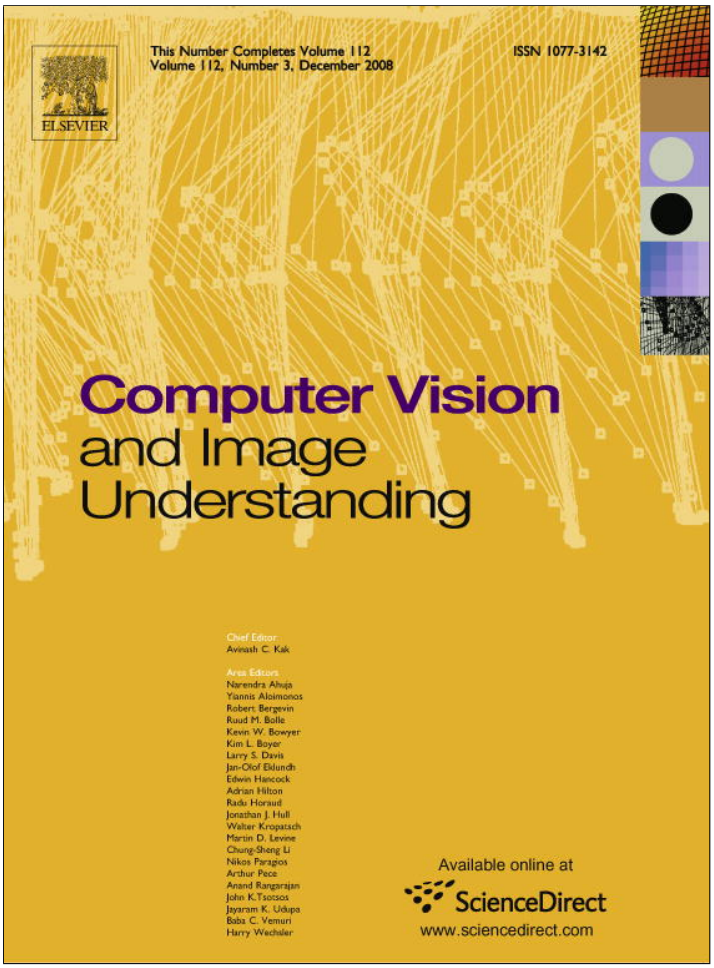

Computer Vision and Image Understanding 112 (2008) 262–273

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Superpipelined high-performance optical-flow computation architecture

Javier Díaz *, Eduardo Ros, Rodrigo Agís, Jose Luis Bernier

Department of Computer Architecture and Technology, University of Granada, E.T.S.I. Informática, C/Periodista Daniel Saucedo, s/n., E-18071 Granada, Spain

ARTICLE INFO

Article history:
Received 14 November 2007
Accepted 6 May 2008
Available online 27 May 2008

Keywords:
Image motion analysis
Real-time systems
FPGAs
Architecture of vision systems

ABSTRACT

Optical-flow computation is a well-known technique and there are important fields in which the application of this visual modality commands high interest. Nevertheless, most real-world applications require real-time processing, an issue which has only recently been addressed. Most real-time systems described to date use basic models which limit their applicability to generic tasks, especially when fast motion is presented or when subpixel motion resolution is required. Therefore, instead of implementing a complex optical-flow approach, we describe here a very high-frame-rate optical-flow processing system. Recent advances in image sensor technology make it possible nowadays to use high-frame-rate sensors to properly sample fast motion (i.e. as a low-motion scene), which makes a gradient-based approach one of the best options in terms of accuracy and consumption of resources for any real-time implementation. Taking advantage of the regular data flow of this kind of algorithm, our approach implements a novel superpipelined, fully parallelized architecture for optical-flow processing. The system is fully working and is organized into more than 70 pipeline stages, which achieve a data throughput of one pixel per clock cycle. This computing scheme is well suited to FPGA technology and VLSI implementation. The developed customized DSP architecture is capable of processing up to 170 frames per second at a resolution of 800 × 600 pixels. We discuss the advantages of high-frame-rate processing and justify the optical-flow model chosen for the implementation. We analyze this architecture, measure the system resource requirements using FPGA devices and finally evaluate the system's performance and compare it with other approaches described in the literature.

1. Introduction

Optical-flow is a well-known technique used to recover 2D motion from image sequences. Different approaches to the subject include image block-matching, gradient constraints, phase conservation and energy models [1]. Until now most comparative studies have focused on the different estimation approaches and their accuracy [1,2]. Some of them have also gone on to cover implementation feasibility [3]. These studies have usually taken synthetic sequences with known ground-truth flow fields to compare their accuracy with different types of motion. They have concluded that greatest accuracy is achieved when using phase-based and differential methods but, though these models work well for low-velocity motion, they fail when trying to estimate fast motion, i.e. their accuracy is significantly degraded due to temporal aliasing [4,5]. In these cases, phase-based and differential models use complex methods and typically multi-scale approaches [4,6–8] to improve their estimations. Another valid alternative is that of block-matching approaches, which are quite capable of computing fast motion but with high computational cost and usually without subpixel accuracy. This feature makes them interesting for real applications but we do not include them in the following discussion because their flow is not very accurate [1]. A third option is to combine the properties of block-matching and differential methods [9] to take advantage of the positive features of both approaches. Finally, some authors have combined multiscale approaches and iterative-diffusion methods to improve flow accuracy and the range of velocities [10]. It is important to note that temporal aliasing is a complex problem in which we cannot separate the temporal sampling rate and the image structure. According to the Nyquist–Shannon theorem (discussed in [5]), spatial frequency is very important in calculating the maximum speed that can be recovered from an image sequence.

These approaches partially overcome the problem of fast motion but their real-time implementation is still beyond the possibilities of current technology (at least at affordable system costs). Furthermore, these models require very complex architectures that imply very high hardware resource consumption to achieve real-time processing, and in some cases their inherent parallelism is constrained, by the warping method in multiscale approaches for instance, or due to the use of iterative algorithms. Our approach focuses on an alternative method which involves the

* Corresponding author. Fax: +34 958 248993.
E-mail addresses: jdiaz@atc.ugr.es (J. Díaz), eros@atc.ugr.es (E. Ros), ragis@atc.ugr.es (R. Agís), jbernier@atc.ugr.es (J.L. Bernier).

1077-3142/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2008.05.006

implementation of a very regular motion–estimation approach that allows the efficient use of high-frame-rate cameras. Advances in imaging-sensor technology have made it possible to acquire images at a very high-frame-rate (for instance, see products from [11–13]). The use of such hardware reduces the range of motion in the video sequences, thus allowing the simplification of the optical-flow models and increasing the accuracy of the system [5]. The specific-purpose computing architecture described here is able to compute at significantly higher frame rates than the standard 25 frames per second (fps), allowing us to take advantage of modern sensor capabilities.

In various previous works, the Lucas & Kanade (L&K) approach [1,14] has been highlighted as a good candidate to be implemented on hardware with affordable resources [2,3,15,16]. A comparison of L&K with other differential approaches [17,18] (also of feasible hardware implementation [19]) concludes that the L&K least-squares-fitting approach results in the greatest accuracy. From our previous experience [16,17], and also according to [5], we conclude that to achieve accurate estimations with this model a high acquisition and processing frame-rate is desirable. This motivates the development of a high-frame-rate optical-flow computing system. Although in previous works we have described a system capable of processing 41 kilopixels per second (Kps) [15,16], the approach presented here adopts a different algorithmic strategy according to the modifications proposed in [20]. Furthermore, we implement here a novel superpipelined processing architecture capable of computing one pixel per clock cycle. We have had to undertake a deep analysis of the circuit arithmetic to achieve accurate results with affordable hardware resources. These innovative aspects result in our approach outperforming any previous system described in the literature by one order of magnitude. This performance allows real-time processing of over-sampled frame rates, which opens the door to the use of advanced image sensors in real-time motion–estimation systems and their potential applications.

The challenge is to design a novel architecture capable of processing optical-flow at over-sampled frame rates. The state-of-the-art processing architectures (see Section 3.5) are unable to process 640×480 resolution images (we will assume this resolution in the rest of the paper unless explicitly mentioned) at frame rates higher than 13 fps for subpixel methods or 26 fps for correlation-based approaches (see Table 4 in Section 3.5) and therefore a novel design strategy is needed. In the following sections we describe the architecture of a customized DSP designed for FPGA with this aim in mind. We illustrate how our superpipelined architecture outperforms the fastest processing system described to date by more than one order of magnitude. In Section 3.4, we evaluate the system resource's cost and the performance of the final system.

2. Description of the model

In our discussion, we have presented the L&K model as being a good candidate for real-time optical-flow computation but one important point must be born in mind: most of the evaluations and tests of this model use the implementation described by Barron et al. [1], which is open to significant improvement. Brandt [20] explores this topic in a study in which he demonstrates that a suitable choice of the different parameters improves flow accuracy and density significantly. Furthermore, other publications reveal that commonly used optical-flow measurements [1] are biased due to errors in numerical differences [18,20].

In Appendix A, we describe briefly the computations upon which the L&K approach is based and the adopted modifications of the system parameters according to [20]. Appendix B reviews 3D spatio-temporal sampling theory and investigates the effects of motion aliasing as well as the main limitations of the L&K model. Readers not familiar with these topics may find these appendixes useful for a better understanding of the motivation and specific motion-estimation model addressed in this work.

2.1. High-frame-rate system motivation

The analysis in Appendix B calls for the use of a first-order Gaussian-derivative kernel of 5 taps. According to the sampling theorem and as described in [5] we cannot reliably compute the velocity of objects with a high spatial-frequency content; we can only compute fast motion for low spatial-frequency objects using complex multiscale methods, as described in [4,6–8]. Therefore, we propose to increase the temporal sampling frequency to recover fast motion patterns and improve model assumptions.

We have commented in Section 1 upon the availability of image sensors with very high frame rates at different image resolutions. The combination of such high-frame-rate image sensors and a specific processing system capable of computing the incoming data stream is of considerable interest for a wide range of real-world applications. The main improvements to such a system are the following:

1. The processing scheme is simplified, avoiding complex multiscale approaches which require equally complex architectures and translate into higher system costs. The improved version of the L&K optical-flow model that we have adopted combines high accuracy and implementation feasibility [3,15,16].
2. Temporal aliasing is reduced significantly through high-frame-rate sampling. This allows the computation of high-spatial-frequency contents of image motion, thus increasing the density of the flow field.
3. The constant luminance condition is better satisfied [5] through high-frame-rate sampling. Therefore the first-order constraint is better satisfied, thus improving the accuracy.

To date there have only been a few approaches capable of processing optical-flow in real-time at standard video rates of up to 30 fps, and faster processing systems are required to deal with high frame rates in real-time. Nevertheless, even though we are able to compute flow at one order of magnitude faster than previous approaches, there is a trade-off between flow accuracy and camera image signal-to-noise ratio (SNR). SNR is usually proportional to the square root of the exposure time. Thus, a higher frame rate implies a lower exposure time, which can increase noise dramatically.

As shown in [5], an over-sampled factor of 3 (90 fps) increases the accuracy of motion-estimation models dramatically but this strategy is not recommended for factors higher than 6 due to the degradation of the image SNR.

In Fig. 1, we show the qualitative results of an optical-flow sequence computed at 90, 30 and 10 fps using a progressive area scan CCD sensor. Because the over-sampling factor is small, as a first approximation we consider image SNR to be constant at the different frame rates. The results shows that the flow computed at 90 fps is very homogeneous and stable and the confidence areas clearly identify the motion boundary, which corresponds with the areas of higher spatio-temporal structure. The results computed at 10 fps are quite noisy, which may well derive from the reliability areas of the third row. Note that at 30 fps the flow quality is significantly degraded (compared with the 90 fps flow) and the fastest movements (the left arm pattern for instance) are lost. This demonstrates the importance of having a computing system capable of processing the data stream of 90 fps sequences.

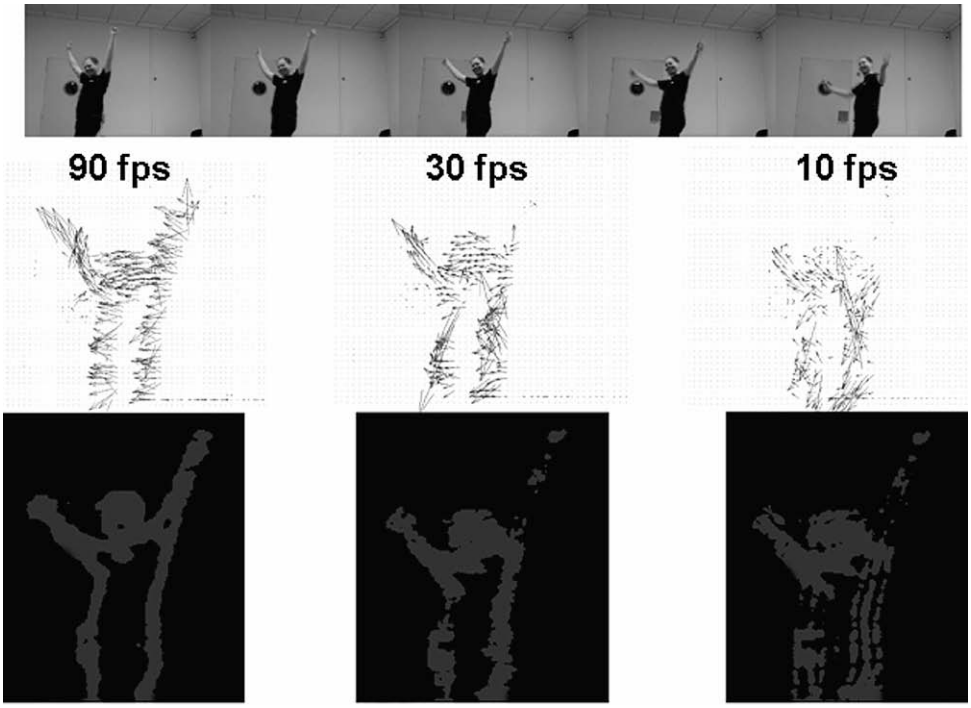

Fig. 1. Qualitative effects of different frame-rate sequence acquisitions. The top row shows the example of a "walking and raising the arms" sequence captured at 90 fps. The low-level structure of the clothes allows us to focus on the motion at the body boundaries. The second row shows three optical-flows, computed at 90, 30, and 10 fps using sequence subsampling. The third row shows the pixels over the confidence threshold for the sequences computed at the three different frame rates. Note that although the walking movement is slow (the camera-to-object distance is approximately 4 m), the system is unable to compute its motion pattern at 10 fps. At 30 fps the flow is still noisy and some important details (such as the left arm pattern) are lost.

2.2. Target applications

The high-frame-rate computation has two important implications. First, this allows us to achieve very high optical-flow accuracy, which is of considerable interest. Second, the system image latency is significantly reduced (for instance a system running at 25 fps with 3 frame latency has a delay of 120 ms, whilst when running at 125 fps the delay is 24 ms).

The first property (high accuracy) is important for applications that require the use of motion as input for further processing to extract higher level sequence information. Highly accurate motion is required for higher level tasks such as motion-in-depth estimation [21], structure from motion [22], the computation of independent moving objects [23] and accurate tracking [24]. The final system performance of these techniques dependents very much upon the accuracy of the input flow field because errors lead to wrong higher level estimations (since some of these tasks depend on local variances of the input optic-flow).

The second property (low latency) is of crucial importance for physical systems which interact with other agents in the real-world. For instance, in dynamic scenarios where an autonomous agent has to close the perception action loop [25], long latencies make grasping operations more unstable. Furthermore, tasks such as autonomous navigation also require low latency motion in order to avoid collisions and properly compute time-to-contact [26] on the loop. This property is of critical importance in the field of advanced driver assistance systems for vehicular technology, as is shown in [27] (the warning distance is closely related to system latency, making this a very critical factor for the viability of the final system). Finally, other examples of target scenarios are in the field of augmented view applications, which also require low latency. For instance, some vision-aid systems need to fuse the input images dynamically with the processed ones and display the results to the vision impaired subject by using a head-mounted display [28]. If motion cues are going to be used for improving image information but the latency is high, the results will be difficult to integrate with the remaining visual capabilities of the subject due to the temporal delay of the extracted cues. This will make augmented-vision aids based on long-latency processing engines useless.

The previous examples represent several fields of applications where the system described here is of interest.

3. Hardware architecture

Standard PC processors nowadays have considerable computing power for image processing thanks to high system clock frequency and MMX and SSE instruction extensions, which allow them to exploit their DSP capabilities. Nevertheless, although there are some optical-flow approaches running on software in near real-time [29], the intensive computation required to process optical-flow still renders it unrealistic to process over-sampled sequences in real-time. DSPs are suitable for embedded image-processing applications but their computing performance is also below the over-sampled rate. Therefore, we consider reconfigurable hardware, which has already shown their suitability in diverse applications, to be a suitable option [30]. On the basis of previous experience [15,16] the main theoretical drawback to this technology is the limited system clock frequency compared to processors but, as we shall see, the efficient use of the pipeline and parallel computing resources available on such devices allows us to achieve the outstanding performance presented here.

3.1. Coarse-pipelined optical-flow architecture

The overall concept is a pipelined architecture (Fig. 2) and the basic computational stages represent the different steps of the

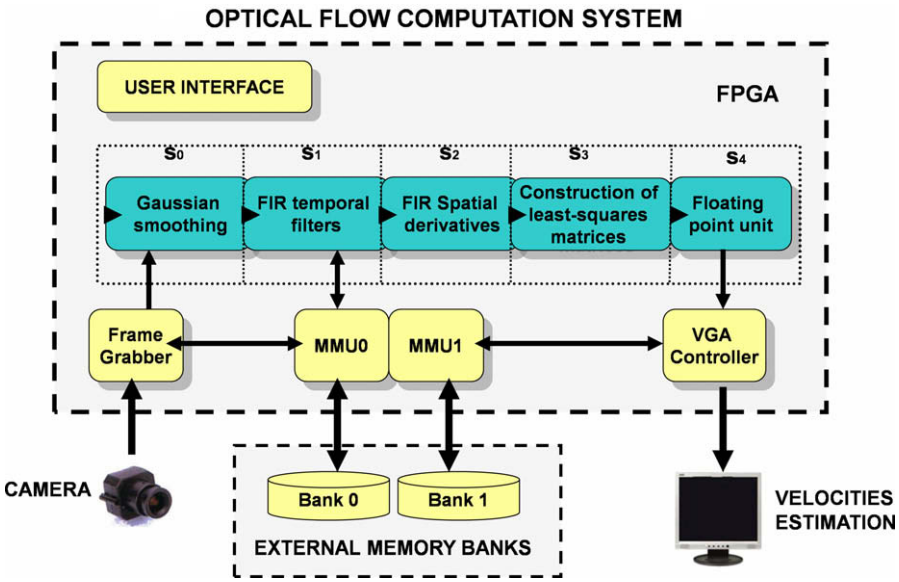

Fig. 2. Optical-flow system structure. The thin dotted line marks the processing core. Light-grey blocks indicate external memories and hardware controllers inside the FPGA. The user interface consists of a LCD display plus mode-selection buttons. All the computations are made inside the FPGA device.

L&K algorithm. The system has been designed as an embedded processing platform that can be used on mobile applications and thus we have designed the user interface, hardware controller for memory, VGA visualization and input camera interface for the same FPGA device. This strategy means that the system can be used in combination with a high-frame-rate camera as a stand-alone processing platform for various potential applications. The scheme of the entire system is shown in Fig. 2. Note that the thin dotted line marks the optical-flow processing core, the functional stages of which can be summarized as follows:

- S_0. Gaussian-filter smoothing stage.
- S_1. The FIR temporal filter computes temporal-derivative and space-time smoothed images.
- S_2. Spatial derivatives and complementary Gaussian filtering operations.
- S_3. Construction of least-square matrices for the integration of neighborhood velocity estimations.
- S_4. Custom floating-point unit. The final velocity estimation requires the computation of a matrix inversion, which includes a division operation. Although the previous stages use fixed-point arithmetic, in S4 the resolution of the incoming data bits is significant and expensive arithmetic operations are required. Thus, fixed-point arithmetic becomes unaffordable, prompting us to design a customized floating-point unit.

The frame-grabber buffers the inputs from the camera to allow for the difference in the clock rates between the input circuitry (pixel acquisition) and processing engine. The MMU handles this buffering and reads the pixels needed to be buffered. Since internal memory of the FPGA is very limited, only a few image lines are stored at each processing stage. For 2D convolution operation we use a memory buffer of kernel length minus one number of pixels for the horizontal kernel. For the vertical kernel we need once more to store a number of columns equal to the kernel length minus one. In our design horizontal convolution buffers are implemented using small distributed registers. The vertical convolution operation uses embedded memory blocks for data buffering. This leads to the following internal FPGA memory usage: we store four rows for image smoothing, 12 rows for image derivative computations and 24 rows for the least-squares-fitting stage. Due to the organization of the FPGA's internal memory use is fixed for address depths up to 1024 elements. Image resolutions with more than 1024 pixels per column need to duplicate the internal memory usage (allowing an image resolution of up to 2048 pixels per column). Nevertheless, this represents no critical constraint since, as shown in Section 3.4, we have plenty of resources in our target chip.

Although we maintain the coarse structure described in [15] and [16], each stage of our new architecture has been carefully redesigned. Below we highlight the points that are completely novel with respect to our previous works:

We have adopted improved optical-flow parameters according to [20] and designed 3D complementary smoothing-derivative FIR filters based on Simoncelli kernels [31] to improve the derivation operations. This structure is shown in Fig. 3.

In previous works we used Fleet & Langley [32] IIR filters for optical-flow computation, which are more hardware-friendly than FIR ones because they reduce the system memory requirements to three images. The drawback of this approach is that due to the iterative process required for IIR operations fixed-point arithmetic magnifies errors, leading to a significant degradation in temporal derivative accuracy. The modifications adopted from [20] allow the use of smaller FIR filters in our new approach, which makes this option more reliable.

We have implemented a superpipelined processing architecture able to compute one pixel per clock cycle, as described in the next section.

3.2. From coarse to superpipeline architecture

The previous scheme, with a pipelined structure divided into 5 basic stages, would give a good performance but still far from high-frame-rate processing requirements. The main reason is that the architecture in Fig. 2 is similar to a DSP processor. There is a trade-off between pipeline length and system performance based on dependence problems (branch conditions often break the pipeline, causing a significant waste of time). For this reason long pipelines are not included in standard DSPs and microprocessors. We describe here, therefore, a specific-purpose processing architecture that benefits greatly from a fine-grain pipeline datapath. In this way we take full advantage of the regular data flow of the L&K

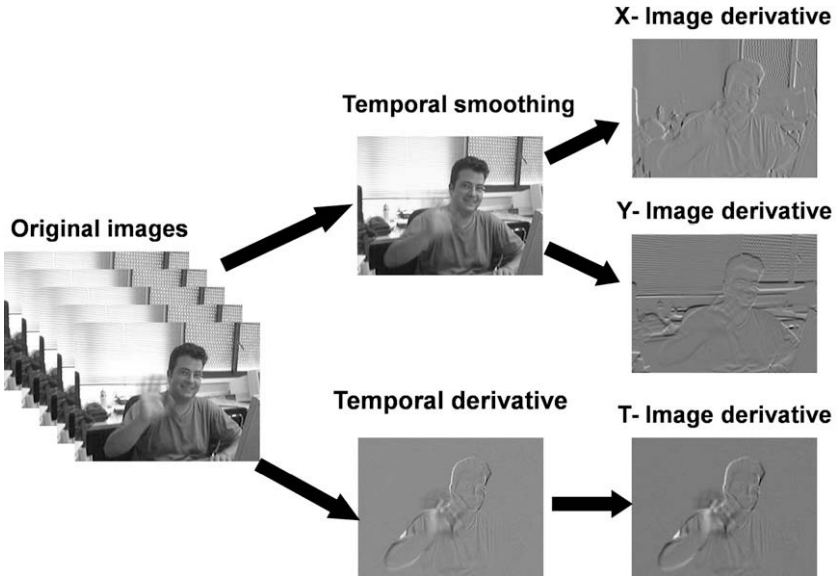

Fig. 3. Complementary 3D smoothing-derivative filter structure. There are two basic primitives corresponding to the smoothing and derivation operation. First, we compute these two primitives in the temporal domain and then complementary smoothing/derivation is carried out in the spatial dimension. The final results correspond to a 3D derivation over each spatio-temporal axis. For a high-performance design, all the operations (2 temporal and 6 spatial FIR filtering) are processed in parallel based on separable convolutions.

algorithm. According to [33], the best architecture is a superscalar, superpipelined structure. This design strategy has been adopted and the resulting system allows our system to be used as an embedded coprocessor in the resolution of other problems, for inclusion in mobile devices such as the one described in [34] for instance. In Fig. 4 we present the overall scheme. Each coarse stage has been finely pipelined, leading to a processing datapath of more than 70 stages. The number of scalar units grows in stages required by the L&K model to maintain system throughput. This expansion of parallelism is outlined in the following:

- Stage S_0 uses one scalar unit for spatial smoothing.
- Stage S_1 uses two scalar units, one for temporal smoothing and another for temporal differentiation.
- Stage S_2 uses three scalar units; corresponding to the three dimensions I_x, I_y, and I_t in which the image derivatives are computed.
- Stage S_3 uses five scalar units, corresponding to the five cross-products $I_x \bullet I_x$, $I_y \bullet I_y$, $I_x \bullet I_y$, $I_x \bullet I_t$ and $I_y \bullet I_t$, which are used in the weighted-squared sum during the least-squares-fitting stage of the L&K approach (see Appendix A).
- Finally, stage S_4 uses one scalar unit to compute the final motion for each pixel but internally several parallel pathways drive the data process. Therefore, this stage must be seen as a superscalar floating-point unit.

The legend to Fig. 4 indicates the internal data representation of the scalar units. The number of parallel units is driven by the

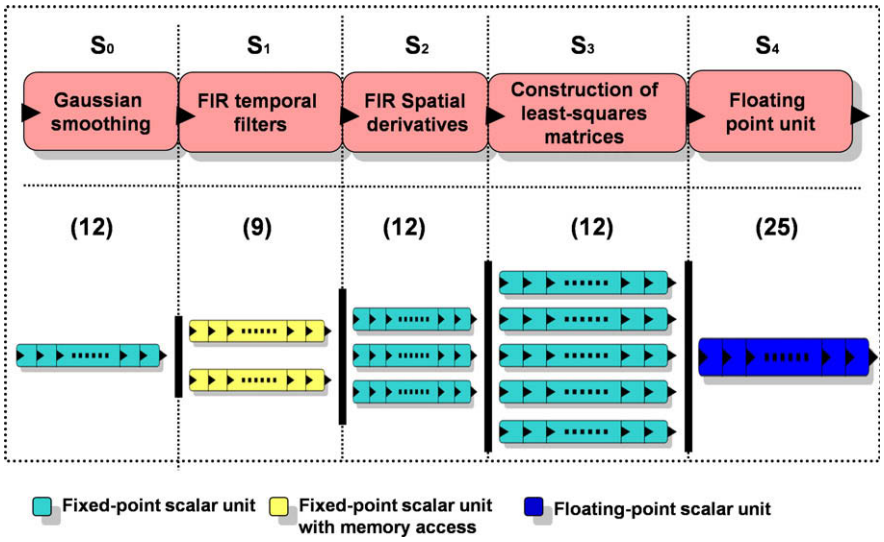

Fig. 4. Optical-flow processing core. Coarse pipeline stages are represented at the top and superpipelined scalar units at the bottom. Gray levels indicate the type of scalar units according to the legend at the bottom of the figure. The number of parallel datapaths increases according to the structure of the optical-flow algorithm. The whole system has more than 70 pipelined stages (counting the motion-processing core and other interfacing hardware controllers). This allows the computation of one optical-flow measurement per clock cycle. The number of substages for each coarse-pipeline stage is indicated in brackets.

intrinsic parallelism of the model. Note that the level of parallelism is only schematically represented at each stage. There are some internal operations computed in parallel at each scalar unit to get the final throughput of one estimation per clock cycle (therefore some of these datapaths can be seen as being superscalar units). We used fixed-point representation for all the stages but S_4, which uses floating-point representation. This was a critical decision motivated by the large incoming bit-width at this stage, which makes fixed-point representation very expensive in terms of computational resources. This topic is amply discussed in the next section.

The floating-point stage has been designed to execute each instruction in one cycle. The division operation has been carefully pipelined with a total number of 14 pipelined stages (of the 25 of the S_4 floating-point unit). This is the limiting clock-frequency stage. Although it can be further pipelined to increase the clock frequency, it is unnecessary because the system already fulfills the high-frame-rate specification.

Fig. 4 also shows scalar units with memory access, corresponding to the temporal filtering stages (smooth and derivation filters). The limited number of memory banks accessible on board puts constraints upon the parallelism available within the system, which translates into performance degradation and increases design complexity. Therefore an efficient memory management unit (MMU) is essential to abstract the sequential access to external memory resources. We have designed an efficient MMU, which makes it feasible to describe systems at a high abstraction level. The specific-purpose MMU allows the easy management of the large number of read-write processes necessary for FIR temporal filters with a minimum FPGA resource consumption.

It is worth mentioning that latency of the system is high as far as the number of clock cycles is concerned but negligible in terms of time for image-processing applications. The 70 processing stages require 70 cycles to produce output data. According to the discussion in Section 3.1, each 2D convolution operation requires fetch data to the internal FPGA memory before starting the processing stages. This means that each section, S_1, S_2 and S_3, needs to read a number of pixels approximately equal to four times the horizontal image resolution, producing a final latency of 12 image-column clock cycles. For instance, working at 80 MHz with an image resolution of 800 columns, the total latency is less than 120 μs (including data fetching and pipeline filling). This is completely negligible for most of the image-processing applications and so there is no performance degradation due to large pipelined architecture.

3.3. Bit-width optimization

A proper choice of data structuring is essential to the successful implementation of a customized system. First we need to consider the trade-off between accuracy and resource consumption. Greatest accuracy is obtained with double floating-point representation, which is the choice most often adopted in software approaches. This choice leads to maximum precision at the cost of low performance. Custom processors for real-time systems lack large resources compared to current general purpose processors, so customized datapaths need to be carefully designed to achieve either a comparable or higher performance. Customized systems normally use fixed-point data representation because it fits better onto current digital technology but this strategy requires careful analysis to avoid any degradation in accuracy. Second, good bit-width design is very important for power consumption [35]: low significant bits tend to switch their state more frequently and this should be avoided if they do not drive any information. Therefore the elimination of low significant data allows us to decrease the transition of bit values, thus reducing the switching power, which is important for migration to VLSI devices.

In our design process, we have incorporated several measures and strategies to evaluate how well the specifications are fulfilled. First, conscious of the fact that our goal is to optimize optical-flow accuracy, we have used a simulator of our circuits with several bit-width configurations and the well-known synthetic sequence of a flight through the Yosemite valley [1] as a reference to measure the degradations involved in each approach compared to a double-precision data representation. The decision as to what bit-widths should be used at each stage has a considerable impact on the accuracy of the system. Fig. 5 shows the effects of an insufficient use of bit-width at stage S_4, which leads to significant quantization noise (visually similar to salt-and-pepper noise).

Stages S_0 to S_3 are implemented using fixed-point arithmetic because these operations are based on convolutions. After an extensive analysis of the accuracy vs. resource-consumption trade-off we decided to implement stages S_0 to S_2 with 9 bits and stage S_3 with 18 bits.

This corresponds to using one fractional bit in the image derivatives. With these choices and using a threshold that allows an optical-flow density of 36.47%, the angular error and its changes in variance from 3.38° (8.93° variance) for double-floating-point to 3.43° (8.96° variance) for fixed-point representation (Fig. 6). This choice of bit-width is also motivated by hardware structure (some of the internal hardware resources, such as embedded multipliers and memories, are optimally used for a maximum of 18 bits) and to the fact that the increase in absolute error remains very low (0.05°). Note that along the datapath the accuracy of previous stages limits the maximum precision achievable during the following stages and thus their requirements must be studied carefully in the light of the whole datapath. According to this, increasing the bit-width of these stages only makes sense if the bit-widths of the next stages also increase in order to obtain higher accuracy.

At stage S_4 the incoming bit-width is too large and specific arithmetical representation is required. With these precision

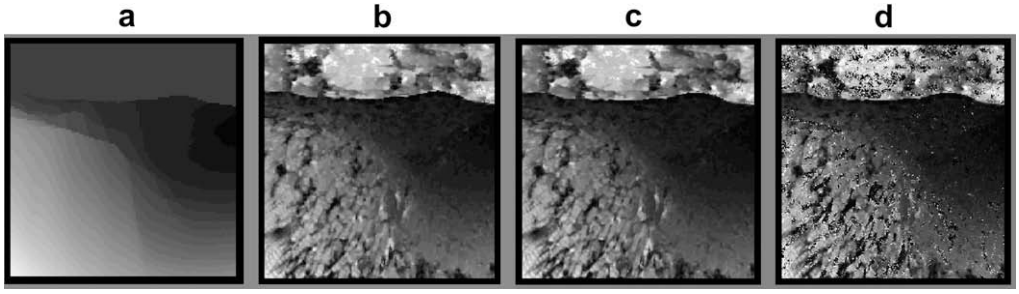

Fig. 5. Magnitude of the velocity for the Yosemite valley fly-through sequence (thresholds not used). From left to right: (a) velocity real ground-truth module; (b) module computed with the software program; (c) module computed with the designed system; (d) module computed with inadequate bit-width for stage S_4 (floating-point mantissa of 6 bits). Note that the quantization errors are visible as a salt-and-pepper-like noise but there is no significant difference between the module of the software (b) and our hardware implementation (c).

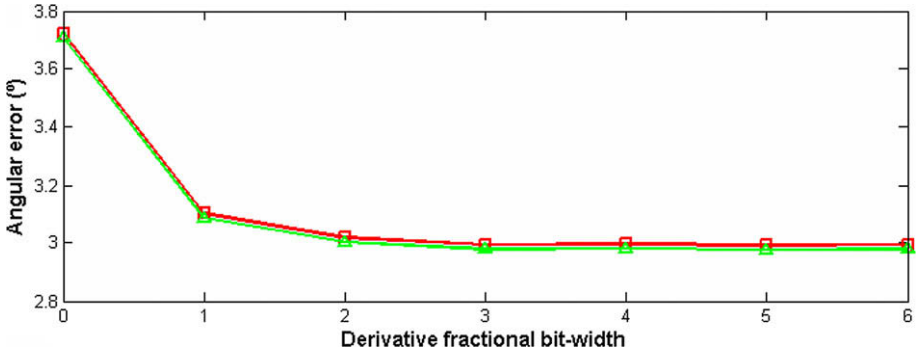

Fig. 6. Angular error considering image derivatives of different bit-widths. We take an integer part of 8 bits plus different values of the fractional part. The following circuit's stages are designed according to this decision. The dark line corresponds to a mantissa of 6 bits used at stage S_4. Light one corresponds to a 12-bit implementation. There is no appreciable improvement for larger mantissa bit-widths. It is clear that the fixed-point stages dramatically compromise the accuracy of the whole system compared to a bit-width decision at the mantissa in stage S_4, i.e., stage S_4 cannot take advantage of larger bit-widths if precision is compromised in previous stages.

requirements, customized floating-point arithmetic produces faster circuits with similar system resources. In Table 1 an implementation using 36-bit fixed-point arithmetic achieves similar accuracy to the one using a customized representation with 19 bits with similar resource consumption but the maximum clock frequency is more constrained (by a factor of 0.54). Thus our study leads us to the conclusion that floating-point arithmetic should be applied in S_4.

To decide what bit-width to use during stage S_4 we carried out an extensive batch of simulations of different cases. The main parameters being studied were the angular error, its variance and the SQNR of the signal. These parameters are represented against the mantissa bit-width of S_4 in Fig. 7.

Once we had decided on the bit-widths to be used at the various stages we went on to evaluate the degradation due to quantization errors in our design. To this end we checked the

Table 1
System accuracy vs. resource consumption trade-off based on stage S_4 precision and data format

Pipelined stages	NAND gates	FFs	Max clock frequency (MHz)	Angular error (°)	Error variance (°)
S_4 Floating-point unit (11 man + 7 exp)	91470	4670	57	3.42	8.95
S_4 Fixed-point unit (36 bits)	90626	2603	31	3.37	8.93

Hardware resources in terms of gate consumption taken from the DK synthesizer [36]. Max clock frequency from ISE Foundation software. Optical-flow error measured using the Fleet & Jepson method [37] at a density of 36.44%. (*man* stands for mantissa and *exp* for exponent).

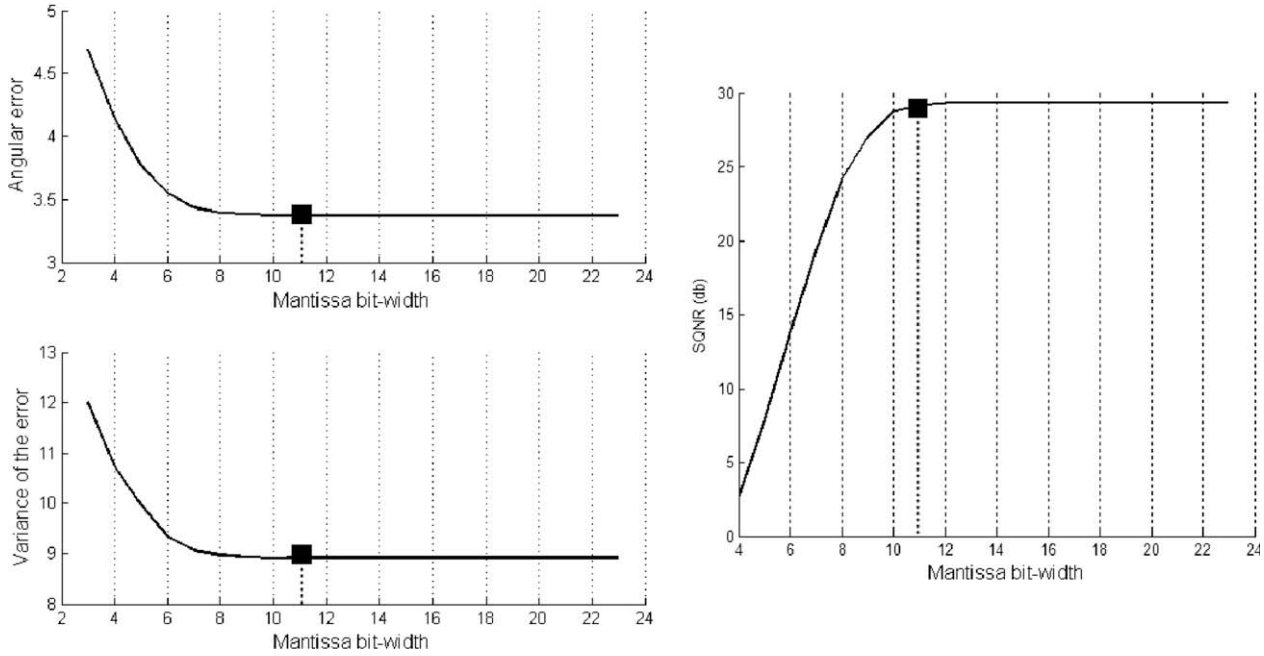

Fig. 7. Angular error (top-left), variance (bottom-left) and SQNR (right) against the number of bits of the mantissa at stage S_4. NB error and variance are approximately constant for values larger than 10 bits and SQNR also becomes stable for values higher than 12 bits. As a satisfactory trade-off we chose a mantissa of 11 bits, which leads to greater accuracy, similar to double floating-point arithmetic with a high SQNR.

accuracy of different implementations, measuring the results in the same pixels (and thus the same densities) (Table 2). As can be seen in the table, the system still achieves very high accuracy. This validates our previous data analysis and confirms that the high bit-width used on the standard computer can be reduced significantly.

A comparative analysis of the optical-flow reliability areas vs. the high quantization error areas highlights the fact that the scenarios with lower confidence are those that produce noisier results due to the quantization effects (Fig. 8). Fig. 8a illustrates confidence values at each image pixel. Bright-grey levels indicate higher confidences. In the right-hand image (Fig. 8b), where light areas represent pixels with higher quantization noise, we show the difference in angular error between software and hardware estimations. This image clearly demonstrates that the areas prone to higher quantization noise are the ones with lower optical-flow confidences. Therefore, the confidence filtering efficiently helps to ignore mistakes caused by quantization errors.

This effect can be easily understood in the light of image structure. Image pixels with low confidence correspond to areas of low structure, which produce smaller image derivatives, and therefore the division operations required to estimate motion are prone to errors.

When these results are computed using a limited bit-width the relative importance of their quantization is larger and translates into erroneous estimations. This requires the use of a proper threshold for the optical-flow computation, allowing us to get high accuracy circuits with lower resource consumption and rejecting unreliable estimations (i.e. those prone to quantization errors).

Table 2
Comparing angular errors between the software double-data type and the customized data structure used in the hardware implementation

Density (%)	Software double-data implementation		Customized hardware implementation	
	Angular error (°)	Variance (°)	Angular error (°)	Variance (°)
36.47	3.4330	9.1056	3.5166	9.2406
42.14	4.0731	9.8389	4.2128	10.0986
57.20	6.9166	12.9283	7.8611	14.5013
91.95	17.3303	20.3685	18.3185	20.6841

Note that although the bit-width has been dramatically reduced, due to the previous analysis the results obtained with the hardware approach are only slightly degraded even at significant estimation densities.

3.4. System resources and performance

The whole system benefits from the accuracy analysis of the previous section and has been successfully implemented on several prototyping boards for embedded image-processing applications (RC300 from Celoxica [36] and Xirca-V2 from Seven Solutions [38]). The prototyping boards are provided with a Virtex II XC2V6000-4 Xilinx FPGA as processing element and also video input/output circuits and user interfaces/communication buses. Table 3 shows the hardware costs of the motion-processing core as well as the control/interface logic. After implementation, the whole design (including user interface, camera, memory and VGA) consumes 27% of the FPGA slices, 27% of the available embedded memory and 8% of the internal multipliers. Its maximum frequency clock rate is 82 MHz and therefore the maximum experimental processing performance is 82 *mega pixels per second* (Mpps). The image resolution can be selected according to image input camera standard or processing capabilities. This architecture is modular and scalable, making it possible to reduce the system's parallelism (and performance) to fit onto smaller devices. Furthermore, the processing core can be replicated (more than 73% of the system's resources are available) and. thanks to the MMU architecture, the frame-grabber can be easily modified to split the image and send it to several processing units. This high-level scalability allows the processing performance to be multiplied if required.

It should be pointed out that the processing performance set out in Table 3 represents the performance of the processing core itself. The design strategy based on a highly parallel architecture allows the optical-flow core to compute one pixel per clock cycle. Of course, for the complete system, including memory, camera interface and command port, the performance may be constrained by these elements rather than by the computing core. For instance, if all the modules work in only one clock domain, we need to use three independent memory banks of 36 bits data words each to avoid memory access conflicts. We use two for the temporal filter stage and one for reading/writing results. If the incoming camera data stream is slower than our processing circuit, this module will determine the maximum performance. In our experiments we have used a couple of FPGA boards for prototyping. The RC300 platform from Celoxica [36] has two analog composite video inputs. Working at 20 MHz we have been able to process video NTSC format from two cameras at 28 fps. We have also tested the system using a coprocessing PCI board from Seven Solutions [38]. Working at 66 MHz clock rate we have been able to compute 1 megapixel resolution cameras at 16 fps. In both cases the limiting factors were the camera input/PCI interface and not the processing core itself.

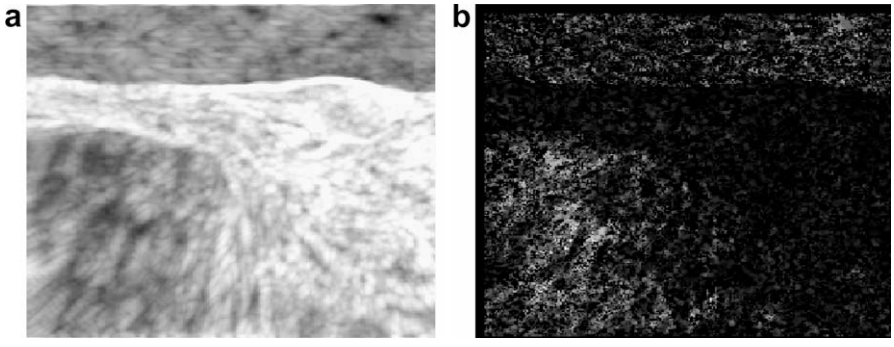

Fig. 8. Confidence areas and quantization error: (a) logarithm of optical-flow confidence values (light-grey indicates high confidence estimations); (b) software-hardware angular error difference. Data range (logarithmic) scaling is done to improve visualization. Note that areas prone to higher quantization noise correspond to the lower optical-flow confidence areas.

Table 3
Resource consumption of the different stages (results taken from the DK synthesizer [36]) and maximum clock-frequency values (results from the Xilinx ISE Foundation)

Pipelined stages	NAND Kgates	FFs	Memory bits	Max clock frequency (MHz)	Maximum frame rate
Interfaces + hardware controllers	83.4	2148	23584	99	170
Motion-processing core	1641.9	8022	737280	57	
Complete system	1731.9	10433	760864	82	

First row: interfaces and hardware controllers (camera frame-grabber, MMUs, VGA signal output and user configuration interface). Second row: motion-processing core. Third row, complete system resources required. The last column shows the complete system maximum performance (frame rate) at a resolution of 800 × 600 pixels. The results presented allow a maximum horizontal image resolution of 1024 pixels. If greater resolution is required, then the use of memory resources will be doubled (allowing a maximum horizontal resolution of 2048 pixels) with a very small extra logic overhead.

Finally, it is important to indicate why the optical-flow core has a lower maximum frequency clock than that of the whole system, as is shown in Table 3. The maximum clock rate estimations for the different modules are produced by the vendor FPGA tool (ISE Foundation of Xilinx). We have used advanced synthesis and place & route capabilities such as *retiming* with the result that under several circumstances complex circuits are able to run faster than their different sub-modules. Since our design uses a long pipeline, the tool has a significant chance to move registers for path delay balancing (*retiming*). Therefore, the performance of the optical-flow core is improved when it is placed and routed together with more logic, which explains the differences that appear in Table 3.

3.5. Performance comparison with other approaches

The implementation of the optical-flow algorithm with FPGAs has only been addressed by some authors in very recent years. Table 4 summarizes the performances obtained by the different approaches. In previous works [15,16] we proposed the basic implementation of the L&K model and presented a detailed study into its performance vs resources trade-off. These papers also cover the topic of system degradation related to bit-widths at different processing stages. Although the performance was correct, neither the image resolution nor frame rates measured up to the high-frame-rate requirements addressed here. The iterative algorithm of Horn & Schunk (H&S) [39] has also been implemented by different authors. Martin et al. [40] described a system implementation that conforms quite well to the specifications of a standard-frame-rate optical-flow system. The main disadvantage of their approach is that the accuracy of the model itself does not compare well with the L&K model, as shown by Barron et al. [1]. Cobos et al. [41] have also described the implementation of the H&S model but using modest resources and therefore achieving lower performance.

Using the block-matching approach, the implementation described by Niitsuma & Maruyama [42] achieves 30 fps with an image size of 640 × 480 but with high hardware cost (90% slices of a XC2V6000 FPGA) and without subpixel accuracy. Strictly speaking, our approach is only 5 times faster than [42]. Nevertheless, using the same target device, our system requires only 24% of the resources, which allows us to increase our performance by means of core replication (up to 4 cores fit in this device and therefore we could still scale up the system's performance). Furthermore their approach [42] does not provide subpixel accuracy.

Based on the L&K approach, Correia & Campilho [43] described a real-time implementation of the system using a MaxVideo200 pipeline image processor. Though still far from our results, they achieved high-performance (1666 Kpps) because they took full advantage of the pipeline architecture. Nevertheless, the use of an acceleration processor (such as the MaxVideo200) makes it difficult to transfer the system to embedded platforms.

For standard PC platforms, the approach described in [29] achieves high accuracy but with a performance still far below our results. Moreover, the method used in [29] is not hardware.

Finally, the model described here, running on software on an Intel Pentium 4 HT, 3200 MHz, can compute 47.6 fps of 160 × 120 pixels (914 Kpps, as indicated in the last row of Table 4), though this can be further optimized by using MMX and SSE instructions. But in any case this requires the full computing power of the machine.

It is important to remark that the differences are not only in terms of frame rates; we should also consider the range of speeds supported by different implementations. Systems based on gradient models, such as like H&S or L&K, have similar ranges of speed. On the other hand, the method described on [42] is based on block-matching and has a maximum speed range of ±13.5 pixels. The system described in [29] uses a global energy minimization technique and therefore the speed range is not explicitly limited by local filters. The speed range of the approach presented here depends on the input image stimulus as shown in Appendix B but, taking advantage of a high-frame-rate camera, this is not a limitation for our system.

Since the cited works are very recent (some of then using even the same evaluation devices), the outstanding performance of our approach is not due to improvements in technology but rather to a very efficient processing architecture (superpipelined and superscalar datapath) that has extensive recourse to the parallel resources of the FPGA device.

Table 4
Comparison with previous works

Implementation	Throughput (Kps)	Image size (pixels)	Image rate (frames/s)
Improved L&K (described here)	82000	800 × 600	170
L&K (stand-alone board, [15])	4100	320 × 240	53
L&K (PCI-board [16])	1776	320 × 240	24
H&S (Martin et al. [40])	3932	256 × 256	60
Block-matching,(Niitsuma & Maruyama [42])	9216	640 × 480	30
L&K (Correia and Campilho [43])	1666	256 × 256	25
H&S (Cobos et al. [41])	47.5	50 × 50	19
Variational (Bruhn et al. [29])Intel Pentium4, 3 GHz	1444.7	316 × 252	18
Intel Pentium 4 HT (3.2 GHz)	914	160 × 120	47.6

Our data uses the maximum available clock frequency for comparison with previous approaches.

4. Conclusions

The need for high-frame-rate optical-flow systems has been clearly demonstrated. Current image sensors make very fast image acquisition possible and this leads in turn to significant improvements in optical-flow accuracy [5]. Simple gradient-based optical-flow approaches seem to be a suitable alternative for moderate cost systems (compared with complex multi-scale approaches). Accordingly, we have implemented an improved version of the L&K model [20] that complements the capabilities of high-frame-rate cameras, providing highly accurate real-time image-motion analysis.

We have designed a scalable, modular, versatile architecture and have described how parallel and superpipelined structures can be used in the implementation of algorithms to create novel high-performance architectures for image processing. Our system outperforms by more than one order of magnitude any previous approach, thus validating the computing scheme adopted. From our accuracy analyses we conclude that with a much reduced bit-with data representation the system achieves accuracy close to the software implementation (with double-precision floating-point representation).

One of the important contributions of this paper is the novel digital design technique based on long pipelines, which go beyond an optimized hardware implementation. This technique is significantly different from standard processors or DSP architectures, in which the pipeline paths are be kept short to reduce backing-up pipeline operations due to cache miss. This paper represents an illustrative example of this design technique, rather than a specific processing engine design for motion estimation.

In addition, fine pipeline architecture design is a useful procedure for developing complex systems that result in other benefits apart from high performance. As shown in [44,45], the power consumption of FPGA devices is lower when deep pipelines are used and this can be easily explained, circuit glitches significantly contribute to the power consumption in digital devices. Intense data registering reduces the combinational logic path length and therefore reduces the propagation of the glitches. Thus, fine circuit pipelining leads to a saving of up to 70% of the power consumption with this kind of device, just by reducing glitch propagation.

Finally, we have evaluated the system's resources consumption and the performance of an implementation on a stand-alone platform, which fulfills the high-frame-rate optical-flow requirements. A comparison with publications by other authors clearly shows the improved performance of our new system and opens the door to a wide range of fields of application.

Future work will cover the use of such systems for real-world applications involving moving robotic platforms, such as robot navigation and tracking [46], and structure extraction from motion analysis. On-chip integration schemes of multiple vision cues, such as stereo information and colour [47], are currently being explored.

Acknowledgments

This work was supported by EU Research Framework VI funding via the European Project DRIVSCO (IST-016276-2), the National Spanish Grant DEPROVI (DPI2004-07032) and the Junta de Andalucía Project: P06-TIC-02007. We thank A.L. Tate for revising our English text.

Appendix A. The L&K model and adopted improvements

In the following equations, we describe briefly the computations upon which the L&K approach is based and the modification of the system parameters according to [20]. The L&K algorithm belongs to gradient-based techniques. Upon the assumption of constant luminance values through time, the first-order gradient constraint Eq. (A.1) is obtained as:

$$\nabla_{xy} I(x,y,t)\cdot(V_x,V_y)+I_t(x,y,t)=0 \tag{A.1}$$

This equation only allows us to estimate velocity in the direction of maximum gradient, i.e. in the normal direction of moving surfaces. To overcome this limitation the L&K method constructs a flow estimation based on the first-order derivatives of the image. By least-square fitting, the model extracts an estimation of motion based on the hypothesis of similarity of velocity values in the neighbourhood of a central pixel. This is described mathematically by the minimization of Eq. (A.2).

$$\min \sum_{x\in\Omega} W^2(x)[\nabla_{xy} I(x,y,t)\cdot(v_x,v_y)+I_t(x,y,t)]^2 \tag{A.2}$$

where $W(x)$ weighs the constraints of the spatial neighbourhood Ω (with larger values near the centre).

The known solution to this problem is expressed in Eq. (A.3),

$$\vec{v}=[A^T W^2 A]^{-1} A^T W^2 \vec{b} \tag{A.3}$$

where

$$A^T W^2 A=\begin{bmatrix} \sum_{x\in\Omega} W^2 I_x^2 & \sum_{x\in\Omega} W^2 I_x I_y \\ \sum_{x\in\Omega} W^2 I_x I_y & \sum_{x\in\Omega} W^2 I_y^2 y \end{bmatrix} \tag{A.4}$$

$$A^T W^2 \vec{b}=\begin{bmatrix} -\sum_{x\in\Omega} W^2 I_x I_t \\ -\sum_{x\in\Omega} W^2 I_y I_t \end{bmatrix} \tag{A.5}$$

The main parameters to modify with respect to the most widely used implementation [1] are described in [20]. We encourage the reading of this work for a fuller understanding of these modifications. To summarize, the changes are the following:

1. The pre-filter can be reduced from $11\times11\times11$ taps (Gaussian variance 1.5) to a separable kernel of $3\times3\times3$, P = [1,2,1]/4 attached to the rest of the modifications described below. Without any pre-filters the density of the system increases dramatically but undergoes a deterioration in accuracy, which means that the pre-filter rejects useful information as well as noise. A good trade-off solution consists of using these 3 taps smoothing kernel P, which also contributes as an anti-aliasing filter.
2. The derivative kernels used in the standard implementation of L&K [1] ([−1 8 0 −8 1]/12) have a systematic bias towards 1 pixel/frame in the optical-flow, thus degrading its accuracy. Brandt [20] shows that the adequate use of smoothing and derivative kernels, as proposed also by Simoncelli [31], significantly improves the accuracy of the system. Such a task requires the use of 3D complementary derivative kernels, which work as smoothing kernels in 2 dimensions and a derivative operator on the other axis. In terms of performance, these kernels represent n factor 3increase in computation load during this stage, but this only represents slightly higher resource consumption when designing customized hardware because it can be implemented into the pipeline structure without throughput degradation.
3. Neighbourhood area Ω. In their implementation Barron et al. [1] use a 5×5 spatial central-weighting neighbourhood with a separable kernel [0.0625,0.25,0.375,0.25,0.0625]. Brandt demonstrates in [20] that a uniformly weighted kernel of 5x5 increases the density with only a small degradation in accuracy. Another option is to use a 3D smoothing neighborhood with a small separable kernel [121]/4. This kernel optimizes the space/time resolution of the estimator symmetrically and, depending on the motion flow field, also improves accuracy.

We adopt the first two modifications of the original L&K implementation, which improve motion accuracy as well as flow density, as can be seen in Table A.1. Furthermore, for the spatial integration of constraints we have used the spatial kernel [0.0625,0.25, 0.375,0.25,0.0625] to enhance accuracy. The overall support of the algorithm is reduced from $19\times19\times15$ pixels (in the implementation of Barron et al. [1]) to $11\times11\times7$ pixels in Brandt's [20] modifications; thus just 7-image storage capacity is required. In a previous implementation of the L&K model [15,16] we used the IIR temporal filter described by Fleet et al. [32], which requires only 3-image storage capacity.

Table A.1
Accuracy evaluation of different implementations of the L&K model using the Yosemite fly-through synthetic sequence with known ground-truth

Motion estimation approach	Average error (°)	Standard deviation (°)	Density (%)
Standard L&K implementation (from [1])	4.28	11.41	35.1
Standard L&K implementation (using Eigen-values product as confidence parameter)	4.57	12.77	36.44
L&K using IIR temporal filters such as those described in [32]	6.4716	13.0057	36.46
Improved L&K (based on [20])	3.3757	8.9263	36.44

The error measure is described in Fleet et al. [37]. Note that the confidence thresholds are slightly different for each version to arrive at similar optical-flow densities. The threshold parameter used is the product of Eigen-values because it is hardware friendly and gives a good error discrimination sensibility [20].

The drawback of this approach is that IIR filters are not so accurate for the estimation of optical-flow and need a higher fixed-point bit width to compute filter values. The 7-image storage requirement of the modified L&K model is less than half that of Barron's approach [1] and feasible on embedded systems.

All the accuracy tests commented upon above are conditioned by the reliability test parameter used. The equations to estimate pixel velocities are often under-determined or ill-conditioned because the gradient direction does not vary sufficiently within the integration neighbourhood. Several measures can be used to detect these situations [18]. According to the results provided by Brandt [20], we use the product of the Eigen-values of the matrix of Eq. (A.4) since it provides similar results to those obtained when using the well-known minimum Eigen-value criteria from Barron et al. [1] and is more hardware-friendly because it corresponds to the determinant of Eq. (A.4), which is already computed in the motion estimation datapath.

Appendix B. First-order Gradient model limitation analysis

This section reviews 3D spatio-temporal sampling theory and investigates the effects of motion aliasing (this being, as a first approximation, the main limitation of the L&K model). It is discussed in a simplified but insightful way.

The L&K model is based on a first-order Taylor expansion of the image [Eq. (B.1)], which is correct only if quadratic and further terms can be neglected. This is true for small velocity vectors but errors grow fast when high-order terms become significant. Nevertheless, the consideration of how large or small a velocity might be depends on the structure of the image presented in the neighbourhood of each pixel. According to the Nyquist-Shannon theorem, the maximum velocity that can be measured in an image without aliasing is limited by the local spatial bandwidth. According to Weber et al. [4], if we consider a sinusoid grating of wavelength λ, we can limit the maximum acceptable displacement given by Eq. (B.1.a):

$$V_{Max-theoretical} < \lambda/2 \tag{B.1.a}$$

$$V_{Max-experimental} < \lambda/2\pi \tag{B.1.b}$$

For real images, they show an even smaller velocity limit, about one third of the theoretical limit [Eq. (B.1.b)] [4]. This equation implies that maximum velocity is strongly correlated with the spatial frequencies presented at each image position. The maximum frequency in units of pixels is 0.5 pixels-1 which means that $\lambda = 2$ pixels and thus the maximum theoretical speed that can be recovered is less than 1 pixel per frame at the highest spatial frequency. Therefore, using pixels as units, the sampling period is 1 pixel and we cannot recover 1 pixel motion of $\lambda = 2$ sinusoidal gratings. This also means, however, that the maximum value of the velocity can be very high for images with spectral contents of long wavelengths. For example, if we consider $\lambda = 100$ pixels, the algorithms could theoretically recover motion up to 49 pixels/frame (first integer value below 100/2) and experimentally 16 pixels/frame. But note that in order to get this estimation we need to tune the image derivative to the proper frequency in order to get a response from the filters.

The next point concerns the pre-filters and derivative kernel sizes. The derivative operation is usually computed as a convolution with Gaussian derivatives, which works as band-pass filters with an optimum frequency response given by Eq. (B.2), where σ^2 represents the variance and n the derivative order [48].

$$\omega_n = \sqrt{n/\sigma^2} \tag{B.2}$$

The use of large filters allows us to recover fast motion because it corresponds to long wavelengths but high-frequency image information is lost. Furthermore, the Gaussian derivative bandwidths are approximately constant and asymptotically equal, as expressed in (B.3) [49]. This means that the spatial extension of the Gaussian derivative filters is inversely proportional to the filter bandwidth.

$$\Delta\omega \to \frac{1}{\sqrt{2\sigma}} \tag{B.3}$$

According to Eq. (B.3), smaller kernels allow us higher flow densities because a larger frequency range is considered (although filters are not optimally tuned for the whole range). Nevertheless, the drawback is that these small-spatial-resolution filters provide estimations prone to noise, which typically affect high spatial frequencies more significantly. Thus, for low noise sequences, a small smoothing kernel may be profitable for real images but the final decision as to which pre-filters and derivative kernels are best must take into account the SNR of the input images and their spectral properties.

Another important hypothesis is the implicit assumption of constant luminance. Large temporal filters impose a high restriction on the illumination condition, which is not always preserved in real scenarios. This might advise the use of shorter temporal windows for computing the optical-flow. The drawback of this approach, however, is that higher temporal frequencies are available, thus increasing the aliasing artefacts.

We decided to use 5-pixel-long, first-order Gaussian-derivative kernels, which is a strategy that is widely used in most implementations and evaluations because it results in a good trade-off between accuracy and computing resources. This strategy implies two basic assumptions:

1. Low noise. Standard micro-cameras achieve SNR > 45 dB in standard environments (i.e. not industrial) with homogenous illumination.
2. Only low speeds can be computed, at least for high spatial frequencies. This is a more restrictive assumption. First-order Gaussian-derivative kernels of 5 pixels have a variance of 1 pixel and a top cut-off frequency of 1.35 rad/pixels [using Eqs. (B.2) and (B.3), $\omega 0 + \Delta\omega/2$] which corresponds to a wavelength of $\lambda = 1.48\pi$ pixels and gives us an experimental maximum speed for this frequency of 0.74 pixels/frame. This highly recommends the use of high-frame-rate cameras for motion estimation in real scenarios.

References

[1] J.L. Barron, D.J. Fleet, S. Beauchemin, Performance of optical-flow techniques, International Journal of Computer Vision 12 (1) (1994) 43–77.
[2] B. McCane, K. Novins, D. Crannitch, B. Galvin, On Benchmarking optical flow, Computer Vision and Image Understanding 84 (2001) 126–143.
[3] H.C. Liu, T.S. Hong, M. Herman, T. Camus, R. Chellappa, Accuracy vs efficiency trade-offs in optical flow algorithms, Computer Vision and Image Understanding 3 (1998) 271–286.
[4] J. Weber, J. Malik, Robust computation of optical flow in a multi-scale differential framework, International Journal of Computer Vision 14 (1995) 67–81.
[5] S. Lim, J.G. Apostolopoulos, A.E. Gamal, Optical flow estimation using temporally over-sampled video, IEEE Transactions on Image Processing 14 (8) (2005) 1074–1087.
[6] D.J. Fleet, A.D. Jepson, Computation of component image velocity from local phase information, International Journal of Computer Vision 5 (1) (1990) 77–104.
[7] E.P. Simoncelli, E.H. Adelson, D.J. Heeger, Probability distributions of optical flow, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, Mauii, Hawaii, 1991, pp. 310–315.
[8] T. Gautama, M.M. van Hulle, A phase-based approach to the estimation of the optical flow field using spatial filtering, IEEE Transaction on Neural Networks 13 (5) (2002) 1127–1136.
[9] P.R. Giaccone, G.A. Jones, Feed forward estimation of optical flow, in: IEE Conf. on Image Processing and its Applications, Dublin, 1997, pp. 204–208.
[10] L. Alvarez, J. Weickert, J. Sánchez, Reliable estimation of dense optical flow fields with large displacements, International Journal of Computer Vision 39 (1) (2000) 41–56.
[11] Dalsa company, Web site and products datasheets available at: http://mv.dalsa.com/.
[12] Hitachi Company, Web site and products datasheets available at: http://www.hitachi-service.net/.
[13] IMS-Chips Company, Web site and products datasheets available at: http://www.ims-chips.com/index.php3.
[14] B.D. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, in: Proc. of the DARPA Image Understanding Workshop, 1984, pp. 121–130.
[15] J. Díaz, E. Ros, S. Mota, F. Pelayo, E.M. Ortigosa, Sub-pixel motion computing architecture, IEEE Proceedings on Vision, Image & Signal Processing 153 (6) (2006) 869–880.
[16] J. Díaz, E. Ros, F. Pelayo, E.M. Ortigosa, S. Mota, FPGA based real-time optical-flow system, IEEE Transactions on Circuits and Systems for Video Technology 16 (2) (2006) 274–279.
[17] A. Bainbridge-Smith, R.G. Lane, Determining Optical Flow Using a Differential Method, Image and Vision Computing 15 (1) (1997) 11–22.
[18] A. Bainbridge-Smith, R.G. Lane, Measuring confidence in optical flow estimation, IEE Electronic Letters 10 (1996) 882–884.
[19] S. Maya-Rueda, M. Arias-Estrada, FPGA processor for real-time optical flow computation, Lecture Notes in Computer Science 2778 (2003) 1016–1103.
[20] J.W. Brandt, Improved accuracy in gradient based optical flow estimation, International Journal of Computer Vision 25 (1) (1997) 5–22.
[21] S.P. Sabatini, F. Solari, G.M. Bisio, Spatiotemporal neuromorphic operators for the detection of motion-in-depth, in Proc. 2nd ICSC Symposium on Neural Computation, 23–26 May 2000, Berlin, Germany, pp. 874–880.
[22] P.C. Merrell, D. Lee, Structure from motion using optical flow probability distributions, in: Kevin L. Priddy, (Ed.), Intelligent Computing: Theory and Applications III, Proceedings of the SPIE, vol. 5803, 2005, pp. 39–48.
[23] K. Pauwels, M.M. Van Hulle, Segmenting Independently Moving Objects from Egomotion Flow Fields. Early Cognitive Vision Workshop, Isle of Skye, May 28–June 1, 2004.
[24] T. Brox, B. Rosenhahn, D. Cremers, H. Seidel, High accuracy optical flow serves 3-D pose tracking: exploiting contour and flow based constraints, ECCV (2) (2006) 98–111.
[25] PACO-PLUS EU project. Web site at: http://www.paco-plus.org/.
[26] T. Camus, Real-time quantized optical flow, Real-Time Imaging 3 (2) (1997) 71–86. April.
[27] J. Díaz, E. Ros, A. Rotter, M. Muehlenberg, Lane-change decision aid system based on motion-driven vehicle tracking, IEEE Transactions on Vehicular Technology, accepted for publication.
[28] E. Ros, J. Díaz, S. Mota, F. Vargas-Martín, M.D. Peláez-Coca, Real time image processing on a portable aid device for low vision patients. LNCS, 2006, pp. 158–163.
[29] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, C. Schnorr, Variational optical flow computation in real time, IEEE Transaction on Image Processing 14 (5) (2005) 608–615.
[30] A. Amir, L. Zimet, A. Sangiovanni-Vincentelli, S. Kao, An embedded system for an eye-detection sensor, Computer Vision and Image Understanding 98 (1) (2005) 104–123.
[31] E.P. Simoncelli, Design of multi-dimensional derivative filters, in: Proc. IEEE International Conf. on Image Processing, Austin, TX, 1994, pp. 790–794.
[32] D.J. Fleet, K. Langley, Recursive filters for optical flow, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (1) (1995) 61–67.
[33] M.J. Forsell, Architectural differences of efficient sequential and parallel computers, Journal of Systems Architecture: the EUROMICRO Journal 47 (13) (2002) 1017–1041.
[34] J. Hannuksela, P. Sangi, J. Heikkilä, Vision-based motion estimation for interaction with mobile devices, Computer Vision and Image Understanding 108 (1-2) (2007) 188–195.
[35] G. Constantinides, Perturbation analysis for word-length optimization, in: Proc. IEEE Sym. on Field-Programmable Custom Computing Machines (FCCM'03), Los Alamitos, California, September 2003, pp. 81–90.
[36] Agility Company (previously named Celoxica), Web site and products datasheets available at: http://www.agilityds.com/.
[37] D.J. Fleet, Measurement of Image Velocity, Engineering and Computer Science, Kluwer Academic Publishers, Norwell, MA, 1992.
[38] Seven Solutions S.L., Web site and products available at: http://www.sevensols.com.
[39] B.K.P. Horn, B.G. Schunck, Determining optical flow, Artificial Intelligence 17 (1981) 185–203.
[40] J.L. Martín, A. Zuloaga, C. Cuadrado, J. Lázaro, U. Bidarte, Hardware implementation of optical flow constraint equation using FPGAs, Computer Vision and Image Understanding 3 (2005) 462–490.
[41] P. Cobos, F. Monasterio. FPGA implementation of the Horn & Shunk optical flow algorithm for motion detection in real time images, in: Proc. of the XIII Design of Circuits and Integrated Systems Conference. Madrid, Spain, November 1998, pp. 616–621.
[42] H. Niitsuma, T. Maruyama, Real-time detection of moving objects, Lecture Notes in Computer Science 3203 (2004) 1153–1157.
[43] M.V. Correia, A.C. Campilho, Real-time implementation of an optical flow algorithm, International Conference on Pattern Recognition (ICPR2002), Quebec City, Canada, 11–15 August 2002, pp. 247–250.
[44] A.A. Shen, A. Ghosh, S. Devadas, K. Keutzer, On average power dissipation and random pattern testability of CMOS combinational logic networks, in Proc. IEEE International Conference on Computer Aided Design, Santa Clara, California, United States, November 1992, pp. 402–407.
[45] G. Sutter, S. López-Buedo, E. Todorovich, E. Boemo, Logic Depth, Power, and Pipeline Granularity: Some Examples on FPGAs, presented at JCRA, Madrid/Spain, 2003, pp. 201–208.
[46] G. Medioni, A.R.J. Francois, M. Siddiqui, K. Kim, H. Yoon, Robust real-time vision for a personal service robot, Computer Vision and Image Understanding 108 (1-2) (2007) 196–203.
[47] R. Munoz-Salinas, E. Aguirre, M. Garcia-Silvente, People detection and tracking using stereo vision and color, Image and Vision Computing 25 (6) (2007) 995–1007.
[48] J.A. Bloom, T. R Reed, A Gaussian derivative-based transform, IEEE Transactions on Image Processing 5 (3) (1996) 551–553.
[49] J.J. Koenderink, A.J. van Doorn, Representation of local geometry in the visual system, Biological Cybernetics 55 (1987) 367–375.