FEMIP: A high performance FPGA-based features extractor & matcher for space applications

FEMIP: A HIGH PERFORMANCE FPGA-BASED FEATURES EXTRACTOR & MATCHER FOR SPACE APPLICATIONS

Stefano Di Carlo, Giulio Gambardella, Paolo Prinetto, Daniele Rolfo, Pascal Trotta
Dipartimento di Automatica e Informatica
Politecnico di Torino
Corso Duca degli Abruzzi 24
I-10129, Torino, Italy
email: {name.familyname}@polito.it

Piegiorgio Lanza

ThalesAlenia Space Italia S.p.a.

Strada Antica di Collegno 253
I-10146, Torino, Italy
email: piergiorgio.lanza@thalesaleniaspace.com

ABSTRACT

Nowadays, Video-Based Navigation (VBN) is increasingly used in space-applications. The future space-missions will include this approach during the Entry, Descent and Landing (EDL) phase, in order to increase the landing point precision. This paper presents FEMIP: a high performance FPGA-based features extractor and matcher tuned for space applications. It outperforms the current state-of-the-art, ensuring a higher throughput and a lower hardware resources usage.

1. INTRODUCTION

VBN is a well-studied area of computer vision, embracing several application fields, including robotics, unmanned vehicles, avionics [9] and, more recently, space-applications. Space-missions are increasingly resorting to VBN systems to assist and enhance the precision of the EDL phase of space modules.

VBN extracts geometrical information from a set of real-time sampled images, performing two distinct computational tasks: *Features Extraction and Matching* (FEM) and *Motion Estimation*. During FEM, interesting *features* (e.g., corners or edges) of each image are extracted; features that match between two consecutive images are then identified. FEM is a very computational-intensive task and it must be performed at a high frame rate to assure high precision. Very efficient hardware accelerators are therefore mandatory to ensure the required performances.

Several feature extraction algorithms have been proposed in the literature (e.g., Beaudet, SUSAN, Harris, SURF) [11]. The Harris corner detector is the best trade-off between precision and complexity. It is therefore a good candidate to be implemented as a hardware accelerator block.

This paper proposes *FEMIP*, a high performance FPGA-based FEM IP-core based on the Harris algorithm and optimized for being used in space EDL systems.

Only few publications proposed FPGA-based implementations of the Harris algorithm. Benedetti et al. [1] presented a very high speed hardware architecture (i.e., 30 fps output frame rate). However, their solution requires the parallel use of four FPGAs and is therefore not suitable for space-applications. Cabani et al. presented in [2] an interesting scale-invariant implementation of Harris. However, area occupation is not well-optimized.

To the best of our knowledge, the state-of-the-art solution has been developed in the framework of the *ESA NPAL Project* [5]. It provides good performances in terms of resources utilization and output frame rate (20 fps). Nonetheless, it requires an external co-processor to perform the matching phase. The present paper overcomes these limitations, proposing a high performance FEM architecture able to process images at up to 33 fps, with very limited hardware resources and without resorting to external co-processors.

The rest of the paper is organized as follows. In Section 2 FEMIP and its internal architecture are described; in Section 3 experimental results are reported and finally, Section 4 summarizes the main contributions and concludes the paper.

2. FEMIP ARCHITECTURE

FEMIP gets a 32-bit input stream representing 1024x1024 grey scale images with 10 bit per pixels (bpp) resolution, as provided by almost all space-qualified CMOS cameras [3]. It provides a set of features that match between two consecutive images. FEMIP internal structure includes three functional blocks: *Gaussian Filter*, *Harris Features Extractor*, and *Features Matcher*.

2.1. Gaussian Filter

The *Gaussian Filter* performs Gaussian smoothing [7] of the input image. It reduces the level of noise in the image, improving the accuracy of the feature extraction algorithm. In

our architecture, Gaussian filtering is performed via a 2D-convolution of the input image with a 7x7 pixels Gaussian kernel mask [7]. A 7x7 kernel is enough to forcefully reduce the noise that strongly affects images taken in space environments. Fig. 1 shows the optimized architecture of the *Gaussian Filter*.

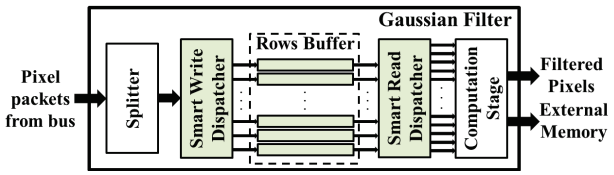
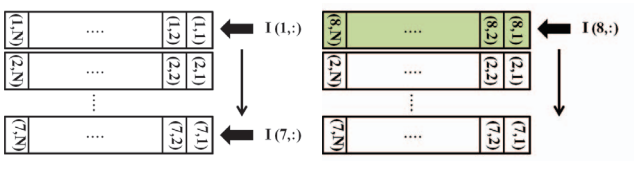


Fig. 1: *Gaussian Filter* internal architecture

The *Splitter* gets the image flow through the 32-bit FEMIP input interface and unpacks it in order to reconstruct the original 10-bit pixel flow. Images are received in a raster format, line-by-line from left to right and from top to bottom, without any control or padding bit. Pixels are then sent to the *Smart Write Dispatcher* (SWD), that stores them inside the *Rows Buffer* (RB) before the actual convolution computation. RB is composed of 7 FPGA Block-RAMs (BRAMs) [12], each one able to store a full image row. The number of rows of the buffer is dictated by the size of the used kernel matrix (i.e., 7x7). Image rows are buffered into RB using a circular policy, as reported in Fig. 2.

The *Smart Reader Dispatcher* (SRD) works in parallel with SWD, retrieving a set of consecutive 7x7 image blocks from RB, following a sliding window approach.

The SRD activity starts when the first 7 rows of the image are loaded in RB. At this stage, pixels of the central row (row number 4) can be processed and filtered. It is worth to remember here that, using a 7x7 kernel matrix, a 3-pixel wide border of the image is not filtered, and related pixels are therefore discarded during filtering. At each clock cycle, a full *RB column* is shifted into a 7x7 pixels *Sliding Window Buffer* (SWB), composed of 49 10-bit registers. After the 7th clock cycle, the first image block is ready for convolution. The *Computation Stage* (CS) convolves it by the *Kernel Mask* and produces an output filtered pixel. At each following clock cycle, a new RB column enters the SWB and a new filtered pixel of the row is produced.



(a) 1st to 7-th row received (b) 8-th image row received

Fig. 2: *Smart Write Dispatcher* operations. (i,j) indicates pixel coordinates.

While this process is carried out, new pixels continue to feed RB through SWD, thus implementing a fully pipelined computation. When a full row has been filtered, the next row can be therefore immediately analyzed. However, according to the circular buffer procedure used to fill RB, the order in which rows are stored changes. Thus in order to provide the pixel in the right order to the *Computation Stage*, the SRD includes a dynamic connection network with the SWB. This network guarantees that, while rows are loaded in RB in different positions, SWB is always fed with an ordered column of pixels (See Fig. 3). The *Computation Stage* performs the

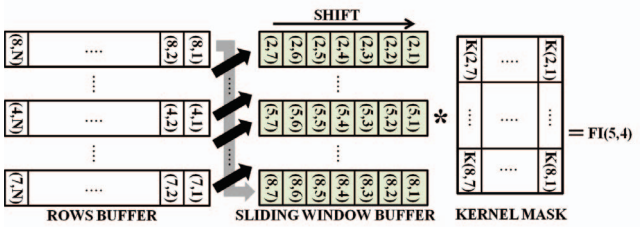


Fig. 3: SRD behavior example. Pixel (5,4) is elaborated and filtered.

7x7 matrix convolution using the MUL/ADD tree architecture presented in [4]. The tree executes 49 multiplications in parallel and then adds all 49 results. It contains 49 multipliers and 6 adder stages, for a total of 48 adders. Adders and multipliers must support fixed-point representation, because the used kernel values are fractional numbers. Kernel factors are represented in the 0.15 format, assuring a minimal error introduced by the fixed-point approximation. All filtered pixels, represented in 10.15 bit format, are sent both to the *Harris Features Extractor* and to an external memory via a second 32-bit output interface. Storing filtered pixels in an external memory is mandatory since this information is needed during the following features matching phase.

2.2. Harris Features Extractor

The *Harris Features Extractor* implements the Harris corner detection algorithm [8], and applies it on the filtered pixels received from the *Gaussian Filter* block. It outputs a set of extracted features represented as: *feature coordinates* and the related *R-factors* (i.e., each *R-factor* quantifies the robustness of each extracted feature).

The Harris corner detector algorithm is computed through a fully parallelized and pipelined architecture in order to strongly speed up the computation. In this way, this module is able to compute an *R-factor* each clock cycle.

Just the features whose *R-factor* is greater than a given threshold are provided in output. This guarantees that only the features that potentially represent a real corner are propagated to the next module.

The value of the threshold strongly depends on the image

environment type (e.g., Mars or Moon) and condition (e.g., brightness, noise or contrast). To increase adaptation, a self-adaptive threshold is computed frame by frame.
The threshold is computed for the next image based on information on the current image. This is acceptable since, thanks to the high frame rate achievable by our architecture, consecutive frames show marginal differences. Obviously, at startup, to output a valid threshold some cycles are required. During this phase, the *Features Matcher* is unable to produce useful results. However, an experimental campaign on a set of planetary images provided by *Thales Alenia Space S.p.a.*, highlighted that the maximum number of frames required to reach a stable threshold is relatively small (i.e., 9 frames). After this transitory phase, the threshold becomes stable and the *Features Matcher* can start processing the extracted features.

2.3. Features Matcher

The *Features Matcher* (Fig. 4) receives the features extracted by the *Harris Feature Extractor* and finds the set of features that match in two consecutive images.

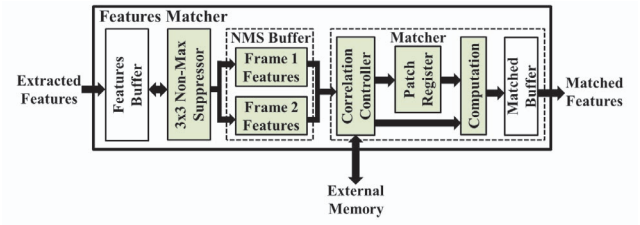


Fig. 4: *Features Matcher* internal architecture

This module adopts two different optimization strategies. The former concerns the matching task, that is performed exploiting un-normalized Cross-Correlation. In this context the high frame rate leads to negligible differences in the conditions (e.g., brightness or contrast) of two consecutive images. Thus, the usage of un-normalized Cross Correlation does not introduce any error in the matching task. In addition, if compared to a NCC approach [13], it leads to a very simple hardware implementation, providing a significant gain in FPGA resources utilization and throughput.
The latter concerns the selection of potentially correlated features. Analyzing the speed of a space-module during the descending phase, and considering the high input frame rate used to sample images, we identified that a feature can perform a maximum movement of 17 pixels between two consecutive images [10]. Thus, two features can be considered as potentially correlated if they are both in a 35x35 pixel neighborhood between the two considered images. Cross-Correlation is therefore computed on these features, only.
The *Features Matcher* receives feature coordinates and associated *R-factor* from the *Harris Feature Extractor*, and stores them in the *Features Buffer* (FB), implemented as a group of BRAMs.
Whenever an entire image is processed and all features are stored in FB, the *3x3 Non-Max Suppressor* scans, for each feature, a 3x3 pixels neighborhood looking for close features. If they are found, just the feature with the highest *R-factor* in this region is marked as valid. To speed up this operation, that would require a complete search into FB, we observed that, in our experimental campaign, no more than 10 features per image row have been identified. Thus, considering that features are obtained analyzing the image row by row and then saved into FB, a neighbor feature will be for sure stored in a (+20, -20) region of FB, centered on the considered feature. This allows us to reduce the neighbor search space and therefore to dramatically decrease the execution time, without increasing area occupation.
The *NMS Buffer* is composed of two sub-buffers (*Frame 1 Features* buffer and *Frame 2 Features* buffer) that are alternatively used to internally store features associated with two consecutive images.
The *Correlation Controller* scans the *Frame 1 Features* buffer and the *Frame 2 Features* buffer and compares the coordinates associated with a feature contained in one of the two buffers with all the coordinates in the other buffer. Whenever two potentially correlated features are found, their un-normalized Cross-Correlation is computed using the intensity of all pixels contained in the two 11x11 pixels windows surrounding the two correlated features.
The 11x11 window related to the first feature is loaded into the *Patch Register*. Then, while the window associated to the feature of the second image is loaded, the cross-correlation is computed "on-the-fly". Each time a new pixel is received from the external memory, it is subtracted from the corresponding pixel of the first image, that is already stored in the *Patch Register*. This operation is performed by the *Computation* module that contains a 25-bit subtractor connected to an accumulator. This approach makes the area occupation of this module independent from the correlation window dimension, making the designer free to select the more appropriate correlation window without any area occupation penalty.
Finally, the Cross-Correlation results are thresholded, in order to eliminate fake-matchings. If the calculated Cross-Correlation value is less than a given threshold, the coordinates of the correlated features are stored inside the internal *Matched Buffer*. This buffer is able to store up to 512 matched features pairs.
Moreover, since a feature of the first image can be correlated to several features of the second image, only the match that has the lowest Cross-Correlation value (i.e., the highest probability to be correlated) is considered valid. This ensures unique matched pairs, and higher quality of matches.

3. EXPERIMENTAL RESULTS

To evaluate the hardware resources usage and the timing performances, the proposed architecture has been synthesized resorting to Xilinx ISE Design Suite 14.4 on a space-qualified Xilinx *Virtex 4-QV VLX200* FPGA device. Post place and route simulations have been done with Modelsim SE 10.0c. Table 1 compares the performances and the area occupation of FEMIP w.r.t. the actual state-of-the-art implementation (FEIC)[5] [6]. Slices and Internal memory of FEIC are reported for a Virtex II device (as in [6]), but the slice and memory architectures are the same as in Virtex 4 family devices. The reported data confirm the great improvements of FEMIP, both in terms of resources usage and speed (i.e., frames per second (*fps*)).

Table 1: Resource Usage for *Xilinx XQR4VLX200 Virtex 4 FPGA device*

	Resource Usage		**Max. Speed**
	Slices	**Internal Memory [KB]**	**[fps]**
FEMIP	9,801	76.5	33
FEIC [6]	25,344	162.5	20
Improvements	**-61.3%**	**-52.9%**	**+65%**

To verify the correctness of the implementation, a software model of FEMIP, written in MATLAB, has been developed. The model reflects exactly the fixed-point data parallelism adopted in the hardware implementation. Results extracted from hardware simulations have been compared with the ones taken from the MATLAB simulation, and equivalence has been verified.

Finally, in order to evaluate the overall performances of the proposed architecture, two parameters, namely *Number of Extracted Matches* (NEM), and *Spatial Distribution Percentage* (SDP), were evaluated for an images dataset (i.e., *Verification Data-set*) provided by *Thales Alenia Space S.p.a.*. The first represents the number of extracted matches between a pair of images, instead the second quantifies the distribution of the matched points on the image. The higher is the value of these parameters, the more accurate will be the motion estimation task.

The *Verification Dataset* is composed of 89 image pairs covering different environmental conditions (i.e., image quality), with different camera movement types, in a synthesized Mars environment.

For the given dataset, the NEM and the SDP are always greater than 100 and 50%, respectively. These data demonstrate that FEMIP always guarantees high matching capability, and well distributed matches.

Unfortunately, a comparison of these parameters with the ones associated with FEIC (i.e., the current state-of-the-art) cannot be performed since the FEIC implementation is not public available.

4. CONCLUSION

This paper presented *FEMIP*, a high performance FPGA-based feature extractor and matcher IP core based on the Harris algorithm. This IP core enables to accelerate the feature extraction and matching task of an EDL system. The effective improvement on both area occupation and performances with respect to the state-of-the-art implementation allows to exploit the free hardware resources to integrate in the FPGA device the complete VBN system.

5. REFERENCES

[1] A. Benedetti and P. Perona. Real-time 2D feature detection on a reconfigurable computer. In *Conference on Computer Vision and Pattern Recognition (CVP)*, pages 586–593, 1998.

[2] C. Cabani and W. MacLean. A proposed pipelined-architecture for FPGA-based affine-invariant feature detectors. In *Computer Vision and Pattern Recognition Workshop (CVPRW)*, pages 121–126, 2006.

[3] Cypress Semiconductor Corporation. *STAR1000 1M Pixel Radiation Hard CMOS Image Sensor*, 2007.

[4] S. Di Carlo, G. Gambardella, M. Indaco, D. Rolfo, G. Tiotto, and P. Prinetto. An area-efficient 2-D convolution implementation on FPGA for space applications. In *6th International Design and Test Workshop (IDT)*, pages 88 –92, 2011.

[5] M. Dunstan, S. Parkes, and S. Mancuso. Visual navigation chip for planetary landers. In *Conference on DAta Systems In Aerospace (DASIA)*, pages 1–7, 2005.

[6] M. Dunstan and M. Souyri. The FEIC development for NPAL project: A core image processing chip for smart landers navigation applications. In *MicroElectronics Presentation Days, ESA/ESTEC*, 2004.

[7] R. González and R. Woods. *Digital Image Processing*. Pearson/Prentice Hall, 2008.

[8] C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, 1988.

[9] P. Nangtin, P. Kumhom, and K. Chamnongthai. Video-based obstacle tracking for automatic train navigation. In *International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pages 21–24, 2005.

[10] Thales Alenia Space S.p.a. Aerospace module speed and trajectory estimation - internal report. Technical report, 2012.

[11] T. Tuytelaars and K. Mikolajczyk. *Local Invariant Feature Detectors: A Survey*. Now Publishers Inc., 2008.

[12] Xilinx Corporation. *Virtex-4 FPGA User Guide*, 2008.

[13] F. Zhao, Q. Huang, and W. Gao. Image matching by normalized cross-correlation. In *International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, pages II:729–732, 2006.