Real time architectures for the Scale Invariant Feature Transform algorithm

G. Doménech-Asensi, .J. Garrigós
Dpto. de Electrónica, Tec. de Computadoras y Proyectos
Universidad Politécnica de Cartagena
Cartagena, Spain
gines.domenech@upct.es

P. López, V. Brea, D. Cabello
Centro de Inv. en Tecnoloxías da Información (CITIUS)
University of *Santiago de Compostela*
Santiago de Compostela, Spain
p.lopez@usc.es

***Abstract*—Feature extraction in digital image processing is a very intensive task for a CPU. In order to achieve real time image throughputs, hardware parallelism must be exploited. The speed-up of the system is constrained by the degree of parallelism of the implementation and this one at the same time, by programmable device size and the power dissipation. In this work, issues related to the synthesis of the Scale-Invariant Feature Transform (SIFT) algorithm on a FPGA to obtain target processing rates faster than 50 frames per second for VGA images, are analyzed. In order to increase the speedup of the algorithm, the work includes the analysis of feasible simplifications of the algorithm for a tracking application and the results are synthesized on an FPGA.**

Keywords—FPGA; VHDL; Scale Invariant Feature Transform

I. INTRODUCTION

There are several proposals in the literature for hardware implementations of the SIFT algorithm. This algorithm is composed by two major stages of computation [1]: keypoint (KP) extraction and descriptor generation. While the first stage requires most of the workload of the whole algorithm, the second one requires complex arithmetic operations. This has led to different synthesis approaches to obtain designs which progressively reach the real time operation. However due to its inherent parallel nature, the natural choice for a real time SIFT implementation are FPGAs. In [2,3], hardware/software co-designs using embedded FPGA processors able to generate descriptors for QVGA and VGA images respectively at 30 frames per second (fps) are described. A pure hardware FPGA based hardware implementations is found in [4] for VGA images and 30 fps. CMOS implementations are proposed in [5, 6], which reach the same frame rate for VGA and HD1080 images respectively. In [7-9], pure hardware FPGA implementations able to generate faster descriptors (above 50 fps) are presented. These last implementations require however some simplifications to the original algorithm. Because of its structure of serial steps, the SIFT algorithm allows its implementation following pipeline architectures, which are easy to implement in FPGAs. Hence the speed will be limited by the slowest stage of the pipe. The following sections propose some implementations to exploit parallelization of the algorithm

II. KEYPOINT EXTRACTION

Fig. 1 shows a proposal of pipeline which exploits parallelism not only at SIFT-step level but also at inter-step level. All the filtering operations, (Gauss), Difference of Gaussians operations (DoG) and neighborhood evaluation to detect extrema points (Neigh) generate a valid pixel value per cycle. Thus the first smoothing operation (Gauss1) requires 3215 cycles to output the first valid pixel and then 640x480 extra cycles to complete the initial image filtering. Then a next image can be processed. Operating as a pipeline, a new image can be processed every 310425 cycles (3,104 ms at 100 MHz).

This work has been partially funded by Spanish government projects TEC2015-66878-C3-2-R (MINECO/FEDER, UE) and TEC2015- 66878-C3-3-R (MINECO/FEDER, UE).

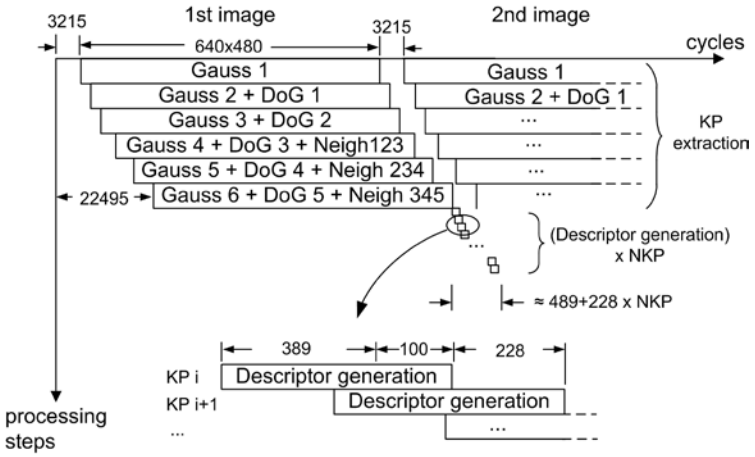

Fig. 1. Chronogram of pipelined SIF algorithm

In order to save computational resources, this implementation performs recursive filtering from the first image instead of absolute filtering. This requires smaller values of σ and also smaller mask sizes: 9x9, 7x7, 9x9, 11x11, 13x13 and 15x15. This allows the use of fewer registers in the sliding window architecture and also fewer processing units. On the other hand, the penalty is an increase in the latency of the pipeline, although it is negligible compared to the throughput achieved. Finally, the use of decomposed two-dimensional Gaussian filtering reduces the number of multipliers and also allows the reuse of intermediate results. Table I shows the device utilization for the first step of the algorithm for a single octave configuration.

TABLE I. DEVICE UTILIZATION SUMMARY (XC6VLX240T)

Logic Utilization	Step 1	All steps	Available
Slice Registers	3130 (1%)	4850 (1%)	301440
Slice LUTs	4404 (2%)	6584 (4%)	150720
fully used LUT-FF pairs	567 (8%)	1318 (13%)	8815
Block RAM/FIFO	63 (15%)	88 (21%)	416
BUFG/BUFGCTRL/BUFHCEs	1 (0%)	1 (0%)	176
DSP48E1s	124 (16%)	129 (16%)	768

A fixed point data format has been used. It uses 8 bits for the integer part. To select the number of digital bits, the performance of the fixed point has been compared with the floating point algorithm implemented in Matlab. Data formats of 2,3,4 and 5 decimal offered 22%, 59.5%, 76.31% and 85.46% precision in the KP identification. A 5 decimal bit was finally implemented.

III. DESCRIPTOR GENERATION

Fig. 2 shows the implementation proposed in [10] to perform the descriptor generation operation. It generates a short descriptor based on 27 elements, which can be valid for tracking applications [11]. Other simplifications employed are the use of Manhattan distance instead of the Euclidean one or 3x3 KP neighborhood and 3-bin histogram instead 4x4 and 8-bin one. Finally it combines CORDIC and Taylor approach to perform the two normalization operations at the end of this stage. Table I shows the device utilization for the proposed implementation (KP extraction + descriptor generation).

Table II shows a performance comparison of different architectures. With the proposed architecture, the first descriptor is obtained after 489 cycles (Fig. 1). Then, a new descriptor is output every 228 cycles. Thus, to obtain 3072 descriptors the number of cycles required is 489+(3071*228)=700677 (7.01 ms at 100MHz). Since this is slower that the KP extraction stage, this is the value which limits the implementation speed. So, for this architecture, the throughput is 143 fps for a single octave and six scales.

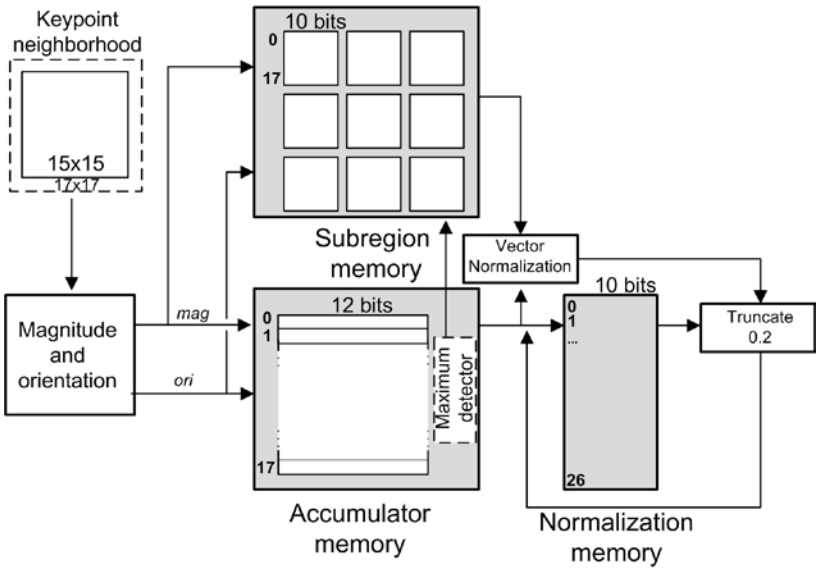

Fig.2. Time-optimized architecture.

TABLE II. PERFORMANCE COMPARISON .

Work	Implementation	Image size	Number of KP	Time (Frame rate)
[2]	Stratix II 2 + NIOS II	320x240	-	33 ms
[3]	Virtex5 + MicroBlaze	640x480	-	31 ms
[7]	Cyclone® II	640×480	-	(56 fps)
[8]	Virtex6	640x480	132100 (per second)	(60fps)
[5]	TSMC 0.18µm CMOS	640×480.	890	33 ms.
[6]	90 nm CMOS	1920x1080	6000	(30 fps)
[9]	Virtex5	512×512	2900	60.64
[4]	Virtex5	640×480	1270	30 fps
This work	Virtex6	640×480	3072	12,5 ms (143 fps *)

* Máximum thoughput exploiting pipeline of stages 1 and 2

Processing of two more octaves would require an increase of 75% of cycles in the first stage if no additional parallelization is done. This yields 543244 cycles (5,432 ms at 100 MHz) which is still faster than the descriptor generation time.

REFERENCES

[1] D. Lowe, “Distintive image features from scale-invariant key-points”, Int. J. Compt. Vis., vol. 60, no. 2, pp 91-110, Nov. 2004.

[2] V. Bonato et al, "A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection," IEEE Trans. on Circ. and Syst. for Video Techn., vol.18, no.12, pp. 1703-1712, Dec. 2008.

[3] L. Yao et al, "An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher," Intl. Conf. on Field-Programmable Technology, 2009. FPT 2009, pp.30-37, 9-11 Dec. 2009.

[4] M. Qasaimeh, A. Sagahyroon and T. Shanableh, "FPGA-Based Parallel Hardware Architecture for Real-Time Image Classification," in IEEE Trans. on Computational Imaging, vol. 1, no. 1, pp. 56-70, March 2015

[5] F.C. Huang, et al, "High-Performance SIFT Hardware Accelerator for Real-Time Image Feature Extraction," IEEE Trans. on Circ. and Syst. for Video Tech., vol.22, no.3, pp. 340-351, Mar. 2012.

[6] L. C. Chiu et al, "Fast SIFT Design for Real-Time Visual Feature Extraction," in IEEE Trans. on Image Processing, vol. 22, no. 8, pp. 3158-3167, Aug. 2013.

[7] K. Mizuno et al., “Fast and low-memory-bandwidth architecture of SIFT descriptor generation with scalability on speed and accuracy for VGA video,” in Proc. IEEE FPL, 2010, pp. 608–611.

[8] W. Deng et al, "An efficient hardware architecture of the optimised SIFT descriptor generation," in Proc IEEE FPL, 2012, pp. 345-352.

[9] J. Jiang, X. Li and G. Zhang, "SIFT Hardware Implementation for Real-Time Image Feature Extraction," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 24, no. 7, pp. 1209-1220, July 2014.

[10] P. Leyva et al., "Simplification and hardware implementation of the feature descriptor vector calculation in the SIFT algorithm," in Proc. IEEE FPL, 2014, pp. 1-4.

[11] S. Gauglitz, T. Höllerer and·M. Turk, "Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking", Int J Computer Vision, vol. 94, no 3, pp 335-360, Sep. 2011.