FPGA-based smart camera mote for pervasive wireless network

Cédric Bourrasset, Jocelyn Sérot, François Berry

FPGA-based Smart Camera Mote for Pervasive Wireless Network

Cédric Bourrasset, Jocelyn Sérot, François Berry
Institut Pascal, UMR 6602 CNRS, Université Blaise Pascal, Clermont-Fd, France

***Abstract*—Smart camera networks raise challenging issues in many fields of research, including vision processing, communication protocols, distributed algorithms or power management.**

The ever increasing resolution of image sensors entails huge amounts of data, far exceeding the bandwidth of current networks and thus forcing smart camera nodes to process raw data into useful information. Consequently, on-board processing has become a key issue for the expansion of such networked systems.

In this context, FPGA-based platforms, supporting massive, fine grain data parallelism, offer large opportunities. Besides, the concept of a *middleware*, providing services for networking, data transfer, dynamic loading or hardware abstraction, has emerged as a means of harnessing the hardware and software complexity of smart camera nodes.

In this paper, we prospect the development of a new kind of smart cameras, wherein FPGAs provide high performance processing and general purpose processors support middleware services. In this approach, FPGA devices can be reconfigured at run-time through the network both from explicit user request and transparent middleware decision. An embedded real-time operating system is in charge of the communication layer, and thus can autonomously decide to use a part of the FPGA as an available processing resource.

The classical programmability issue, a significant obstacle when dealing with FPGAs, is addressed by resorting to a *domain specific* high-level programming language (CAPH) for describing operations to be implemented on FPGAs.

I. Introduction

The main challenges in wireless smart camera networks come from the limited capacity of network communications. Indeed, wireless protocols such as the IEEE 802.15.4 protocol target low data rate, low power consumption and low cost wireless networking in order to fit the requirements of sensor networks. Since nodes more and more often integrate image sensors, network bandwidth has become a strong limiting factor in application deployment. This means that data must be processed at the node level before being sent on the network. Solutions based on general purpose processors (GPPs) can not cope with real-time processing constraints in a high-resolution camera environment. On the other hand, FPGA-based platforms, supporting massive and fine grain data parallelism, offer large opportunities for gobbling up streaming data coming from high resolution image sensors. But, if FPGAs provide a suited hardware architecture to meet the constraints of high resolution real-time computation, they raise programmability issues, especially in the context of collaborative programming development.

Indeed, collaborative programming is a key issue when dealing with wireless smart camera networks. These networks now typically rely on a decentralized approach, where processing is carried out on-board. Furthermore, the integration of smart camera motes into the "Internet of Things" is viewed as a desirable property. This means that a manager must be able to address requests on end devices using the Internet Protocol. This approach requires flexibility, scalability, dynamic retasking and hardware abstraction at the programming level. In response, middleware layers have been introduced in order to abstract physical devices and provide logical distributed computing services. Most middlewares for smart camera networks are based upon the *mobile agents* programming paradigm [1]–[4]. A mobile agent is a software unit which has the ability to move autonomously through a computer network. In this approach, developers write applications as independent modules ("agents") and the system dynamically distributes the modules through the network, taking decision to migrate agents from node to node.

While relatively easy to implement on architectures built on GPPs, the mobile agents programming model approach raises significant technical difficulties on architectures using FPGAs because agents are no longer pieces of source (or binary) code but static hardware descriptions. Fortunately, last generation of FPGA devices support dynamic partial reconfiguration. This means that the implemented functionality can be updated without having to stop and reprogram the circuit entirely. However, current solutions are based on modules that can be loaded or unloaded at run-time by a soft/hard-core processor [5], [6]. With this approach, available modules have to be stored in a pre-compiled library and algorithms must be built from components taken in this library. A more flexible approach should allow the integration of ad-hoc components, fully specified by the application programmer. But this in turn raises significant programming issues because developing components for FPGAs is typically done using hardware description language (HDLs) and requires skills in digital design.

In this paper, we describe a smart camera architecture, and an associated programming methodology, addressing the aforementioned problems. This architecture is specifically designed to support real-time image processing in a high resolution sensor context, dynamically reconfigurable through the network, abstracted on the "Internet of Things" and can be programmed with a high-level language. The proposed architecture integrates both a GPP and a FPGA component. The GPP - a microcontroller - is in charge of networking and hardware abstraction. The FPGA performs heavy data processing. FPGA programming is achieved by using a high-level domain-specific language (DSL), introduced in [7] and called CAPH. Based on the dataflow programming model, CAPH can effectively bridge the gap between high level

specifications and efficient hardware implementations.

The remainder of the paper is organized as follow. The motivations and objectives sketched in this introduction are detailed and related to the state of the art in Section II. The proposed architecture is presented in Section III and the associated programming methodology in Section IV. Section V concludes with a presentation of the current state of the work deriving from this project, including the actual hardware platform on which some ideas are prototyped to demonstrate the effectiveness of the approach.

II. Motivations

Hardware architectures for visual sensors network have been reviewed in many surveys [8], [9]. On-board processing is essential for recent camera platforms. Most existing platforms are based on GGP units [10], [11]. This kind of architecture offers high level programming and modern embedded processors provide good performances but cannot meet real-time processing constraints when image resolution increases. Using hardware configurable devices provides massive computing capability. For example, the Cyclops platform [12] was one of the first solution including a CPLD to perform image processing. The CPLD was in charge of memory control and could perform a limited amount of low-level image processing such as background subtraction or frame differentiation.

Since then, FPGA-based architectures, offering more resources and functionalities than CPLD, have been the subject of tremendous attention for implementing image processing applications on smart camera nodes [13]–[15]. For instance, the NICTA smart camera [15] uses a CMOS sensor with a 2592x1944 resolution and is equipped with a Xilinx FPGA for image processing and a MicroBlaze core as network processor.

However, if the introduction of FPGAs can address some performance issues in smart camera networks, it introduces new challenges concerning node programmability and/or hardware abstraction. Indeed, in the current state of the art, programming FPGAs is done using low-level hardware description languages (HDLs). The GestureCam platform [14], for example, relies on VHDL to program the FPGA. This severely limits the programmability of these solutions because it requires skills in digital design. Classical solutions to increase FPGA programmability relies on high-level synthesis (HLS) tools [16]. Unfortunately, in the general case, HLS tools have difficulties in generating efficient circuit implementations from high-level behavioral specifications because of the semantic gap between the two levels of description. A possible approach to circumvent this problem is to focus on a specific application domain, in which dedicated tools can rely on application-specific abstractions to provide an optimized compilation path from high-level specifications down to low-level descriptions. For instance, the Trident compiler [17] is a tool focusing on the implementation of floating-point scientific computing applications. Another example is the CAPH compiler [18], [19] for real-time streaming processing applications described in the sequel.

The second issue raised by the introduction of FPGAs in smart camera architectures concerns the ability to provide a network-level abstraction of the provided functions in order to facilitate distributed applications development. Typically, hardware abstraction and distributed computing rely on middleware layers, as exemplified in Impala [20] or TinyCubus [21] for wireless sensor networks. A middleware for distributed vision applications has been recently introduced in [22]. The proposed framework provides a set of services, following the approach advocated in the "Internet of Things" concept and a Service Oriented Architecture. To support these middleware specifications, the node-level architecture has to be run-time updatable. Since FPGA supports partial dynamic configuration, middleware support can become suitable. However, current solutions are based on fixed libraries of predefined modules forcing applications to be dependent on the library content, and thus hindering flexibility [5], [6].

What is really needed is a FPGA-based platform offering high-capacity for image processing, but also being fully configurable and updatable. This architecture must also be addressable from the Internet protocol and abstracted as a simple resource in order to simplify distributed vision development.

III. Architecture proposal

A. Hardware Architecture

The proposed hardware architecture for the smart camera mote is sketched on Fig. 1. It embeds both a microcontroller and a FPGA device. The FPGA is the core of the system. It controls the image sensor, and implements all low-level image processing tasks. For this, it can use up to six external SRAM blocks of 1 Mo each, addressable in parallel.

The microcontroller is in charge of network management and executes the middleware layer, providing a bridge between the application protocol and the hardware processing. The application protocol will request information from the node and the microcontroller will reconfigure the FPGA in order to obtain the corresponding features. Physical network connection is provided classically by a IEEE 802.15.4 transceiver.

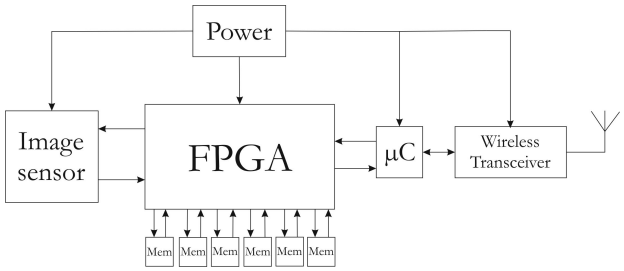


Fig. 1: System overview

The top level logical architecture is depicted in Fig. 2. It consists of three IP blocks: one providing the interface to the image sensor; a second dedicated to memory management; and a third implementing the actual image processing, as a network of dataflow actors. The specification of this network is done using the CAPH domain-specific language and the corresponding IP is produced automatically by the CAPH compiler (See Sec. IV).

Algorithm prototyping is carried out off-line. When completed, the application layer can request any node to execute this algorithm. For this, the corresponding FPGA configuration

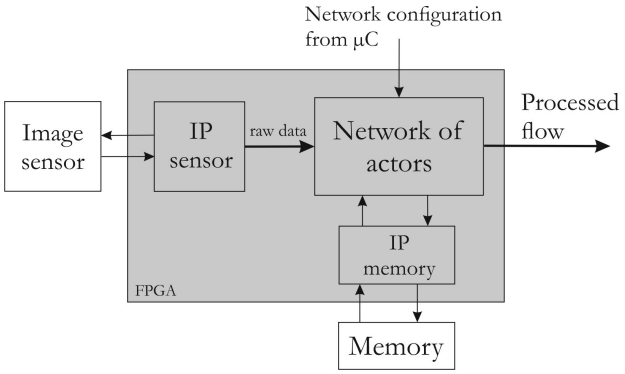


Fig. 2: FPGA logical architecture

will be sent to the targeted node through the network. The microcontroller will then dynamically reconfigure the FPGA unit with the received configuration. As soon as this reconfiguration has occurred, the new extracted features are pulled up to the network manager layer.

B. Software Architecture

Fig. 3 shows the top-level software architecture implemented on the microcontroller. It is based upon the ERIKA Real Time OS. ERIKA offers a rich library for communication, control, sensor data handling, vision processing and has a low flash memory footprint (1-4 Kb). It is also the first open-source kernel certified OSEK/VDK (automotive). The kernel layer of ERIKA contains a set of modules implementing task management and real-time scheduling policies. Available policies are fixed priority with preemption threshold and EDF[1] with preemption threshold. Both use a Stack Resource Protocol (SRP), a protocol that allows sharing resources between threads and sharing the system stack among all the threads while preserving time predictability.

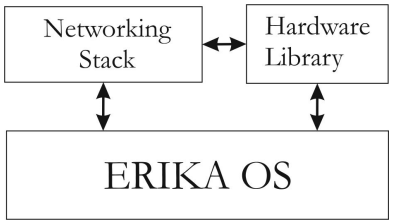


Fig. 3: Software overview

In order to be addressable in the "Internet of Things" world, our smart camera mote implements a network protocol based on IPv6 and CoAP protocols (Fig. 4). The physical layer and media access control layer meet the IEEE802.15.4 specifications, for a low-rate wireless personal area network, with a 10-meter range and a transfer rate of 250 kbit/s. The 6LoWPAN layer is an adaptation layer for IPv6 allowing the transmission of IPv6 packets over IEEE 802.15.4 networks. UDP is the classical transport layer. The application layer protocol will be an implementation of CoAP (Application Protocol for Constrained Networks/Nodes). CoAP protocol is an HTTP-like protocol allowing the creation of embedded web services, based on RESTful services [23]. According to this network protocol, mote architectures are abstracted and viewed as abstract resources by the application layer.

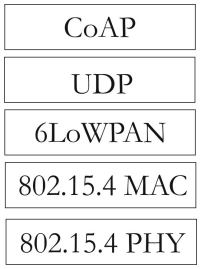


Fig. 4: Networking Stack

The last element of the software architecture is a hardware library usable both by the Erika OS and the networking stack. This library will contain all the FPGA configurations implementing the image processing services that can be provided by the node. Each requested service coming from the network manager must be present in the library to be honored. If not, the corresponding FPGA configuration must be sent along with the CoAP request. On the node side, an ERIKA task will be in charge of receiving requests, potentially with a new FPGA configuration, and of reconfiguring a part of the FPGA to provide the requested service.

IV. Programming Methodology

A. Programming model and tools

Image processing resources are specified as networks of dataflow actors. This is done at a very high level of abstraction using the CAPH programming language. CAPH [7], [19] is a domain specific language, offering a fully-automated compilation path from high-level dataflow descriptions to FPGA configuration for stream-processing applications. Applications are described as networks of computational units, called actors, exchanging streams of tokens through FIFO channels. Interaction between actors is strictly limited to token exchange through channels, so that the behavior of each actor can be completely described in terms of actions performed on its inputs to produce outputs (no side effect, strictly local control).

The current tool chain supporting the CAPH language is shown on Fig. 5. It comprises a graph visualizer, a reference interpreter and a compiler producing both SystemC and synthetizable VHDL code.

The compiler is the core of the system. Compilation of a CAPH program is done in two steps. An elaboration step first generates a target-independent intermediate representation (IR) of the program, which is then processed by specific back-ends. The VHDL back-end is in charge of generating the FPGA configuration for hardware synthesis. The SystemC back-end produces cycle-accurate SystemC code which is used to provide back-annotations to customize and optimize the VHDL code

The reference interpreter is based on a fully formalized semantics of the language, written in axiomatic style. Its role

[1]Earliest Deadline First (EDF): Priority increases dynamically when the deadline comes closer

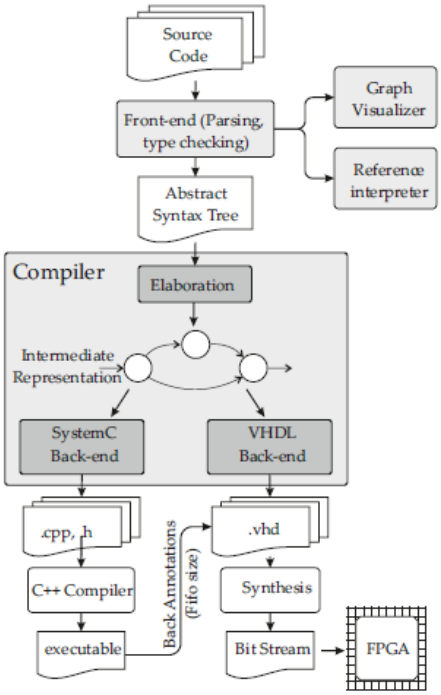


Fig. 5: CAPH Toolset

is to provide reference results to check the correctness of the generated SystemC and VHDL code. It can also be used to test and debug programs, during the first steps of application development (in this case, I/O streams are read from/written to files). Several tracing and monitoring facilities are provided. For example, it is possible to compute statistics on channel occupation or on rule activation.

The CAPH chain of tools provides an effective rapid prototyping environment for FPGA programming, allowing new image processing services to be prototyped and added to the network with a minimum effort from the application programmer and a good efficiency (compared to hand-crafted HDL code).

B. Toward the Internet of Things

The proposed mote architecture has been designed in order to be integrated in the "Internet of Things". Since the embedded microcontroller runs the CoAP application protocol, each device in the network will provide information concerning its own embedded services. A virtual control room (VCR) - implemented for example as a web application - will collect information in a database system about CoAP end-devices, for instance IP address with associated services. The web application will use this database to address node requests, thus offering true distributed computing. To add a new service making use of FPGA-based image processing, users can develop algorithm using the CAPH tool chain. Once compiled and downloaded, the corresponding FPGA configuration will be included in the list of provided services. Moreover, when a new node will be deployed in the network, it will be automatically discovered by the application layer thanks to CoAP protocol specifications.

V. Current State of Work

The smart camera architecture proposed in this paper is the basis of an actual hardware platform currently under development. This platform builds on some existing realisations, most noticeably a smart camera developed at our institute and called DreamCam.

A. DreamCam Architecture

The DreamCam platform is depicted in Fig. 6. This platform is currently working as a stand-alone smart camera for feature extraction. It will be extended in order to integrate the network management facilities and middleware services of the proposed architecture. This extension will take advantage of the modular architecture of the platform, which is composed of boards stacked on a FPGA chip (see Fig. 6). This modular architecture allows us to easily switch boards, for example, from USB to Giga-Ethernet communication board.

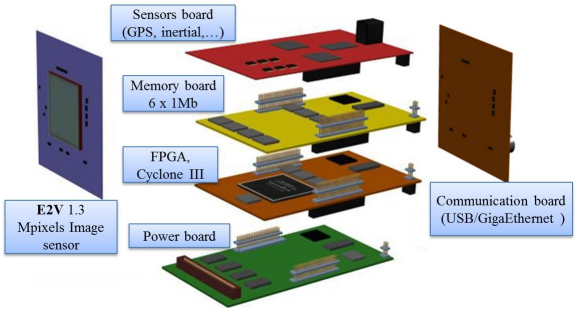


Fig. 6: Overview of the DreamCam Camera

The DreamCam platform is equipped by a 1.3 Megapixels active-pixel digital image sensor from E2V, supporting sub-sampling/binning and multi Region of Interests (ROIs). The FPGA is a Cyclone-III EP3C120 FPGA manufactured by Altera. This FPGA is connected to 6x1MBytes of SRAM memory blocks wherein each 1MB memory block has a private data and address buses (hence programmer may address six memories in parallel). The FPGA also embeds a CMOS driver to control the image sensor and a communication controller (USB or Giga-Ethernet). In order to migrate towards the proposed architecture, a new board will be stacked on top of the current platform. This board will integrate the microcontroller providing the network level services and the wireless transceiver.

B. A sample application

In order to assess the validity of the proposed architecture, we started the development of a testbench application. At this stage of progress, the goal is to validate the overall programming methodology.

Because the proposed smart camera node will be deployed in an Intelligent Transport System context, we targeted a vehicle detection application. This application aims at providing a simple information about vehicle presence in the camera field of view. It is based on the computation of the histogram of oriented gradients (HOG) features. Dalal and Triggs have showed in [24] that HOG descriptors provide

excellent performance for visual object recognition. Since then, HOG descriptors have been commonly used and many FPGA implementations have been proposed [25]–[27]. We started from an implementation developed by Bauer [25], originally designed for pedestrian detection on a GPU-FPGA architecture. We adapted the algorithm to vehicle detection and decided to implement the full algorithm in the FPGA.

1) Overview of the method: The HOG feature extraction chain is sketched in Fig. 7. It is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. The idea is to characterize an object appearance by the distribution of local intensity gradients. HOG detection are basically composed of a HOG feature extraction followed by a classification step. For now, only the feature extraction step has been implemented on the FPGA.

Fig. 7: HOG feature extraction algorithm

In this implementation, a sliding detection window is used to locate the initial position of the vehicle in the field of view. This window is shifted over the image and for each position, a HOG feature set is generated from the corresponding image patch and evaluated by a pre-trained classifier that categorizes unknown samples into one of the predefined classes (vehicle or non-vehicle). This detection technique is often considered unfeasible due to its heavy computing requirements. Here, we take advantage of massive fine-grain parallel computing facilities provided by the FPGA device. Moreover, if we decide to work on a specified region of interest (ROI), we will take advantage of our programmable image sensor capable of working in multi-ROI and this HOG implementation will still be usable.

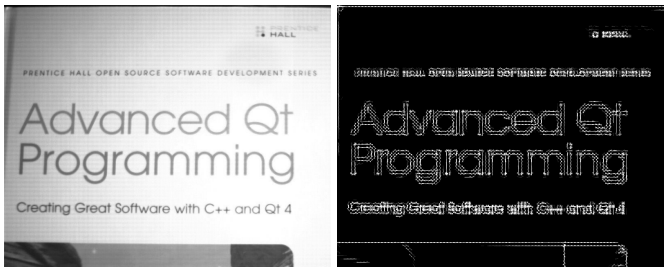
2) Implementation results: This implementation of the HOG feature extraction step on FPGA of the DreamCam platform was carried out using the CAPH toolset. Real-time execution results are reported in Fig. 8. Hardware resources usage level for Altera Cyclone III EP3C120 are given in table I for a 1280x960 pixel resolution.

TABLE I: Hardware resource usage level for a processing resolution of 1280x960 px

Operation	Hardware resources	Memory usage
Gradient	1500 LE	20 kbit
Histogram	19000 LE	726 kbit
Block normalization	2000 LE	24 kbit
Total (usage level)	22500 LE (20%)	770 kbit (19%)

The HOG descriptor extraction step requires 20% of the full capacity of the FPGA. As said before, the classification step has not been implemented yet. Results in table I show that the histogram generation is the most consuming sub step due to 8x8 pixels buffering.

In order to verify HOG feature extraction, we also developed an interface which takes HOG descriptors coming from the camera and displays the corresponding image patch (8x8 pixels patch in this implementation - Fig. 8).



(a) Input Image (b) HOG features

Fig. 8: HOG feature extraction

VI. Conclusion

We proposed a new architecture of wireless smart camera node where a FPGA provides high processing capacity to meet the demands of high-resolution image processing. This architecture also includes a General Purpose Processor to make the mote addressable from the Internet protocol. This specification makes our smart camera able to implement any algorithm on condition that it can expressed as a network of computational units. The embedded network facilities provided by the architecture offer abstraction of the physical architecture. On top of the network stack, the application layer defines a list of the embedded web services that are provided by the node. A supervisor application collects available services to realize the final distributed application. When the application requires a new image processing to be implemented, we provide an automatic tool chain for FPGA development based on a high level language, avoiding the classical difficulties associate to low-level programming. To activate a service in the targeted mote, the application supervisor sends an Internet request to the on-board GPP which dynamically reconfigures the embedded FPGA to provide the required service.

The proposed architecture is currently under development but preliminary results, obtained with a simplified version of the hardware and the software toolset, are very encouraging. Basing the final version of the platform both on existing extensible hardware and on innovative software makes us rather confident about the feasibility of the project.

Acknowledgment

This work has been sponsored by the French government research programm "Investissements d'avenir" through the IMobS3 Laboratory of Excellence (ANR-10-LABX-16-01), by the European Union through the program Regional competitiveness and employment 2007-2013 (ERDF Auvergne region), and by the Auvergne region. This work is made in collaboration with the Networks of Embedded Systems team at CNIT (Consorzio Nazionale Interuniversitario per le Telecomunicazioni)

References

[1] H. Qi, Y. Xu, and X. Wang, "Mobile-agent-based collaborative signal and information processing in sensor networks," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1172–1183, 2003.

[2] M. Chen, T. Kwon, Y. Yuan, and V. Leung, "Mobile agent based wireless sensor networks," *Journal of Computers*, vol. 1, no. 1, 2006. [Online]. Available: https://www.academypublisher.com/~academz3/ojs/index.php/jcp/article/view/01011421

[3] K. Ross, R. Chaney, G. Cybenko, D. Burroughs, and A. Willsky, "Mobile agents in adaptive hierarchical bayesian networks for global awareness," in *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, vol. 3, 1998, pp. 2207–2212 vol.3.

[4] M. Molla and S. Ahamed, "A survey of middleware for sensor network and challenges," in *Parallel Processing Workshops, 2006. ICPP 2006 Workshops. 2006 International Conference on*, 2006, pp. 6 pp.–228.

[5] D. Ziener, S. Wildermann, A. Oetken, A. Weichslgartner, and J. Teich, "A flexible smart camera system based on a partially reconfigurable dynamic fpga-soc," in *Proceedings of the Workshop on Computer Vision on Low-Power Reconfigurable Architectures at the FPL 2011*, 2011, pp. pp 29–30.

[6] A. Oetken, S. Wildermann, J. Teich, and D. Koch, "A bus-based soc architecture for flexible module placement on reconfigurable fpgas," in *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, 2010, pp. 234–239.

[7] J. Serot, F. Berry, and S. Ahmed, "Implementing stream-processing applications on fpgas: A dsl-based approach," in *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, 2011, pp. 130–137.

[8] W. W. B. Rinner, *Towards Pervasive Smart Camera Networks*. Academic press, 2009.

[9] S. Soro and W. Heinzelman, "A survey of visual sensor networks," *Advances in Multimedia*, 2009.

[10] P. Chen, P. Ahammad, C. Boyer, S.-I. Huang, L. Lin, E. Lobaton, M. Meingast, S. Oh, S. Wang, P. Yan, A. Yang, C. Yeo, L.-C. Chang, J. D. Tygar, and S. Sastry, "Citric: A low-bandwidth wireless camera network platform," in *Distributed Smart Cameras, 2008. ICES 2008. Second ACM/IEEE International Conference on*, 2008, pp. 1–10.

[11] W.-C. Feng, E. Kaiser, W. C. Feng, and M. L. Baillif, "Panoptes: scalable low-power video sensor networking technologies," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 1, no. 2, pp. 151–167, May 2005. [Online]. Available: http://doi.acm.org/10.1145/1062253.1062256

[12] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: in situ image sensing and interpretation in wireless sensor networks," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, ser. SenSys '05. New York, NY, USA: ACM, 2005, pp. 192–204. [Online]. Available: http://doi.acm.org/10.1145/1098918.1098939

[13] I. Bravo, J. Balias, A. Gardel, J. L. Lzaro, F. Espinosa, and J. Garca, "Efficient smart cmos camera based on fpgas oriented to embedded image processing," *Sensors*, vol. 11, no. 3, pp. 2282–2303, 2011. [Online]. Available: http://www.mdpi.com/1424-8220/11/3/2282

[14] T. Tsui and Y. Shi, "An fpga-based smart camera for gesture analysis for healthcare applications," in *Consumer Electronics, 2008. ICCE 2008. Digest of Technical Papers. International Conference on*, 2008, pp. 1–2.

[15] E. Norouznezhad, A. Bigdeli, A. Postula, and B. Lovell, "A high resolution smart camera with gige vision extension for surveillance applications," in *Distributed Smart Cameras, 2008. ICES 2008. Second ACM/IEEE International Conference on*, 2008, pp. 1–8.

[16] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, "High-level synthesis for fpgas: From prototyping to deployment," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 30, no. 4, pp. 473–491, 2011.

[17] J. L. Tripp, K. D. Peterson, C. Ahrens, J. D. Poznanovic, and M. Gokhale, "Trident: An fpga compiler framework for floating-point algorithms," in *Proceedings of the 2005 International Conference on Field Programmable Logic and Applications (FPL), Tampere, Finland, August 24-26, 2005*, T. Rissa, S. J. E. Wilton, and P. H. W. Leong, Eds. IEEE, 2005, pp. 317–322.

[18] J. Serot, F. Berry, and S. Ahmed, "Caph: A language for implementing stream-processing applications on fpgas," in *Embedded Systems Design with FPGAs*, P. Athanas, D. Pnevmatikatos, and N. Sklavos, Eds. Springer New York, 2013, pp. 201–224. [Online]. Available: http://dx.doi.org/10.1007/978-1-4614-1362-2_9

[19] The Caph Programming Language home page. [Online]. Available: http://dream.univ-bpclermont.fr/index.php/caph

[20] T. Liu and M. Martonosi, "Impala: a middleware system for managing autonomic, parallel sensor systems," in *Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming*, ser. PPoPP '03. New York, NY, USA: ACM, 2003, pp. 107–118. [Online]. Available: http://doi.acm.org/10.1145/781498.781516

[21] P. Marron, A. Lachenmann, D. Minder, J. Hahner, R. Sauter, and K. Rothermel, "Tinycubus: a flexible and adaptive framework sensor networks," in *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, 2005, pp. 278–289.

[22] P. Pagano, C. Salvadori, S. Madeo, M. Petracca, S. Bocchino, D. Alessandrelli, A. Azzar, M. Ghibaudi, G. Pellerano, and R. Pelliccia, "A middleware of things for supporting distributed vision applications," in *Proceedings of the 1st Workshop on Smart Cameras for Robotic Applications, SCaBot Workshop*, 2012.

[23] L. Richardson and S. Ruby, *Restful web services*, 1st ed. O'Reilly, 2007.

[24] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *International Conference on Computer Vision & Pattern Recognition*, C. Schmid, S. Soatto, and C. Tomasi, Eds., vol. 2, June 2005, pp. 886–893. [Online]. Available: http://lear.inrialpes.fr/pubs/2005/DT05

[25] S. Bauer, S. Kohler, K. Doll, and U. Brunsmann, "Fpga-gpu architecture for kernel svm pedestrian detection," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, 2010, pp. 61–68.

[26] R. Kadota, H. Sugano, M. Hiromoto, H. Ochi, R. Miyamoto, and Y. Nakamura, "Hardware architecture for hog feature extraction," in *Proceedings of the 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, ser. IIH-MSP '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1330–1333. [Online]. Available: http://dx.doi.org/10.1109/IIH-MSP.2009.216

[27] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, ser. CVPR '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 1491–1498. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2006.119