Computational Self-Awareness as Design Approach for Visual Sensor Nodes

Zakarya Guettatfi*, Philipp Hübner†, Marco Platzner* and Bernhard Rinner†
* Computer Science Department
Paderborn University, Germany
Email: zakarya@ubp.de, platzner@upb.de
† Institute of Networked and Embedded Systems
Alpen-Adria-Universität Klagenfurt, Austria
Email: philipp.huebner@aau.at, bernhard.rinner@aau.at

***Abstract*—Visual sensor networks (VSNs) represent distributed embedded systems with tight constraints on sensing, processing, memory, communications and power consumption. VSNs are expected to scale up in the number of nodes, be required to offer more complex functionality, a higher degree of flexibility and increased autonomy. The engineering of such VSNs capable of (self-)adapting on the application and platform levels poses a formidable challenge.**

In this paper, we introduce a novel design approach for visual sensor nodes which is founded on computational self-awareness. Computational self-awareness maintains knowledge about the system's state and environment with models and then uses this knowledge to reason about and adapt behaviours. We discuss the concept of computational self-awareness and present our novel design approach that is centred on a reference architecture for individual VSN nodes, but can be naturally extended to networks. We present the VSN node implementation with its platform architecture and resource adaptivity and report on preliminary implementation results of a Zynq-based VSN node prototype.

Keywords: Reconfigurable platforms; visual sensor nodes; self-awareness; distributed embedded systems; performance-resource trade-off

I. Introduction

Visual sensor networks (VSNs) consist of smart camera nodes which integrate the image sensor, the processing system, and the wireless transceiver as an embedded device. Smart cameras [1], [2] are able to analyse image data locally and extract relevant information, to collaborate with other cameras on application-specific tasks, and to provide the user with information-rich descriptions of captured events [3]. These networks of spatially distributed smart camera devices are becoming increasingly important in areas such as surveillance and security, entertainment, assisted living and home automation. Current trends show that compared to today's installations future VSNs will scale up in the number of nodes and be required to offer more complex functionalities, a much higher degree of flexibility, and an increased autonomy. Functionality is not only getting more complex due to ever more advanced algorithms but also due to an increasing amount of captured data that needs to be processed. Flexibility is needed not only because the applications change during the VSNs' operation, but also because the network topology and the collective resources of the distributed system can vary strongly during runtime. Increased autonomy allows VSNs to take configuration decisions without external control, e.g., adapting the configuration upon changes in the network or input data, which will facilitate deployment, operation and maintenance phases for large networks.

VSNs are implemented on distributed embedded computing platforms [4], [5]. In the last years, reconfigurable-system-on-chip platforms have emerged that provide designers with a cost-effective way to implement their applications on embedded multi-cores using standard software ecosystems and, additionally, to integrate customised functions as reconfigurable hardware modules for increased performance or energy-efficiency. The ability to explore the resulting *trade-offs in performance, energy and hardware resource usage at runtime* is key to meeting the demands of future VSNs.

In this paper we introduce *computational self-awareness* [6] as a design approach for VSNs that is able to deal with these trade-offs simultaneously at node and network levels. Self-awareness is a well-studied concept in the fields of psychology and cognitive science (e.g., [7], [8]), but these concepts have only recently been transferred to the computing domain [6], [9]. The key novelty of this design approach lies (i) in the integration of (self-)adaptivity and reconfigurability at the application and platform level and (ii) in the support of adapting the complete hardware layout comprising the number and size of hardware cores and the interconnect between them in response to VSN application changes.

The remainder of this paper is organised as follows. Section II discusses related work on self-aware computing systems and camera nodes on reconfigurable platforms. In Section III, we introduce the design approach based on computational self-awareness and the reference architecture for a VSN node. Section IV presents our hardware platform with its levels of adaptivity and reports on preliminary implementation results. Section V concludes the paper with a summary and outlook.

978-1-5386-3344-1/17/$31.00

II. Related work

A. Self-aware computing systems

Over the last decade researchers have been proposing and investigating the construction of computing systems with so-called self-* properties. The self in self-* refers to the capability of a system to modify its own behaviour or structure without any external control in reaction to or even in anticipation of system dynamics [10]. System dynamics can be caused by changes in the system itself or by events external to the system. There are many instantiations of self-* such as self-adaptive, self-optimising, self-coordinating and self-healing, and the appeal of self-* properties has fuelled research fields such as self-organising systems [10], autonomic computing [11], [12], and organic computing [13]. More recently and in a sense on top of the evolution of self-* computing systems, we find systems attributed with the characteristics of being *self-aware* [14].

Self-organising systems remained a rather broad and not that precisely defined category in literature. According to [10], a self-organising system can change its internal structure and functionality at run-time without any explicit direction mechanism. *Autonomic computing* aims at solving the emerging complexity crisis in software engineering. Since humans are no longer able to deal with the rising complexity, dynamics, heterogeneity and uncertainty of future systems, these systems should be enabled to (autonomically) manage themselves [11]. IBM, as a main driving force behind the autonomic computing idea, proposed a reference architecture named *MAPE-k* for the autonomic manager, which executes the monitor, analyse, plan and execute (MAPE) control loop and maintains a knowledge base [15]. The architecture uses sensors to collect information about the environment and the system itself.

Organic computing [13] is a related concept that on one hand extends autonomic computing by the properties of self-organisation and self-explanation, but on the other hand does not require systems to be fully autonomous. The approach introduces an observer and a controller component on top of the adaptive system. External users provide goals to the controller and only in case an adaptation violates these goals or any other given constraints, the controller interferes. This has been denoted as controlled autonomy.

Self-awareness appeared as key attribute in both autonomic and organic computing and, subsequently, research on self-aware computing systems has been supported by DARPA [16] and the European Commission through its Future & Emerging Technology programme [17].A more complete review of this topic is provided in [18], [6], [19]. Together with a recent Schloss Dagstuhl workshop [18], this work has already resulted in two books [6], [19].

B. Camera nodes on reconfigurable platforms

Over the last decade VSNs have been deployed on a variety of hardware platforms including embedded processors, systems-on-chip, field-programmable gate arrays (FPGAs), and commercial off-the-shelf (COTS) based systems [20], [21], [22]. These resource-limited platforms enable on board processing of only low to medium complexity computer vision algorithms such as frame differencing, background modelling, and feature-based object classification [3]. Thus, the functionality provided by VSN nodes has mostly been restricted to image enhancement, compression, motion detection, object detection and tracking. Only recently, VSN nodes including more advanced control, security and privacy-protection functionality [23], [24], [25] have been proposed.

The implementation of applications on distributed (embedded) systems requires a substantial software infrastructure, especially when scalability, flexibility and portability are important design objectives. In the context of VSNs, the software infrastructure mainly concerns operating systems (OSs) and middleware systems. VSN nodes predominantly run standard operating systems such as (embedded) Linux since they provide numerous features and modules like multi-threading and network communication and support various third party libraries (e.g., OpenCV) to ease software development. Middleware systems support the distribution of data and control at the network level [26], however, there is no clear trend towards a specific (class of) middleware systems for VSN. Mostly ad-hoc approaches and proprietary systems are used to realise basic data distribution and networking functionality.

Support for adaptation is currently provided in a rather limited sense for VSN applications. If at all, adaptation capabilities focus only on individual components such as routing [27], service composition [28], or application aspects [29]. Although FPGAs have been used for implementing VSN nodes, only a few (wireless sensor) platforms support dynamic hardware reconfiguration to achieve adaptation, e.g., [30], [31].

III. Self-aware node architecture

A. Computational self-awareness

Self-awareness denotes a paradigm for systems and applications that pro-actively gather information, maintain knowledge about their own internal states and environments and then use this knowledge to reason about behaviours. In our previous work, we have transferred these concepts to the computing domain and proposed the notions of *computational self-awareness* and *self-expression* [6], and a corresponding reference architecture [9].

In their notion, self-awareness describes methods and models for a system to gain information about its own state (private self-awareness) using internal sensors, and about the environment (public self-awareness) using external sensors. Self-expression subsumes reasoning and decision making techniques to adapt to changes in the system state or environment through internal and external actuators. The notion of computational self-awareness can provide computing systems with advanced levels of autonomous behaviour to enable runtime self-adaptation and management of complex trade-offs in rapidly changing conditions. Furthermore, collective self-awareness has been defined as the self-awareness property of a collective system composed of many interacting subsystems.

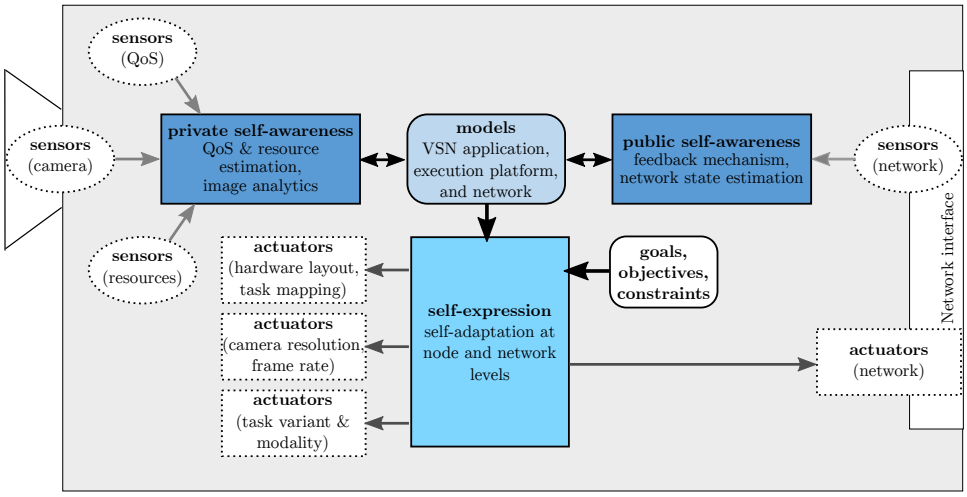

Fig. 1. Reference architecture of self-aware smart camera node.

Finally, Lewis et al. have introduced levels of self-awareness to characterise different capabilities of self-aware computing systems, from rather simple stimulus-aware systems to more advanced goal and meta-self-aware systems. Higher levels of self-awareness will become critical in dealing with the complexities of future computing systems in general and VSNs in particular [9].

B. Reference architecture

Figure 1 depicts the reference architecture of a self-aware smart camera node. This architecture follows the principles of computational self-awareness [9] and is composed by components for private self-awareness, public self-awareness and self-expression. These modules maintain and use various models representing knowledge about the node, the network and the environment.

The private self-awareness module abstracts data from different internal sensors in order to maintain models about the current QoS level of the application and the resource utilisation of the node platform. The camera represents the most important sensor and provides input for the image analysis which is also part of the private self-awareness. The public self-awareness module receives information from other camera nodes and uses this information to update the models from the network-level perspective. Instead of relying on predefined knowledge and rules, the self-awareness modules utilise online learning to maintain the models for the application, the platform and the network.

The self-expression module performs self-adaptation of the hardware platform as well as the application based on these models. Self-adaptation is realised by means of actuators which are able to dynamical modify the hardware layout and task mapping, the camera resolution and frame rate as well as the application's task variants and modalities. The self-expression module can interact with other camera nodes via the network interface and trigger adaptation at the network level. Note that the modules of the reference architecture represent a generic algorithmic framework which can be implemented by various (learning) algorithms and modelling approaches. We introduce some concrete instantiations of these modules in Section III-D.

C. Node platform and VSN application

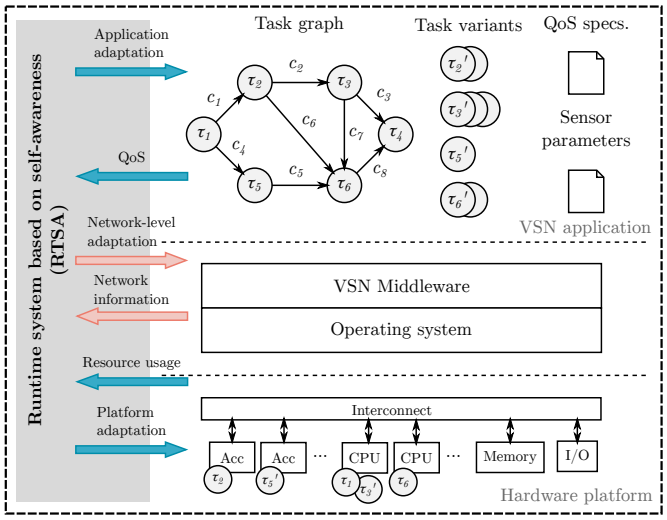

Fig. 2. VSN node platform.

A camera node is implemented on an embedded computing platform such as Xilinx Zynq or UltraScale+ and structured into the layers (i) hardware platform, (ii) operating system, (iii) VSN middleware, and (iv) VSN application (cp. Figure 2). For the operating system and VSN middleware layers we draw on our previous research: To implement the operating system layer we employ and adapt ReconOS/Linux, our previous operating system for FPGA-based CPU/accelerator architectures. ReconOS [32] extends multi-threaded programming to reconfigurable hardware and allows for starting and stopping hardware tasks at runtime as well as migrating tasks across the

hardware/software border through dynamic partial reconfiguration. As VSN middleware for communication within the VSN network we utilise Ella [33] which is realised as distributed publish/subscribe middleware and supports runtime adaptation.

The VSN application layer consists of executable code plus meta-data that facilitates an explicit modelling and reasoning about the application. The meta-data includes a description of the application in form of a task graph. Tasks can come in different variants, which constitute different algorithms for solving a task, and modalities, which means either a software or a hardware implementation. Task variants and modalities differ in functional quality and non-functional parameters, e.g., runtime, memory, reconfigurable hardware area, and energy. The meta-data further includes a specification of the applications expected quality of service (QoS) which can cover functional specifications and non-functional specifications.

Finally, the meta-data specifies possible ranges for sensor parameters, such as the cameras resolution and frame rate. The hardware platform layer comprises a platform FPGA, memory, and peripherals that connect to a camera and a network interface. The platform FPGA implements a multi-core system with several CPU cores, reconfigurable logic for implementing hardware accelerator cores, internal memory and I/O cores. Due to the FPGAs reconfigurability, not only the mapping between tasks and cores but also the layout of the multi-core system, i.e., the number of CPU cores and the number and sizes of reconfigurable accelerator cores, can be changed on demand.

A VSN node's configuration specifies both application specific and resource specific bindings. At the application layer, a configuration denotes a concrete set of tasks, i.e., one variant/modality for each task, with selected sensor parameter settings. At the resource layer, the configuration denotes an instance of the hybrid multi-core with a number of CPUs and hardware accelerator cores, and a mapping of tasks to cores. An *adaptation* is the process of changing the configuration. *Self-adaptation* is an adaptation driven from the VSN node or network itself, without an external controller that dictates when and how to change the configuration. *Self-awareness* is our concept and model for structuring functionalities required for self-adaptation in complex computing systems. The key innovation of our design approach is that we look at *interrelated adaptations at several levels*: Locally, we differentiate between adaptations of resources (hardware platform layer) and applications (VSN application layer) on the node level; globally, we differentiate between adaptations at a single node and at a set of nodes at the network level.

D. Instantiations of SA and SE modules

As previously noted, our reference architecture is composed of generic components for self-awareness (SA) and self-expression (SE). In this section we briefly summarise concrete instantiations of these components based on our previous work.

1) Multi-camera tracking: Our first example is based on object tracking in a multi-camera network [34]. In such a network each camera is capable of tracking dedicated objects within its field of view. Transferring tracking from single cameras to a network of cameras requires coordination of the tracking responsibility. A particular challenge here is to re-identify the objects in different views and to hand-over the tracking responsibility among the cameras. To coordinate the object tracking responsibilities in the camera network, a market-based handover approach was applied where the cameras treat object tracking responsibilities as goods, providing some utility over time. The cameras can decide in a self-expressive manner on their own when to "sell" tracking responsibilities to other cameras using virtual auctions. An important question for the selling camera is to whom to send the auction invitations. Without any a priori knowledge about the network topology, invitations could be broadcasted to all cameras. By following this strategy the "best" camera for taking over the tracking responsibility will receive an invitation and may participate in the bidding.

By observing the virtual market, i.e., the selling and buying of objects, each camera acquires knowledge about the network topology, since neighbouring cameras much more likely participate in the bidding for an object in the field of view. Note that the fields of view of neighbouring cameras are in close proximity or are even overlapping. Thus, the acquired topology information corresponds to the vision graph of the camera network and not the traditional network graph. We model this topology information by a weighted graph where nodes represent cameras and the weights represent the likelihood of neighbourhood [35]. A successful object handover strengthens the weight, whereas the passage of time decays the weight. Thus, each camera maintains a model of its local network topology in a self-aware manner.

This topology model is exploited in the self-expression module to adapt the trading behaviour of the cameras, i.e., to whom bidding invitations should be sent to. Based on the link weights and time for invitation we have proposed six different self-expressive strategies for handover. Obviously, the selected strategy influences the achieved tracking utility as well as communication and computational overhead. In [34], [36], online learning algorithms, specifically multi-armed bandit problem solvers, have been used within each camera to learn the appropriate strategy for each node during runtime. Dynamic strategy selection leads to another level of self-aware and self-expressive behaviour of the camera network and is able to achieve a more Pareto efficient global performance than with any static selection.

2) Multi-core assignment: The second example is a multi-core assignment problem in a hybrid system comprising a CPU core and hardware accelerator cores [37]. The multi-core system receives a workload based on two applications, sorting and matrix multiplication. Sorting tasks require the system to sort 8 kilobyte blocks of 32 bit integers at a varying rate. Matrix multiplication tasks operate on matrices of size 27×27 using Strassen's algorithm. Larger matrices of size $2n \times 2n$ with $n >= 7$ can be handled by performing $7n - 7$ multiplications of matrices of size 27×27. The rate of sorting

tasks is varied to mimic a fractal workload as observed in, for example, the networking domain, and the workload for matrix multiplications is assumed to be infinite. Both sorting and matrix multiplication tasks are enqueued in FIFOs and from there assigned to cores.

The applications have been implemented on a ReconOS system running on a Virtex 6 FPGA with one CPU core and 12 partially reconfigurable regions for hardware accelerators (cmp. Section IV). A reconfigurable region may contain either one sorting thread or one matrix multiplication thread at a time. The multi-core assignment problem is to decide how many sorting and matrix multiplication accelerators to configure and execute in each time point, depending on the workload, the system state, and the assignment objective. The concept of self-aware computing has been employed and demonstrated on several objectives. For example, in one scenario the number of matrix multiplications was maximized under the constraint that all sorting tasks must be executed, i.e., the FIFO storing sorting tasks must not overflow. Another experiment has shown how to operate under conflicting constraints such as execute all sorting tasks and keep the chip temperature under a given limit.

The self-awareness modules gather information about the system state, including the utilization of resources, current performance of the accelerators, FIFO fill levels, and the chip temperature. For deriving the assignment decisions, rule-based self-expression strategies formulated as rules proved to be very effective. Such rules can use all parameters provided by the self-awareness modules, e.g., derivations of FIFO fill levels, and even be hierarchically structured. In [37], it was also shown how to combine several self-expression strategies to so-called meta-strategies, where a rule or rule set, respectively, determines the actual rule set to use.

IV. Node implementation

To realize visual sensor network nodes we leverage modern platform FPGA technology since this technology provides the required flexibility and adaptivity not only in software but also in hardware. In this section we first describe ReconOS, the operating system layer we employ for platform FPGAs, and the levels of adaptivity enabled by ReconOS, and then report on our current VSN node prototype implementation.

A. Adaptive Hardware Platform

ReconOS is an operating system layer for hybrid multi-cores comprising at least one CPU core and a number of hardware accelerators [32]. The hardware accelerators are turned into so-called hardware threads, which enable extending the multi-threaded programming model from software to hardware. By defining a standardised interface for hardware threads, ReconOS allows them to transparently communicate and synchronise with software threads and the host operating system kernel running on the system CPU. Figure 3 shows an exemplary ReconOS architecture with a CPU core and two hardware threads. Hardware threads are loaded into reconfigurable slots, which are rectangular areas on the logic fabric. Each hardware slot is provided with an operating system

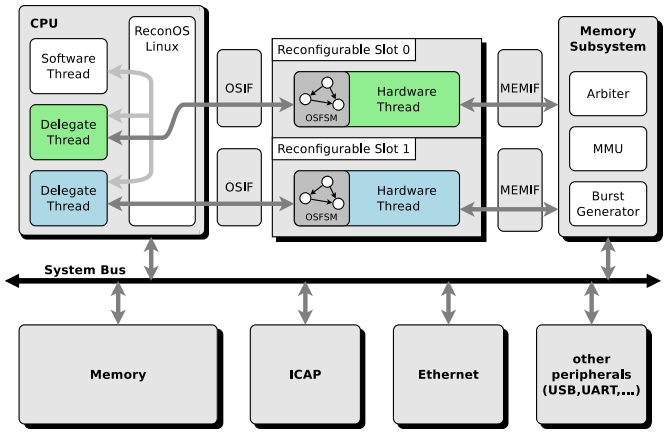

Fig. 3. ReconOS architecture with one CPU core and two hardware threads loaded into reconfigurable slots (from [32]).

interface (OSIF) and a memory interface (MEMIF). The OSIF establishes the communication between the hardware thread and the operating system kernel via a corresponding delegate thread. To that end, each hardware thread includes an operating system finite state machine (OSFSM) that expresses all operating system interactions of the hardware thread. These interactions are naturally sequential, while the remaining part of the hardware thread typically utilises the parallelism offered by the reconfigurable fabric. Another key aspect of ReconOS are the delegate threads. Delegate threads are light-weight software threads that interact with the operating system kernel on behalf of hardware threads. The delegate thread approach facilitates porting of ReconOS to different host operating systems. The hardware thread's memory accesses are routed via the MEMIF to the memory subsystem. The memory subsystem includes a memory management unit that allows hardware threads to use virtual addresses, which is necessary in case the host operating system runs on virtual memory.

A platform FPGA operated under ReconOS supports two levels of runtime adaptation. At the first level, we can start and stop not only software threads but also hardware threads using dynamic partial reconfiguration. Hardware threads can be dynamically loaded into reconfigurable slots. ReconOS even supports time-multiplexing of reconfigurable slots through cooperative multi-tasking [38]. The operating system kernel signals a request for preemption to a hardware thread. The hardware thread accepts this request by yielding only when it wants to get preempted, which typically is at points in time where the context is small and can be easily saved by the thread itself. In case the hardware thread does not yield it continues execution without any delay. This feature not only enables improved resource usage of reconfigurable slots but also migration of threads across the hardware/software border.

The second level of adaptation addresses the layout of the platform FPGA, i.e., the partitioning of the available area into reconfigurable slots. ReconOS defines reconfigurable slots of minimal size, so-called micro slots, which on Zynq devices comprise 600 slices (2400 LUTs, 4800 registers, 180 kB

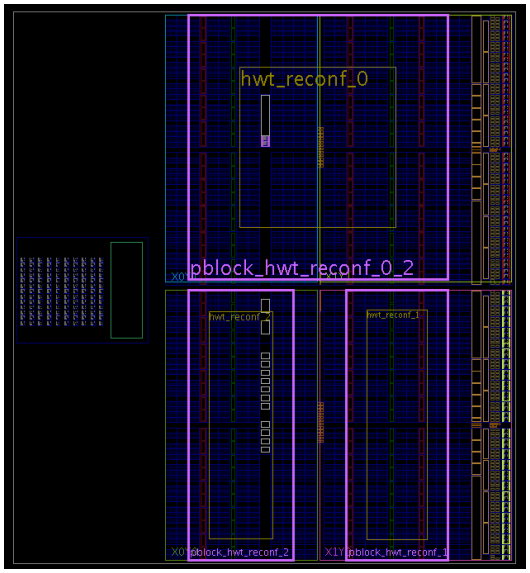

Fig. 4. Exemplary ReconOS layout on a Xilinx Zynq 7010 SoC with three reconfigurable slots.

BRAM, and 20 DSP blocks). Such a minimally sized micro slot can be reconfigured in $30 - 50$ ms. Larger slots can be created by combining abutting micro slots to larger rectangular regions. Figure 4 displays a layout for a ReconOS architecture on a Xilinx Zynq 7010 SoC. The black area in the left of Figure 4 represents the CPU subsystem of the chip. The reconfigurable area of the Zynq 7010 is organised in four micro slots, from which the layout in Figure 4 forms three slots. ReconOS can thus configure and run three hardware threads in parallel. However, the maximum size of a hardware thread is limited which prevents loading and executing larger hardware accelerators. This limitation can be overcome by selecting a different layout, for example one with one reconfigurable slot of maximum size comprising all four micro slots.

The resulting two levels of adaptation allow for exploiting trade-offs in performance, energy and hardware resource usage at runtime, and will be key to optimizing VSN applications. For example, for given data-level parallel VSN application we can increase the performance by executing several hardware threads in parallel. Should we be interested in low power consumption we could employ only one hardware thread and reduce its clock rate. When a new VSN application is received and configured during runtime that also includes larger hardware threads, we might need to adapt the layout.

The corresponding algorithmic methods that drive these adaptations are part of the system's self-expression modules, as shown in Figure 1. These modules have to decide on the assignment of tasks to cores and slots, on the operation frequency of each core and the selection of the layout. The specific challenges for developing methods to solve the corresponding optimisation problem are that the two levels of adaptations are not independent, and that we have to strive for efficient methods since they are executed at runtime. The self-awareness modules measure current resource utilisation figures which helps derive adaptation decisions. In more sophisticated versions, the self-awareness modules learn from task executions and use this knowledge for future mapping decisions.

B. VSN Node Prototype

Figure 5 depicts the hardware architecture and a prototype implementation of our current VSN node. The hardware of the camera consists of a Xilinx Zynq 7010 SoC and an OmniVision OV5642 CMOS image sensor. The Zynq SoC comprises a hard dual-core ARM Cortex A9 processing system (PS) clocked at 666 MHz and a programmable logic (PL) FPGA fabric. An Internal Configuration Access Port (ICAP) provided by the Zynq SoC enables reconfiguration of the programmable logic. The ICAP is used for the initial configuration of the PL during boot and to partially reconfigure the PL at run-time.

The OV5642 image sensor provides image data up to 1920×1080 pixel resolution in YUV422 (colour space) format. An 8-bit parallel low-voltage differential signalling (LVDS) interface is connecting the sensor to the programmable logic (PL) of the Zynq SoC. Additionally, a serial camera control bus (SCCB) implemented on an I2C interface connected to the processing system of the SoC is used to configure the sensor. We implemented a custom IP core to translate the signals from the LVDS interface of the OV5642 to the AMBA AXI4-Stream interconnect. The AMBA AXI4-Stream interface is a 64-bit interface with handshaking data flow and is well suited for applications focussing on the data-centric and data-flow paradigm. Transfer video stream data between memory and hardware is done using a video direct memory access (VDMA) IP core. The VDMA is programmed from the processing system (PS) and allows memory-mapped access to the image data. Using the VDMA also allows the implementation of a frame-buffered application on the processing system.

The programmable logic also incorporates a video processing component. The video processing component is a reconfigurable hardware accelerator for image processing implemented in Xilinx Vivado High-Level-Synthesis (HLS). HLS enables fast design and implementation of hardware accelerated filters using high-level C/C++ and System C specifications. We realised five different filters to evaluate the device utilisation of different implementations.

The system design is split into a static and a reconfigurable partition. The reconfigurable partition of the system design refers to the region of the PL that is used for reconfiguration of the hardware accelerators and has to be large enough to provide resources for the configuration with the highest resource utilisation. Table I presents the resource utilisation of different image filters. All filters are able to process YUV422 images. The resolution of all image filters is adjustable up to a maximum resolution of 1920×1080 pixels. If turned into ReconOS hardware threads each image filter will fit into one micro slot, except the sobel (3x3) filter that takes two micro slots. Naturally, some of the resources provided by a reconfigurable slot will be unused. This effect is known as internal fragmentation and can be minimised by defining smaller micro

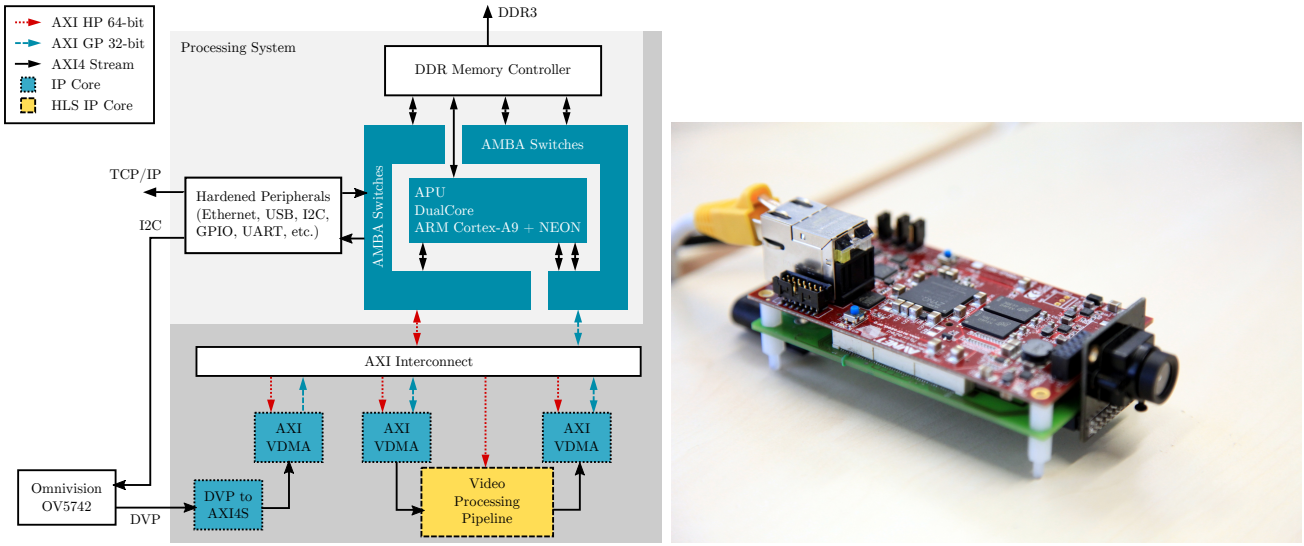

Fig. 5. Hardware architecture (left) and prototype implementation (right) of the VSN node.

Filter	LUTs	Registers	BRAM	DSPs
pass-through	496	396	-	-
simple-threshold	594	447	-	-
simple-median	693	523	-	-
yuv-to-grayscale	744	539	-	-
sobel (3x3)	3288	2219	$54kB$	17

TABLE I
RESOURCE UTILIZATION OF RECONFIGURABLE IMAGE FILTERS.

Unit	LUTs	Registers	BRAM
Sensor Input	2758	3642	$36kB$
Video Processing Core	4606	4956	$27kB$
Processing System Instantiation	371	40	-
Total	7735	8638	$63kB$

TABLE II
RESOURCE UTILIZATION OF STATIC SYSTEM DESIGN.

slots. Importantly, to avoid placement restrictions for hardware threads ReconOS defines micro slots such that they exhibit the same resources.

The static partition includes the input logic for the image sensor, the instantiation and reset logic for the processing system and interface logic that is necessary to transfer data to the reconfigurable image filter. Table II shows the resource utilization of the static system design.

The software stack of the camera consists of a boot loader and a Linux kernel. At the operating system level access to the ICAP and the VDMA is provided by the *xdevcfg* and *xvdma* drivers provided by Xilinx. The *xdevcfg* drivers enable to reconfigure the programmable logic completely or partially from an application in the user-space of the operating system. The *xvdma* driver can trigger the DMA transfer between image sensor, memory and image processing components and enables a non-blocking transfer of data by providing interrupt signalling to the user-space application.

V. SUMMARY AND OUTLOOK

With this paper, we propose computational self-awareness as fundamental concept for designing and operating smart cameras and visual sensor networks. After reviewing important work in self-aware computing systems and camera node platforms, we have elaborated on this concept and presented a reference architecture for a self-adaptive smart camera node based on a reconfigurable system-on-chip platform. The platform runs the ReconOS operating system which provides the multi-threaded programming model for software and hardware as well as two levels of resource adaptivity. Additionally, we have also presented a prototypical implementation of a reconfigurable VSN node.

In future work, we will make available ReconOS on the VSN node prototype to be able to design flexible VSN applications with improved resource efficiency. Setting up a visual sensor network composed of multiple nodes will allow us to quantitatively evaluate the run-time performance of various image processing tasks implemented in either software or hardware as well as the overhead introduced by the different levels of reconfiguration. Based on measurements carried out on the implemented VSN, we will devise algorithms for the self-awareness and self-expression modules covering both the VSN application level and the resource levels. The interrelation of the adaptations at the different levels poses a major and novel challenge, which we hope to successfully address with the computational self-awareness approach.

REFERENCES

[1] B. Rinner and W. Wolf, "Introduction to Distributed Smart Cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1565–1575, 2008.
[2] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach, "Distributed Embedded Smart Cameras for Surveillance Applications," *IEEE Computer*, vol. 39, no. 2, pp. 68–75, 2006.
[3] S. Soro and W. Heinzelman, "A Survey of Visual Sensor Networks," *Advances in Multimedia*, pp. 1–21, 2009.

[4] M. Bramberger, B. Rinner, and H. Schwabach, "An embedded smart camera on a scalable heterogeneous multi-DSP system," in *Proceedings of the IEEE European DSP Education and Research Symposium (EDERS 2004)*, 2004.

[5] B. Rinner and W. Wolf, *Multi-Camera Networks*. Elsevier, 2009, ch. Toward Pervasive Smart Camera Networks, pp. 483–496.

[6] P. R. Lewis, M. Platzner, B. Rinner, J. Torresen, and X. Yao, Eds., *Self-aware Computing Systems: An Engineering Approach*. Springer, 2016.

[7] A. Morin, "Levels of consciousness and self-awareness: A comparison and integration of various neurocognitive views," *Cons. and Cog.*, vol. 15, no. 2, pp. 358–71, 2006.

[8] U. Neisser, "The roots of self-knowledge: Perceiving self, it, and thou," *Annals of the NY AoS.*, vol. 818, pp. 19–33, 1997.

[9] P. R. Lewis, A. Chandra, F. Faniyi, K. Glette, T. Chen, R. Bahsoon, J. Torresen, and X. Yao, "Architectural Aspects of Self-Aware and Self-Expressive Computing Systems: From Psychology to Engineering," *Computer*, vol. 48, no. 8, pp. 62–70, August 2015.

[10] G. M. Serugendo, M.-P. Gleizes, and A. Karageorgos, *Self-Organizing Software - From Natural to Artificial Adaptation*. Springer, 2011.

[11] R. Sterritt and M. Hinchey, "SPAACE IV: Self-Properties for an Autonomous and Autonomic Computing Environment - Part IV A Newish Hope," in *IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems (EASe)*, 2010, pp. 119–125.

[12] M. Wirsing, M. Hölzl, N. Koch, and P. Mayer, Eds., *Software Engineering for Collective Autonomic Systems: The ASCENS Approach*. Springer, 2015.

[13] C. Müller-Schloer, H. Schmeck, and T. Ungerer, *Organic computing: a paradigm shift for complex systems*. Springer, 2011.

[14] A. Agarwal, J. Miller, J. Eastep, D. Wentziaff, and H. Kasture, "Self-aware computing," MIT, Tech. Rep. AFRL-RI-RS-TR-2009-161, 2009.

[15] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[16] L. Paulson, "DARPA Creating Self-aware Computing," *IEEE Computer*, vol. 36, no. 3, p. 24, march 2003.

[17] European Commission, "Self-Awareness in Autonomic Systems," Jan 2013.

[18] Schloss Dagstuhl Workshop on Model-driven Algorithms and Architectures for Self-Aware Computing Systems, "https://www.dagstuhl.de/en/program/calendar/semhp/?semnr=15041."

[19] S. Kounev, J. O. Kephart, A. Milenkowski, and X. Zhu, Eds., *Self-Aware Computing Systems*. Springer, 2017.

[20] A. N. Belbachir, Ed., *Smart Cameras*. Springer, 2010.

[21] C. Bobda and S. Velipasalar, Eds., *Distributed Embedded Smart Cameras: Architectures, Design and Applications*. Springer, 2014.

[22] M. A. Najjar, *Visual Sensor Nodes*. Springer, 2014, ch. 2, pp. 17–35.

[23] P. Natarajan, P. K. Atrey, and M. Kankanhalli, "Multi-Camera Coordination and Control in Surveillance Systems: A Survey," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 11, no. 4, p. 30, 2015.

[24] T. Winkler and B. Rinner, "Security and Privacy Protection in Visual Sensor Networks: A Survey," *ACM Computing Surveys*, vol. 47, no. 1, pp. 1–39, 2014.

[25] J. C. SanMiguel, K. Shoop, A. Cavallaro, C. Micheloni, and G. L. Foresti, "Self-Reconfigurable Smart Camera Networks," *IEEE Computer*, vol. 47, no. 5, pp. 67–73, 2014.

[26] B. Rinner and M. Quaritsch, *Multi-Camera Networks*. Elsevier, 2009, ch. Embedded Middleware for Smart Camera Networks and Sensor Fusion, pp. 511–537.

[27] T. Bourdenas, D. Wood, P. Zerfos, F. Bergamaschi, and M. Sloman, "Self-adaptive routing in multi-hop sensor networks," in *Proceedings of the 7th International Conference on Network and Service Management (CNSM)*, 2011, pp. 1–9.

[28] L. Zhang, M. Ma, G. Zhang, and A. S. Lim, "Distributed Composition Services for Self-adaptation Wireless Sensor Networks ," in *Proceedings of the Computing, Communications and IT Applications Conference (ComComAp)*, 2013, pp. 47–52.

[29] V. P. Munishwar, S. Tilak, and N. B. Abu-Ghazaleh, "Coverage Management for Mobile Targets in Visual Sensor Networks," in *Proceedings of the 15th ACM international Conference on Modeling, Analysis and Simulation of wireless and mobile Systems*, 2012, pp. 107–116.

[30] Y. E. Krasteva, J. Portilla, E. de la Torre, and T. Riesgo, "Embedded Runtime Reconfigurable Nodes for Wireless Sensor Networks Applications," *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1800–1810, 2011.

[31] J. A. Guevara, E. A. Vargas, A. F. Fatecha, and F. Barrero, "Dynamically Reconfigurable WSN Node Based on ISO/IEC/IEEE 21451 TEDS," *IEEE Sensors Journal*, vol. 15, no. 5, pp. 2567–2576, 2015.

[32] A. Agne, M. Happe, A. Keller, E. Lübbers, B. Plattner, M. Platzner, and C. Plessl, "ReconOS — an Operating System Approach for Reconfigurable Computers," *IEEE Micro*, no. Jan./Feb., pp. 60–71, 2014.

[33] B. Dieber, J. Simonjan, L. Esterle, B. Rinner, G. Nebehay, R. Pflugfelder, and G. J. Fernandez, "Ella: Middleware for Multi-camera Surveillance in Heterogeneous Visual Sensor Networks," in *Proceedings of the seventh ACM/IEEE International Conference on Distributed Smart Cameras*, Palm Springs, USA, 2013, pp. 1–6.

[34] B. Rinner, L. Esterle, J. Simonjan, G. Nebehay, R. Pflugfelder, G. Fernandez Dominguez, and P. R. Lewis, "Self-Aware and Self-Expressive Camera Networks," *IEEE Computer*, vol. 48, no. 7, pp. 21–28, jul 2015.

[35] L. Esterle, P. R. Lewis, X. Yao, and B. Rinner, "Socio-Economic Vision Graph Generation and Handover in Distributed Smart Camera Networks," *ACM Transactions on Sensor Networks*, vol. 10, no. 2, pp. 1–24, January 2014.

[36] P. R. Lewis, L. Esterle, A. Chandra, B. Rinner, J. Torresen, and X. Yao, "Socio-Economic Vision Graph Generation and Handover in Distributed Smart Camera Networks," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 10, no. 2, pp. 1–30, June 2015.

[37] A. Agne, M. Happe, A. Lösch, C. Plessl, and M. Platzner, "Self-awareness as a model for designing and operating heterogeneous multicores," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 7, no. 2, 2014.

[38] E. Lübbers and M. Platzner, "Cooperative Multithreading in Dynamically Reconfigurable Systems," in *IEEE International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2009, pp. 1–4.