Zynq FPGA based and Optimized Design of Points of Interest Detection and Tracking in Moving Images for Mobility System

Abdelkader BEN AMARA[1]
University of Monastir, LEME Laboratory, University of Monastir, Tunisia
and LITIS laboratory, University of Rouen Normandy, France

Mohamed ATRI[2]
LEME Lab, Department of Physics
University of Monastir,
Tunisia

Edwige PISSALOUX[3], Richard GRISEL[4]
LITIS Laboratory
University of Rouen Normandy & CNRS FR 3638
Rouen, France

***Abstract*—In this paper, an FPGA based mobile feature detection and tracking solution is proposed for complex video processing systems. Presented algorithms include feature (corner) detection and robust memory allocation solution to track in real-time corners using the extended Kalman filter. Target implementation environment is a Xilinx Zynq SoC FPGA based. Using the HW/SW partitioning flexibility of Zynq, the ARM dual core processor performance and hardware accelerators generated by Xilinx SDSOC and Vivado HLS tools improve the system ability of processing video accurately with a high frame rate. Several original innovations allow to improve the processing time of the whole system (detection and tracking) by 50% as shown in experimental validation (tracking of visually impaired during their outdoor navigation).**

Keywords—Feature detection; harris & stephens corner detector; tracking; extended kalman filter; HW/SW partitionning; zynq SoC; computer vision; memory access; ARM A9; HLS; Interlacing; blancking; progressive video

I. Introduction

The detection and tracking of interest points and corners in real-time from video frames is a key operation in computer vision applied to autonomous mobile systems (e.g. robotics, humanoid robotics, assistive devices for visually impaired). Harris corner detection algorithm is usually used for such characteristics detection because of its algorithmic simplicity and expected level of detection accuracy [1]. For feature tracking, several candidates exist, however only few of them respect the constraints of video real-time. Kalman filter is one of them [2].

Real-time features detection and tracking application raise. The real-time constraints induce additional algorithmic extrinsic (exogenous) constraints as they require larger computation throughput. Furthermore, other requirements such as processing time, used resources, flexibility of implementation (software-hardware co-design) and power consumption, should be also considered for wearable solutions.

FPGAs provide such flexible platform for implementation of algorithms dedicated to wearable systems. Moreover, they simplify and reduce the design cycle, enhance system processing speed and reduce resources demand.

In [3], an architecture for moving object detection and tracking system based on SoPC is proposed; an Altera Cyclone II FPGA platform and NIOS II processor are used. The processing capacity of this implementation need to be confirmed as the NIOS II processor has limited frequency, key parameter especially for processing of large amount of data.

In [4] a solution of task assignment and complex data flow in a system implemented on a mixed DSP and FPGA hardware environment is proposed. Such solutions are possible as new generation of FPGA devices offer unprecedented processing and scalling capabilities:

- New FPGA (FPGA SoC (System of Chip)) allow to implement parallel processing with different programming paradigms having high degree of parallelism on structures such as grids, trees, pyramids with different control strategies (SIMD, MIMD, MISD);
- Unprecedented programmable logic resources [5-6];
- Fast real-time processing, up to billions of MAC operations and memory locations via BRAMs [7];
- Very large memory, up to one gigabyte of DDR3 (SDRAM) [8];

Moreover, the FPGA manufacturers provide support for a sustainable design, such hardware opens the market of big data processing and especially the image processing and computer vision [9] [10].

The efficient and optimal exploitation of available hardware resources become more and more difficult and impose a heavy design load, especially when using the traditional bottom-up design approach. To overcome this complexity, soft processors are proposed which adopt a software design methodology in order to make the FPGA-based design more accessible. Therefore, the co-design and the reasonable hardware-software partition of an algorithm elements play a fundamental role for simultaneous tuning of processing's functional architecture, circuit's architecture and circuit's targeted performance. It fully harnesses the programmable logic performance and the flexibility of soft processing.

ZYNQ-7000 FPGA based SoC family developed by Xilinx displays the scalability, robustness and flexibility of FPGA technology, although it provides high computational performance and easy exploitation [11]. The Zynq-7000 SoC combines a programmable logic FPGA based part and a Cortex-A9 dual-core ARM processor which is completely independent of the programmable units and it can run the software part of the Zynq. The processing subsystem in composed by different peripherals and memory controllers; the programmable logic is composed by millions of programmable units dedicated to implement custom accelerators and expand processing subsystem using its rich bandwidth [3] [12]. We use this platform for hardware-software balanced implementation of algorithms for detection and tracking of corners in video sequences.

Therefore the whole paper is organized as follows. Section II outlines the formal framework for corner detection and their tracking algorithms. Section III proposes a hardware-software co-design of these algorithms implementation in ZYNQ-7000 FPGA SoC. Section IV describes the proposed optimization of memory management of ZYNQ-7000 FPGA SoC. Section V addresses some experimental results, while Section VI concludes the paper and lists its potential extensions.

II. Moving Points of Interest Detection and Tracking: Algorithmic Approach

A. Moving Points of Interest Detection: Harris Corner Detection based

The Harris corner detection [13] is an improvement of the Moravec algorithm [14]. Using pixels' luminosity function derivatives, organized into an auto-correlation matrix, it determines, for each pixel, whether it is a corner or not. The whole computation process requires the following steps:

1) Gradient derivatives computing: The horizontal I_x and the vertical derivatives I_y of a pixel (x,y) luminosity are obtained after the image local convolution with predefined (3x3) masks approximating gradients as shown in (1) and (2).

$$I_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \times I \tag{1}$$

$$I_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \times I \tag{2}$$

The gradients for all pixels of an image are obtained by the convolution of the whole image with the above masks.

2) Computation of the Auto-Correlation matrix: Starting from calculated derivatives and using Taylor series formula, we assumed that are shifting factors over the window w_{xy} where the local interest of the pixel (x,y) is computed by (3).

$$E_{u,v}(x,y) = \sum_{u,v} w_{u,v}\left[I(x+u, y+v)\right]^2 \tag{3}$$

After applying the Taylor formula, derivatives variation can be expressed as shown in (4) and (5) merged in (6). From the simplifications, one can determine the auto correlation matrix presented in (6).

$$I(x+u, y+v) = I(x,y) + I_x u + I_y v + O\left(u^2+v^2\right) \tag{4}$$

Wherein,

$$I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}, I_{xy} = \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

After merging (3) and (4), we obtain (5)

$$\begin{aligned} E_{u,v}(x,y) &= \sum_{u,v} w_{u,v}\left[I_x u + I_y v + O\left(u^2+v^2\right)\right]^2 \\ &\approx \sum_{u,v} w_{u,v}\left[I_x u + I_y v\right]^2 \\ &= \sum_{u,v} w_{u,v}\left[(I_x u)^2 + 2I_{xy}uv + (I_y v)^2\right]^2 \\ &= \sum_{u,v} w_{u,v}[u,v]\begin{bmatrix} I_x{}^2 & I_{xy} \\ I_{xy} & I_y{}^2 \end{bmatrix}^2 \begin{bmatrix} u \\ v \end{bmatrix} \\ &= \sum_{u,v} w_{u,v}[u,v] M \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned} \tag{5}$$

Wherein, auto-correlation matrix is (6):

$$M = w_{u,v}\begin{bmatrix} I_x^2 & I_{xy} \\ I_{xy} & I_y^2 \end{bmatrix} \tag{6}$$

3) Gaussian filtering (smooting: Gaussian filter aims to reduce noise by smoothing the image window; a 3x3 Gaussian kernel mask is applied to the auto-correlation matrix (6) [13].

4) Harris corner "R" response computing and thresholding: The corner response value, also called R-response is highly important to decide whether the pixel will be considered as a corner or not. Equation (7) presents the equation of R , where *det(M)* is the determinant of the auto-correlation matrix M, *TraceM* is its trace and *k* is an empirical factor that typically varies between 0.04 and 0.06 [16].

$$R = \det M - k(TraceM)^2 \tag{7}$$

The λ_1 an λ_2, the characteristics coefficients (eigenvalues) of matrix M, give information about the "interest" value of the pixel (x,y) in the window. Using the relations: $DetM = \lambda_1\lambda_2$ and $TraceM = \lambda_1 + \lambda_2$, the computational formula for the interest value of a pixel (x,y) in its neighborhood w_{xy} is (8):

$$E_{u,v}\left(x,y\right)=D^{-1}\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}D \tag{1}$$

Large values of both characteristics coefficients correspond to maximum of the auto-correlation. This implies that edge curvature variation of the corresponding pixel is big at any direction and the Harris corner response will be at its maximum (and will be more important than the applied threshold); therefore, this pixel is a detected corner.

5) Non-Maximum suppression & Kalman filtering: Kalman tracking is initialized and used to predict the size information and the object location.

Essentially, Kalman filter is an estimation of system behavior (9) through systems states and observations [3] (9):

$$\begin{cases}x_k = A_{x_{k-1}} + w_{k-1}\\ z_k = Hx_k + v_k\end{cases} \tag{2}$$

Where the state vector at the instance t is presented by x_k Process and measurement noises are presented by w_{k-1} an v_k . Measurement vector is z_k , A and H presents respectively the process and observation matrices.

It this paper, we consider two state vectors called V_{s1} and V_{s2} to calculate the size and position prediction of each detected corner. Their corresponding observation vectors are P_1 and P_2 (10-11):

$$\begin{cases}V_{s1} = \left[x,\Delta x,y,\Delta y\right]^T\\ P_1 = \left[x,y\right]^T\end{cases} \tag{3}$$

$$\begin{cases}V_{s2} = \left[u,\Delta u,w,\Delta w\right]^T\\ P_2 = \left[u,w\right]^T\end{cases} \tag{4}$$

Where: $x,y,\Delta x,\Delta y$, express centroid position of a target feature, relative to the movement along x and y directions.

Process and observation matrices are shown in (12):

$$A=\begin{bmatrix}1 & \Delta t & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & \Delta t\\ 0 & 0 & 0 & 1\end{bmatrix}, H=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix} \tag{5}$$

Where:

$$\Delta t = t_k - t_{k-1} \tag{6}$$

III. Hardware-Software Implementation in ZYNQ-7000 FPGA of Interest Point Tracking

Zynq-7000 Xilinx SoCs combines a reconfigurable FPGA area, which represent the programmable logic (PL), with a processing system (PS) part composed by dual-core A9 ARM processor [12]. An accurate embedded system should have the best adequation between hardware acceleration blocks and software execution and processing management, which called a HW/SW co-design environment. Due to the timing and system's speed constraints, hardware acceleration became mandatory, as algorithms implemented in PL will need less execution time compared to pure software-based implementation. Based on these advantages of FPGA, we can accelerate the PS bottlenecks on Zynq SoC using the PL [15]. The proposed architecture was designed and implemented on a Zynq-7020 ZEDBOARD Xilinx HW/SW environment: its internal architecture is shown in Figure 1.

A. *Zynq-7000 Data Communication Capabilities*

The whole system design is centered on the processing system with a programmable logic extension of the (PS). The primary and principle interface between the programmable logic and the processing system is a set of communication protocols composed of multi channels, called AXI bus interfaces [11] [12]. The role of these dedicated communication protocols and interfaces is to perform a convenient and fast data interaction while processing. To control small amount of data and hang on the accelerator registers (control, address, data) implemented on the (PL) part, the (PS) use a general-purpose interface called AXI_GP. But in case of processing and manipulating large amount of data in a high rate communication between (PS) and (PL), dual core ARM A9 processing system uses a controller called DMA accessible using a high performance AXI interface able to manage 32 and 64-bit data width.

B. *Proposed HW/SW partitioning for a Higher Accuracy and Increased Processing Speed*

This part outlines the HW/SW partitioning of the whole system and its sub-systems (Harris corner detection and Kalman tracking).

HW/SW partitioning of the whole system.

When dealing with high rate communication speed and large amount of data, three solutions are possible:

1) Run the algorithm in the processing system part completely. However, due to the limited parallelism capabilities of the PS, time and flexibility performance will be bounded;

2) Implement the complete algorithm in hardware (PL). The system will be fast and accurate but the design complexity will be very important;

3) Find an efficient adequacy between the algorithm and the architecture (Zynq-7000 in our case) to achieve the desired performance.

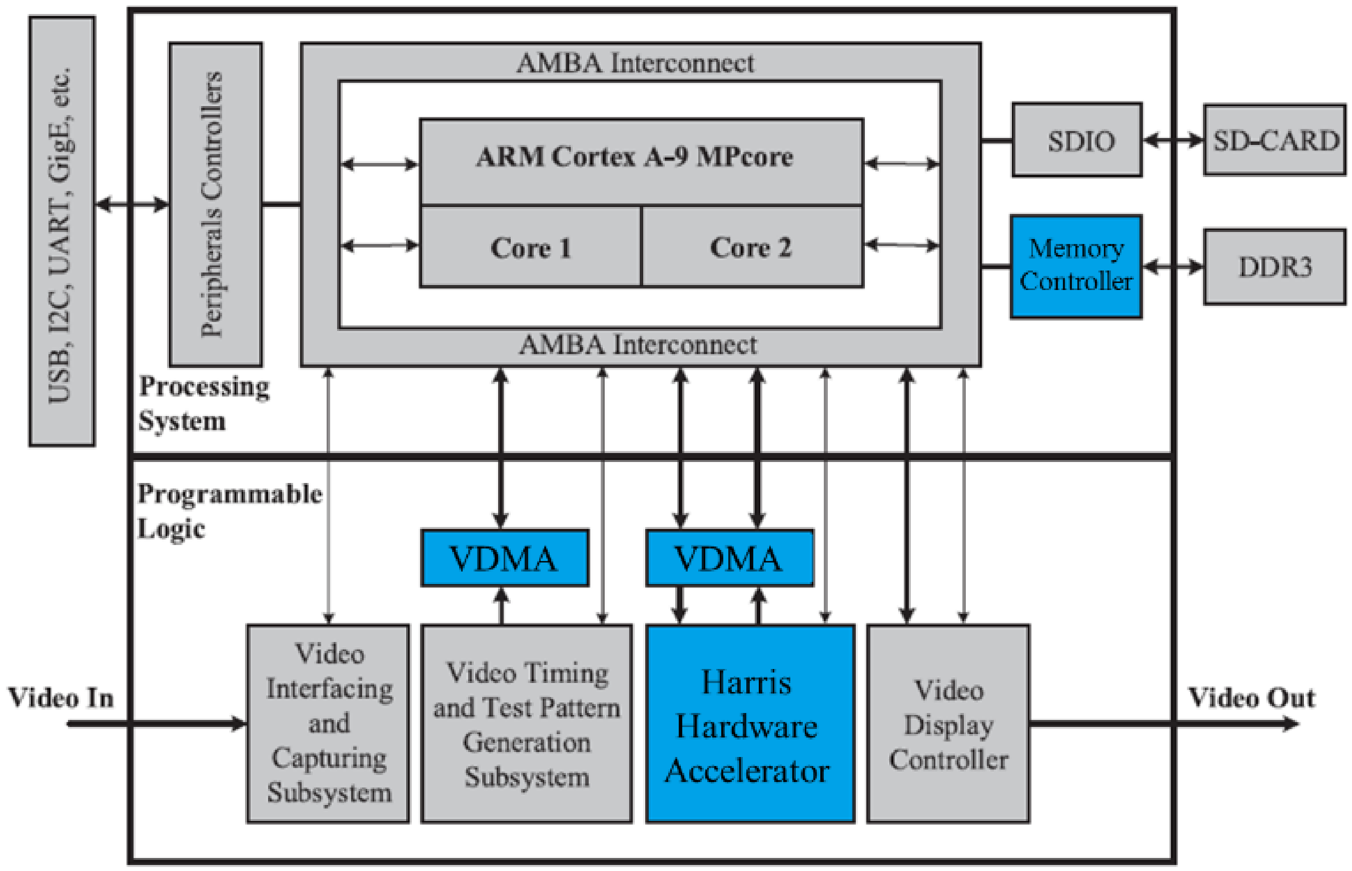

Fig. 1. Proposed Implementation of Ocrner Detection and Tracking in Video Sequences.

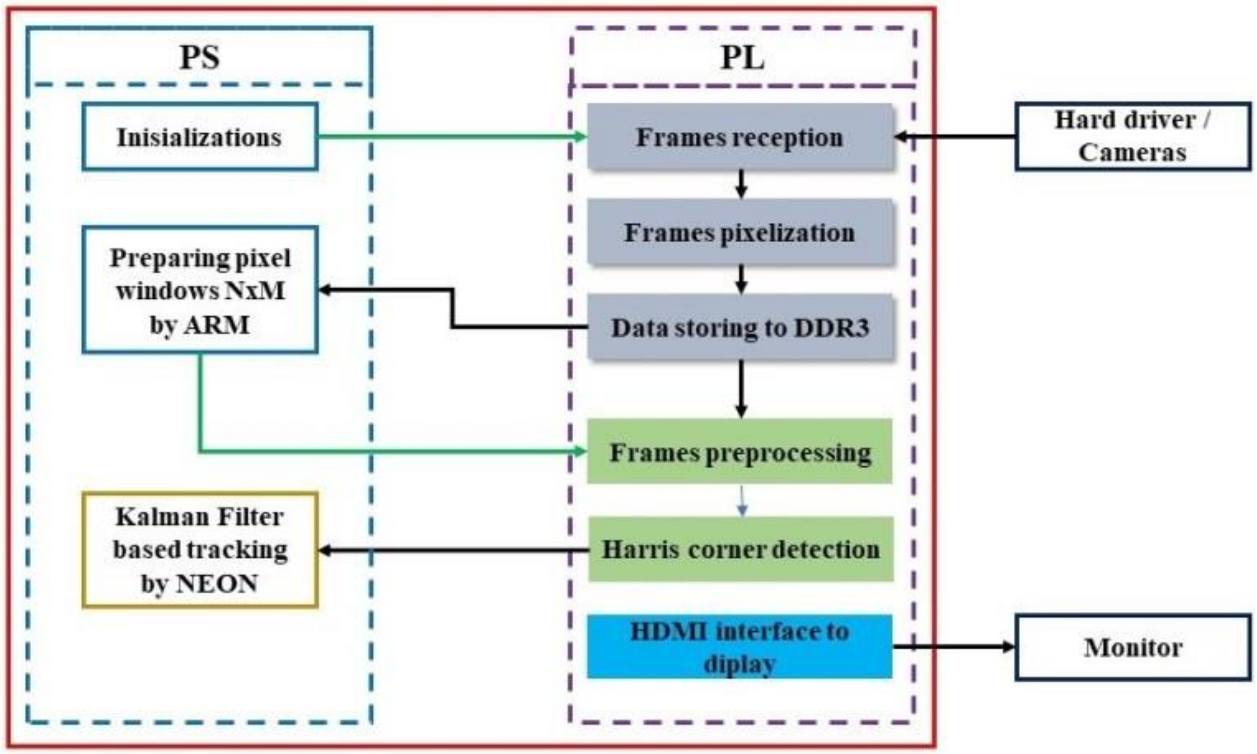

Fig. 2. Hardware-Software Partitionning of the Proposed Processing for Znyc 7000 Soc.

Taking into consideration the Zynq SoC architecture and our algorithm characteristics, we proposed the following hardware and software partitionning of our processing and shown in Figure 2. The system automatic boot and initialization is ensured by software component, and this is using the ARM-A9 processor 1 (which runs an embedded Xilinx Linux distribution dedicated to prepare de hole system to receive data and start the processing). Video data is streamed from cameras through an HDMI external interface, to the programmable logic part (PL). Frames then will be converted to windows of pixels through hardware frame buffers and directly stored in a DDR3 memory through AXI interfaces. Usage of the (PL) to preprocess video data and store them enhances system temporal performance.

Based on the fact that the volume of calculations in Harris corner detection operation is more important than Kalman filter, we proposed to implement it in (PL) part of the Zynq Soc and to assign Kalman based tracking to the ARM based (PS) part.

- Harris corner detection implementation

As discussed in part I, FPGA implementation permits to generate a fully parallel and high-speed core of the algorithm, but classic RTL design of complex algorithms like Harris is hard and needs a long time to validate the implementation. In order to overcome this limitation and improve system development, we used a high-level synthesis methodology based on OpenCV image processing library and Xilinx Vivado HLS design environment. Using this design flow, we generate Harris corner detection accelerator starting from a C++ based description of the algorithm. Figure 3, presents a block diagram of Harris detection generated IP core.

- Kalman based detected corners tracking

Cortex dual core A9 processors run an ARMv7-A set on instructions; it is armed with a high performance floating point SIMD media processing engine called (MPE). This engine is perfectly suitable to execute a multiple range of applications [8] [17]. The advanced single instruction multiple data (SIMD) NEON instructions can give the program the capability to execute image processing applications at an fast rate. For our system, the NEON and the ARM-A9 dual cores are used to accelerate Kalman based detected corners tracking. NEON engine parallelism philosophy is based on the instruction set dedicated to the most significant characteristic.

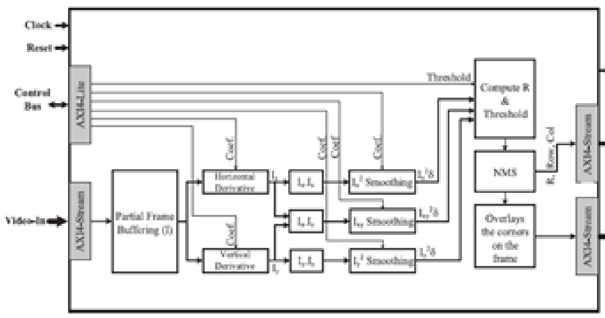

Fig. 3. System Proposed Architecture.

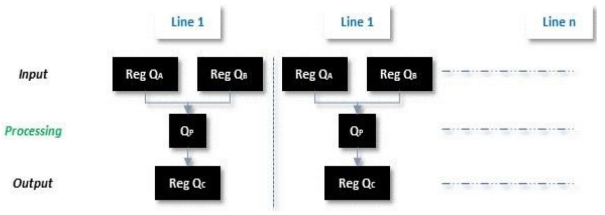

Fig. 4. SIMD Processing Cycle Inside the NEON MPE

Also, this engine has dedicated registers: Q_A and Q_B are input registers that contains N individual input vectors for each [12], as illustrated in Figure 4. A single defined operation is executed between the N input vectors to produce a corresponding set out output vectors which are buffered to an output register called Q_C . Each register can directly load 128 bits from the DDR3. In our case, detected corners will be loaded by the NEON registers for Kalman tracking execution and finally will be stored back to the memory for display.

IV. Optimisation of Memory Resources

In this section, a set of custom blocks integrated in the architecture are presented. Their role is to optimize memory resources needed, improve processing speed and enhance systems security level.

- Frames cropping module

As we are working using 720f resolution, it is necessary to manage frame pixels loaded to the FPGA, so it is necessary to load only the relevant and necessary data for processing achievement. The solution proposed here highly optimizes memory resources dedicated to frames loading and preprocessing, and minimizes the processing time and the energy consumption.

To design an optimized solution it is mandatory to understand the structure of a 720p frame.

720p resolution, also called standard HD, is a progressive video signal with a dimension 1280x720 pixel per frame : 720 horizontal lines and 1280 vertical columns. However, the real 720p frame is bigger than that, because it contains vertical and horizontal blanking, dedicated for many functionalities like ancillary data insertion. Our innovative idea is to load only active region of the frame and ignore all the blanking. To do this we developed a *frame-cropping module*. This block is in charge on the frame cropping functionality. It crops frames starting from pixels data present in the input we called VIDEO_IN, according to a given reference cropping origins, which are SAV flag also known as Start of Active Video, and EAV flag also known as End of Active Video.

Our cropping module could also crop pre-selected areas from frames (through X, Y coordinates), which could be updated through its dedicated registers. Origins are called them CROPPING_X and CROPPING_Y. In addition, the size of the initial frame are detected in real time by two dedicated inputs FRAME_SIZE_X and FRAME_SIZE_Y, that ensure detection in parallel with data streaming. Cropping module specifications are given in Figure 5.

- Frozen video detection module for video loss management.

As we are working on systems that aims to secure, visually impaired peoples, we have think about the systems or camera bug or any kind of incident that could prevent the system to be functional. One of the issues it the frames freeze. So we integrate in our architecture a block the we'ne developed and that deal with frozen frames. Frozen video detection module is designed to assert a flag called "VIDEO_FROZEN" when the video stream is broken or frozen for at least "NB_FROZEN_FRAME" number of frames, see Figure 7. User fixes this number and dedicated registers could modify it. When video is declared as frozen, the information's is sent to a sequencer, which is responsible to act on its outputs accordingly.

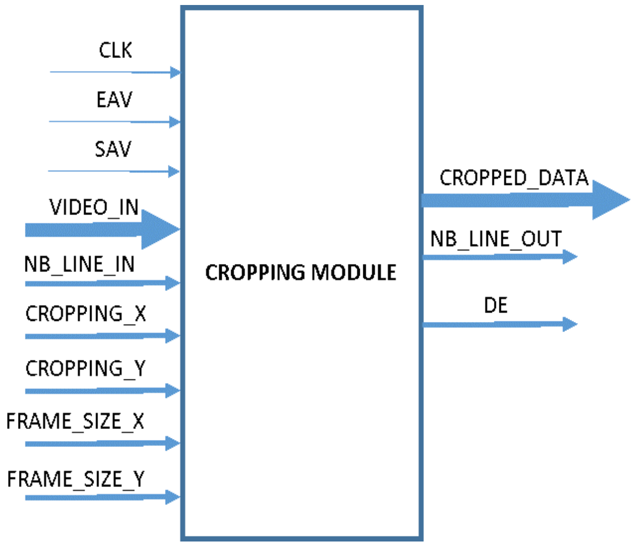

Fig. 5. Cropping Module Specifications.

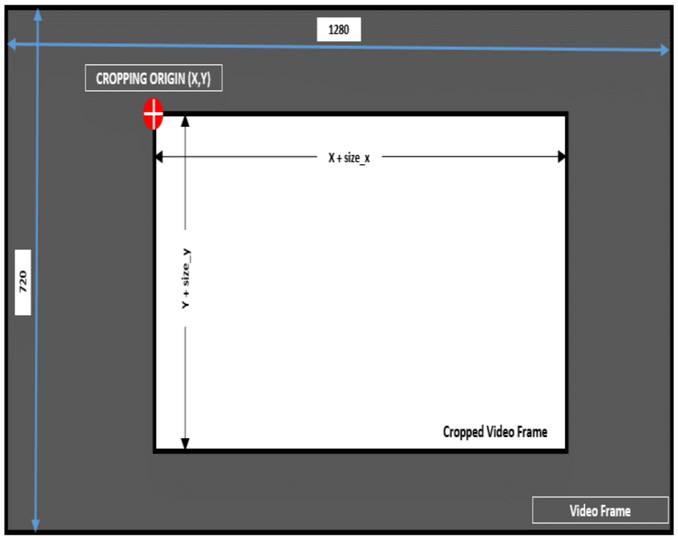

Fig. 6. Frame Cropping using the Proposed HDL Module.

Cropped frame is defined by the Figure 6.

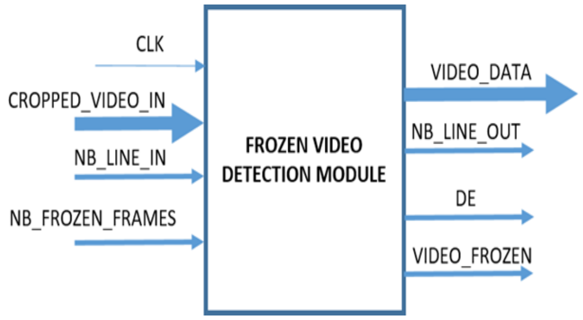

Fig. 7. Video Loss Management Module.

Our management module covers two respective video loss situations: No video input and identical video pattern input.

- No video input

In this case, video signals are inactive; VIDEO_FROZEN flag is active and transmits the information to the sequencer mentioned bellow. This last checks the FIFO dedicated to memorization of loaded video frames. When the FIFO is empty, it continues to transmit black frames.

- Identical video pattern input

A mechanism has been designed to detect several identical input frames, which is considered as an unacceptable behavior, and present a danger for user and the system. To detect and react in case of this incident, a CRC is calculated for every entire current frame and compared with the CRC of the next frame and so on, for a predefined number of frames.

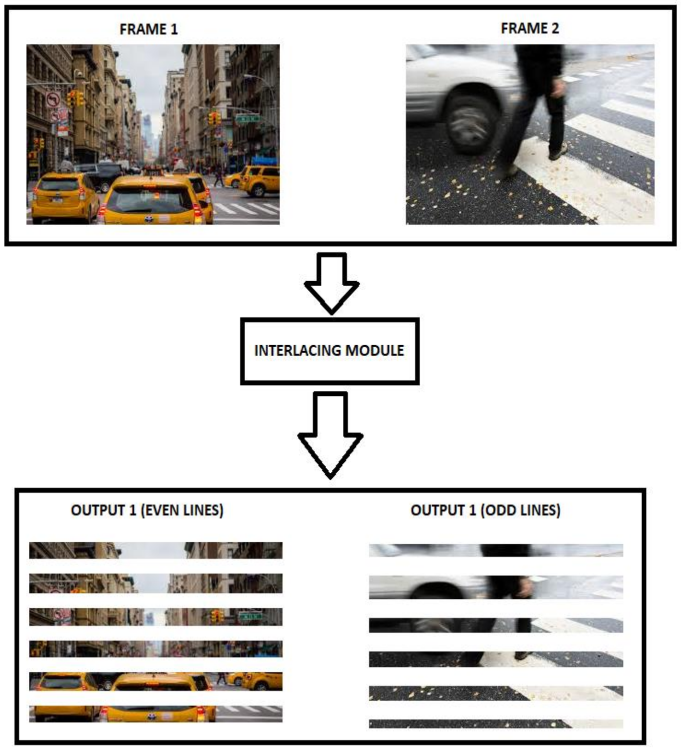

Fig. 8. Interlacing Function.

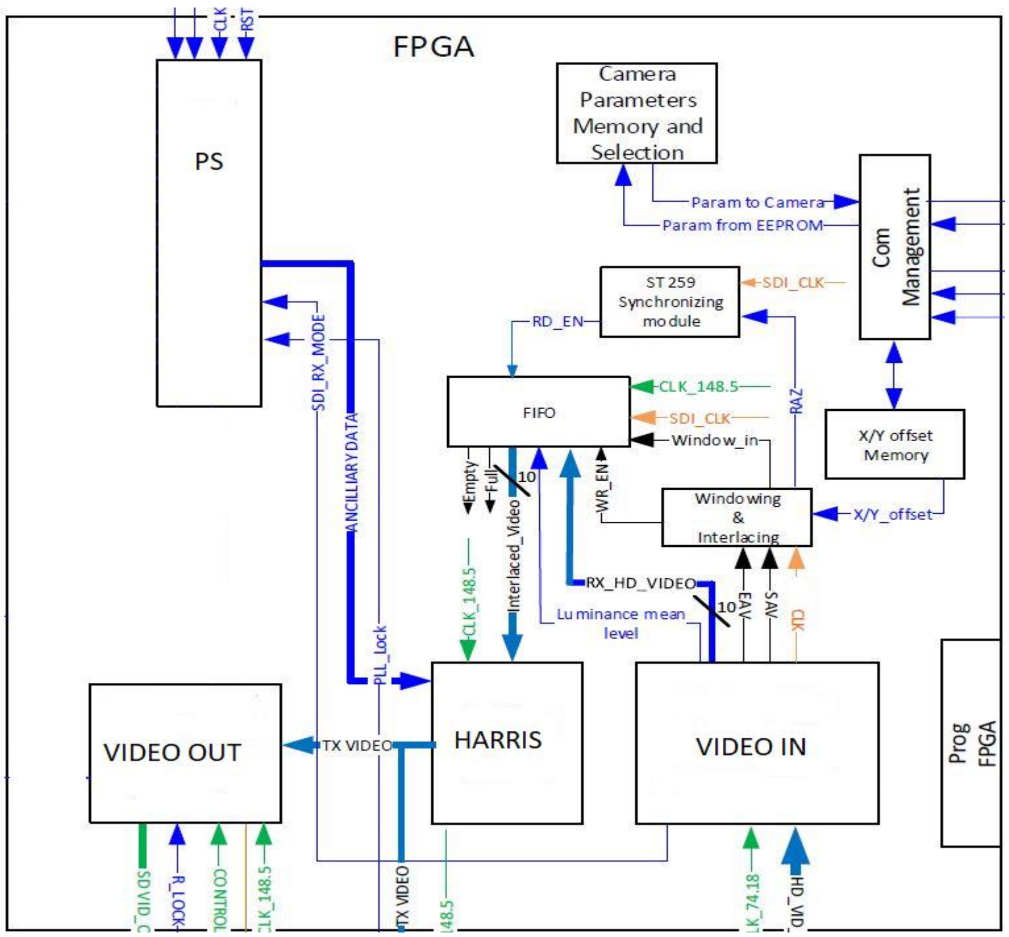

Fig. 9. Global System Architecture.

- Frames interlacing

The video interlacing is another technique for data optimization processing. This technique, never considered for mobile systems, improves by 50% the requirements for energy consumption, processing time and memory size. The idea here is to split every frame into odd and even lines, so for a frame N we take only odd lines and for frames N+1 we load only even lines, and so on. Split frames will alternate at a very high speed between odd and even, what guarantees the claimed improvement.

As shown in Figure 8, the module receives the first frame, the odd lines are transmitted and even lines are discarded. On the next frame, the even lines are transmitted and odd lines are discarded.

The architecture block diagram become as it is illustrate in Figure 9.

V. Results

In this section, some of tests of moving features detection and tracking are realized based on ZEDBOARD Xilinx platform and the system architecture presented in Figures 1, 2, 3 and 4. For implementation we've used a Zynq-7000 based Zedboard platform. It is composed by a 512 Mb RAM, 667 MHz clock frequency ARM A9 based dual core processor that represents the (PS), a 100 MHz frequency (PL) and multiple input and output interfaces. 720p and 1080p video frames are received from a Dell i5-6200U 2.30 GHz laptop via HDMI Input/output interface.

The real experiments track visually impaired persons (VIP) in outdoor scenes. The VIP detects some obstacle using a classic white cane. The video system tracks the VIP and obstacles (represented by interest and edge points – their density creates an illusion of continuous lines). If the VIP is close to an obstacle a specific real-time signal indicates him/her the obstacle. Figures 10 and 11 propose the very first results of the VIP navigation at two different instances. The high quality of the tracking of object features (interest points and edges) can be observed.

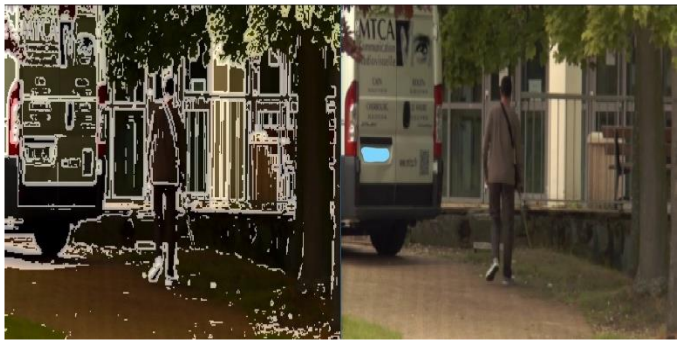

Fig. 10. Points of Interest and of Edge Detection and Tracking (t=0).

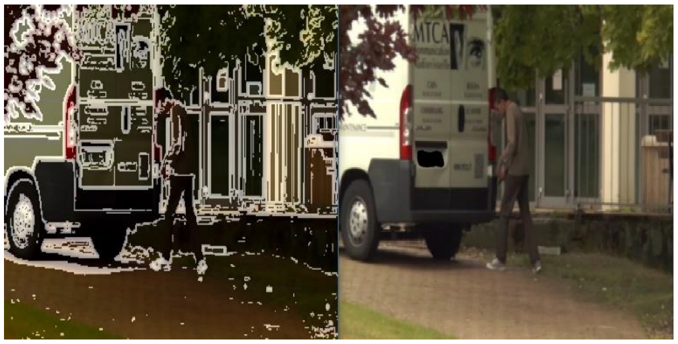
Fig. 11. Points of Interest and of Edges Detection and Tracking (t=t+n).

The proposed implantation of video tracking on the target system requires 6300 LUTs and 11 18Kb BRAMs of the (PL). Each frame was processed in 638 milliseconds, including data preprocessing, Harris accelerator execution and tracking algorithm with NEON based acceleration. These precision and processing performances permit to integrate our proposed architecture in visually impaired mobility assistive devices.

VI. Conclusion

The paper addresses two topics.

1) A proposition of a robust real time embedded architecture for detection and (Kalman) tracking of (Harris) interest points. Our architecture combines the performances of the Zynq-7000 hardware resources and the flexibility of the software partition based on and ARM A9 dual core processor. The hardware –software implementation of targeted algorithms was co-designed and effectively implemented on FPGA based Xilinx Zynq SoC.

2) Architecture performance enhancement and memory optimization.

We proposed two innovative techniques in order to minimize the quantity of processed data which impact the processing power, memory occupation and processing time. There are : interleacing and blancking cropping.

To enhance our system's security and precision, we proposed a HDL modul that aims to avoid making wrong decisions or display bad or wrong data. This module detects the non presence of frames and detect freezed frames.

The temporal performances of our approach are roughly 50% better than these of classic implementations.

The experimental evaluation with VIP people navigating in outdoor scene clearly show that the system may be integrated in mobility assistance for VIP.

The proposed architectures will be integrated in the VIP mobility control devices such as an "intelligent cane" which will allow to avoid unexpected obstacles. Other test scenarios, which several moving obstacles will be also tested.

We hope that through detailed evaluations new co-design rules would be possible to establish and integrated in an automatic tools for hardware-software implementations.

References

[1] B. Hdioud, A. Ezzahoul, Y. Hadi, R. Oulad Haj Thami, "A real-time people tracking system based on trajectory estimation using single field of camera view", Int. Conf. on Computer Applications Technology (ICCAT 2013), 30 May 2013, pp. 1 – 4.

[2] S. Yang, M. Baum, "Extended Kalman filter for extended object tracking", IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2017), 19 June 2017, pp. 4386 – 4390.

[3] Li Yuejing, Xie Weichen, Shi Yixing, Chen Changmin, "Real time moving object detection and tracking system based on SoPC", J. of Chongqing University of Technology, Vol 25, No.4, Apr. 2011.

[4] Liu Weining, "Object tracking under complicated background based on DSP+FPGA platform", Chinese J. Liquid Crystals and Displays, Vol. 29, No. 6. Dec 2014.

[5] 7 Series FPGAs Configurable Logic Block User Guide 474 v1.8, Xilinx Inc, September 2016.

[6] Zynq-7000 All Programmable SoC Overview Data Sheet 190 v1.10, Xilinx Inc, September 2016.

[7] 7 Series FPGAs Memory Resources User Guide 473 v1.12, Xilinx Inc, September 2016.

[8] Zynq-7000 SoC and 7 Series Devices Memory Interfaces Solutions User Guide 586, Xilinx Inc, April 2014.

[9] B.A. Draper, J.R. Beveridge "Accelerated image processing on FPGAs," IEEE Transactions on Image Processing, Vol. 12, No. 12, December 2003, pp. 1543-1551.

[10] G. Bailey, "Design for Embedded Image Processing on FPGAs," John Wiley & Sons Ltd, June 2011, pp. 1-416.

[11] Xilinx Inc, UG585, "Zynq-7000 all programmable SoC", 2017.

[12] A. Ben Amara, E. Pissaloux, M. Atri, "Sobel edge detection system design and integration on an FPGA based HD video streaming architecture", 11th Int. Conf. on Design & Test Symposium (IDT 2016), 6 February 2016, pp. 1-5.

[13] C. Harris, M. Stephens, "A combined corner and edge detector", 4th Conference of Alvey Vision, Manchester, England, August 1988, pp. 147-151.

[14] Moravec, H., Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover, Master Thesis, Stanford University 1980.

[15] Xilinx Inc. DS190, "Zynq-7000 extensible processing platform overview", 2012.

[16] M. Aydogdu, M. Demirci, C. Kasnakoglu, "Pipelining Harris corner detection with a tiny FPGA for a mobile robot", IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen, China, December 2013, pp. 2177-2184.

[17] Xilinx Inc, Xapp 1078, "Simple AMP running linux and bare-metal system on both zynq SoC processors"UG585, 2013