arithmetic is used, the quality of the features is high as can be seen looking at small details such as the hat's plumes.

5. Conclusions

In this contribution, we have presented a hardware system capable of computing into the same chip local image energy, phase and orientation using reconfigurable hardware. The system uses Second Order Gaussian derivatives to implement hardware friendly Quadrature filters. A modification of the Haglund technique (Haglund 1992) is used for features interpolation between the different orientations.

Our target goal has been to achieve a very high performance which is required for the target specifications addressed, for instance, on complex vision projects such as DRIVSCO (DRIVSCO 2007). To get that, a superscalar (understanding as such as parallel replication of processing units) and long-datapath pipelined computing system has been designed. A careful utilisation of the different stage data word-lengths allows us to arrive at a high accuracy system with very low degradation due to the fixed-point arithmetic quantisation effects at a reasonable system hardware resource utilisation.

The final system is able to compute more than 56 Mpixels per second using less than 22% of the resources of a Virtex II XC2V6000. The image resolution can be adapted to each target application. The superscalar and long-datapath pipelined architecture proposed fulfills our requirements and illustrates that, even with a low clock rate, a well defined architecture can achieve high computing performance unviable with current general purpose processors.

Future work will address the use of this system as a first stage of stereo and motion computing system. Furthermore, we plan to study the implementation of this system using image analysis at different resolutions, trying to properly sample the spatial frequency spectrum of the images. Finally, we also plan to evaluate different resource sharing techniques to reduce the system utilisation resources.

Acknowledgements

This work has been supported by the VI EU research framework funds through the European Project DRIVSCO (IST-016276-2), the National Spanish Grant DEPROVI (DPI2004-07032) and Junta de Andalucía Project: P06-TIC-02007.

References

Agility (2008), *"DK Design Tool."* http://www.agilityds.com/products/c_based_products/dk_design_suite/default.aspx

Bernardino, A., and Santos-Victor, J. (2005), "A Real-Time Gabor Primal Sketch for Visual Attention," *Lecture Notes in Computer Science*, 3522, 335–342.

Bonato, V., Fernández, M.M., and Marques, E. (2006), "A smart camera with gestures recognition and SLAM capabilities for mobile robots," *International Journal of Electronics*, 93, 385–401.

Bouganis, C.S., Cheung, P.Y.K., Ng, J., and Bharath, A. (2004), "A Steerable Complex Wavelet Construction and Its Implementation on FPGA," *Lecture Notes in Computer Science*, 3203, 394–403.

Cheung, O.Y.H., Leong, P.H.W., Tsang, E.K.C., and Shi, B.E. (2006), "A scalable FPGA implementation of cellular neural networks for Gabor-type filtering," in *Proceedings of International Joint Conference on Neural Networks*, pp. 15–20.

Clark, C.R., Ulmer, C.D., and Schimmel, D.E. (2006), "An FPGA-Based Network Intrusion Detection System with On-Chip Network Interfaces," *International Journal of Electronics*, 93, 403–420.

Darabiha, A., Maclean, W.J., and Rose, J. (2006), "Reconfigurable Hardware Implementation of a Phase-Correlation Stereo Algorithm," *Machine Vision and Applications*, 17, 116–132.

Díaz, J. (2006a), "Multimodal Bio-Inspired Vision System. High Performance Motion and Stereo Processing Architecture". Ph.D. Thesis, University of Granada.

Díaz, J., Ros, E., Ortigosa, E.M., and Mota, S. (2006b), "FPGA Based Real-Time Optical-Flow System," *IEEE Transactions on Circuits and Systems for Video Technology*, 16, 274–279.

Díaz, J., Ros, E., Prieto, A., and Pelayo, F.J. (2007a), "Fine Grain Pipeline Systems for Real-Time Motion and Stereo-Vision Computation," *International Journal of High Performance Systems Architecture*, 1, 60–68.

Díaz, J., Ros, E., Carrillo, R., and Prieto, A. (2007b), "Real-Time System for High-Image-Resolution Disparity," *IEEE Transacitons on Image Processing*, 16, 280–285.

Díaz, J., Ros, E., Mota, S., and Carrillo, R. (2007c), "Image Processing Architecture for Local Features Computation," *Lecture Notes in Computer Science*, 4419, 259–270.

DRIVSCO Project (2007), http://www.pspc.dibe.unige.it/~drivsco/

Felsberg, M., and Sommer, G. (2001), "The monogenic signal," *IEEE Transactions on Signal Processing*, 49, 3136–3144.

Fleet, D.J., Jepson, A.D., and Jenkin, M.R. (1991), "Phase-Based Disparity Measurement," *CVGIP: Image Understanding*, 53, 198–210.

Fleet, D.J. (1992), *Measurement of Image Velocity*, Norwell, MAEngineering and Computer Science, Kluwer.

Fleet, D.J., and Jepson, A.D. (1993), "Stability of Phase Information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15, 1253–1268.

Freeman, W., and Adelson, E. (1991), "The Design and Use Of Steerable Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13, 891–906.

Frigo, M., and Johnson, S.G. (2007), , "FFTW", http://www.fftw.org

Gautama, T., and Van Hulle, M.M. (2002), "A Phase-Based Approach to the Estimation of the Optical Flow Field Using Spatial Filtering," *IEEE Transactions on Neural Networks*, 13, 1127–1136.

Granlund, G.H. (1978), "In Search of a General Picture Processing Operator," *Computer Graphics and Image Processing*, 8, 155–173.

Haglund, L. (1992), "Adaptive Multidimensional Filtering," PhD. thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden.

Kerins, T., Marnane, W.P., and Popovici, E.M. (2006), "Algorithms and Architectures for a Flexible Elliptic Curve Cryptography Processor," *International Journal of Electronics*, 93, 349–372.

Kovesi, P. (1999), "Image Features from Phase Congruency," *Videre (Journal of Computing Vision Research),* Vol. 1, number 3, http://mitpress.mit.edu/e-journals/Videre/001/v13.html

Kubota, T., and Alford, C.O. (1997), "Computation of Orientational Filters for Real-Time Computer Vision Problems. III. Steerable System and VLSI Architecture," *Real Time Imaging*, 3, 37–58.

Krüger, N., and Felsberg, M. (2004), "An Explicit and Compact Coding of Geometric and Structural Information Applied to Stereo Matching," *Pattern Recognition Letters*, 25, 849–863.

Krüger, N., and Wörgötter, F. (2002), "Multi Modal Estimation of Collinearity and Parallelism in Natural Image Sequences," *Network: Computation in Neural Systems*, 13, 553–576.

Mota, S., Ros, E., Díaz, J., Ortigosa, E.M., and Prieto, A. (2005), "Motion-Driven Segmentation by Competitive Neural Processing," *Neural Processing Letters*, 22, 125–147.

RC300 (2008), "RC300/RC340 Board Information," http://www.agilityds.com/products/c_based_products/re_compiting_platform/rc340/default.aspx

Sabatini, S.P., Gastaldi, G., Solari, F., Pauwels, K., van Hulle, M., Díaz, J., Ros, E., Pugeault, N., and Krüger, N. (2007), "Compact (and accurate) early vision processing in the harmonic space," in *Proceedings of the Second International Conference on Computer Vision Theory and Applications (VISAPP)*, Barcelona (Spain), March 8–11, pp. 213–220.

Sasao, T., Nagayama, S., and Butler, J.T. (2007), "Numerical Function Generators Using LUT Cascades," *IEEE Transactions on Computers*, 56, 826–838.

Solari, F., Sabatini, S.P., and Bisio, G.M. (2001), "Fast Technique for Phase-Based Disparity Estimation with no Explicit Calculation of phase," *Electronics Letters*, 37, 1382–1383.

Voß, N., and Mertsching, B. (2001), "Design and implementation of an accelerated gabor filter bank using parallel hardware," in *Proceedings of the 11th International Conference on Field-Programmable Logic and Applications,* August, pp. 451–460.

Volder, J. (1959), "The CORDIC Trigonometric Computing Technique," *IRE Transactions on Electronic Computing*, EC-8, 330–334.

Xilinx (2007), "ISE Foundation software." http://www.xilinx.com/ise/logic_design_prod/foundation.htm