A Novel Software Framework for Power-Aware Reconfiguration in Distributed Embedded Smart Cameras

Andreas Doblander, Arnold Maier, Bernhard Rinner
Institute for Technical Informatics
Graz University of Technology, AUSTRIA
{doblander, maier, rinner}@iti.tugraz.at

Helmut Schwabach
Video and Safety Technology
ARC Seibersdorf research, AUSTRIA
{helmut.schwabach}@arcs.ac.at

Abstract

Intelligent video surveillance (IVS) systems are based on so called smart cameras that combine video sensing, processing and communication within a single embedded device. Maximizing the service-level and minimizing the power consumption are two important but conflicting objectives in IVS. In this paper we present a novel software framework for power-aware reconfiguration in distributed intelligent video surveillance (IVS) systems. The multi-layer heterogenous software framework hosts services for power-aware system-level task distribution and is based on a publisher-subscriber middleware approach. An online multi-criterion optimizer permanently computes optimal camera configurations with respect to a given cost model for both power consumption and service-levels.

1. Introduction

Intelligent video surveillance (IVS) is based on the recent development of so called smart cameras and is getting more and more attention in industry and research. Smart cameras [15, 2] combine video sensing, processing and communication within a single embedded device. Networks of distributed smart cameras are an emerging technology for a broad range of important applications. Through cooperation among individual cameras these networks have the potential to realize many more challenging applications than traditional systems. An IVS system may be partitioned into logical groups of typically co-located smart cameras—so-called surveillance clusters.

Typically, smart cameras have to execute demanding video processing and compression algorithms [6]. Furthermore, power-awareness is also of major importance in IVS. Especially in solar-powered cameras it leads to prolonged operation time and smaller device sizes. Recent technologies such as Power-over-Ethernet are also deployed in embedded designs but do have strict limitations in the amount of available energy. Thus, maximizing the service-level and minimizing the power consumption are two important but conflicting design objectives for IVS systems. Since service-level and power requirements change during operation online optimization for both objectives followed by dynamic reconfiguration is desirable. The problem of finding optimal camera configurations can be formulated as a combinatorial multi-criterion optimization (MCO) problem with the two above mentioned objectives.

A camera configuration consists of various IVS services, i.e., video analysis algorithms, in different QoS-levels on given hardware resources. Some surveillance services do not have to be assigned to a specific camera but they may be relocated within a surveillance cluster. Flexibility of algorithm configurations, i.e., how tasks are composed to build the application, as well as scalability concerning the number and the different types of employed surveillance tasks have to be addressed by the software framework.

The *SmartCam* [2] is a fully embedded system and serves as the hardware platform for this work. It is realized as a scalable, embedded multi-processor platform consisting of a network processor and a variable number of digital signal processors (DSPs) and is especially targeted for a use

Proceedings of the 12th International Conference on Parallel and Distributed Systems (ICPADS'06)
0-7695-2612-8/06 $20.00 © 2006 IEEE

in distributed IVS setups. A multilayer heterogeneous software framework has been implemented that hosts services for power-aware system-level task distribution. It is based on a publisher-subscriber middleware approach that allows dynamic reconfiguration of services, i.e., the algorithms executed on the DSPs. This is in contrast to current practice where functionality is mostly changed by re-programming a device with a new software image. Furthermore, it contains an online multi-criterion optimizer that permanently computes feasible camera configurations with respect to a given cost model for both power consumption and service-levels.

2. Related work

Middleware for distributed and embedded systems is a very active research field. A lot of work has been done to support transparent communication and to ease distributed application development. Unfortunately, middleware technologies from general purpose computing, such as, e.g., Microsoft DCOM [14], Java RMI [11] and OMG CORBA [12] are not suitable for very resource limited devices [8] as smart cameras are.

An interesting approach is the "Self-*" architecture [5]. It is a data-flow oriented and component-based middleware framework that is aimed for dependable pervasive computing systems.

The notion of *runtime configuration capable embedded systems* by Nitsch and Kebschull [10] is quite similar to our understanding of a dynamically configurable system. However, we do not consider hardware reconfiguration as suggested in their work. Nitsch and Kebschull also use Enterprise Java Beans as enabling technology which is too resource intensive for our application.

A popular inter-process communication model for realtime systems is the realtime publisher/subscriber model (RT-PS) [13]. It supports loose coupling of tasks by message-oriented communication. As the registration of data sources and sinks can be done at runtime the RT-PS approach was chosen as the basis for our software framework.

Minimizing the power consumption and maximizing service-levels in IVS are—similar to a lot of other real-world problems—two conflicting objectives for optimization and therefore are referred to as *multi-criterion optimization (MCO)* problems [3]. Solving a MCO-problem does not result in a single scalar that represents an optimal value but in a set of several so called *non-dominated* solutions (also referred to as *Pareto-optimal* solutions). However, none of the Pareto-optimal solutions is 'better' than another one in general but only in at least one criterion.

So called evolutionary approaches include genetic algorithms (GAs) that are used to solve MCO problems. GAs are heuristic search algorithms that are based on the evolutionary ideas of natural selection and genetics. They are an applicable and robust approach especially for MCO problems with a large and complex search space. All genetic algorithms are based on the same generic concept. They start by a random initial population and generate better populations in each iteration by the principles of mutation and crossover. Each individual gets tested if it fits as possible solution due to a given cost function of each objective function [9].

Each camera configuration corresponds to a certain utilization of hardware components. Therefore, the approach presented in this paper takes use of dynamic power management (DPM) in order to minimize the power consumption [1]. DPM is based on the observation that a lot of power is wasted because of system components that are fully powered up even if they are not in use.

3. A novel software framework for distributed smart cameras

3.1. Software architecture

The main focus of the *SmartCam* software architecture is flexibility and reconfigurability. It consists of several layers which can be grouped into (i) the *DSP-Framework* (DSP-FW), running on DSPs, and (ii) the *SmartCam-Framework* (SC-FW), running on a network processor. This architecture is based on the abstraction that the application logic is running on the network processor and dynamically loads and unloads the actual IVS services (i.e., algorithms) onto the DSPs as needed. Thus, the software framework fulfills the main prerequisite for enabling dynamic camera reconfiguration. An overview of the software architecture of our smart camera is depicted in Figure 1.

SmartCam Framework The SC-FW that is illustrated in the left part of Figure 1 serves two main purposes. First, it

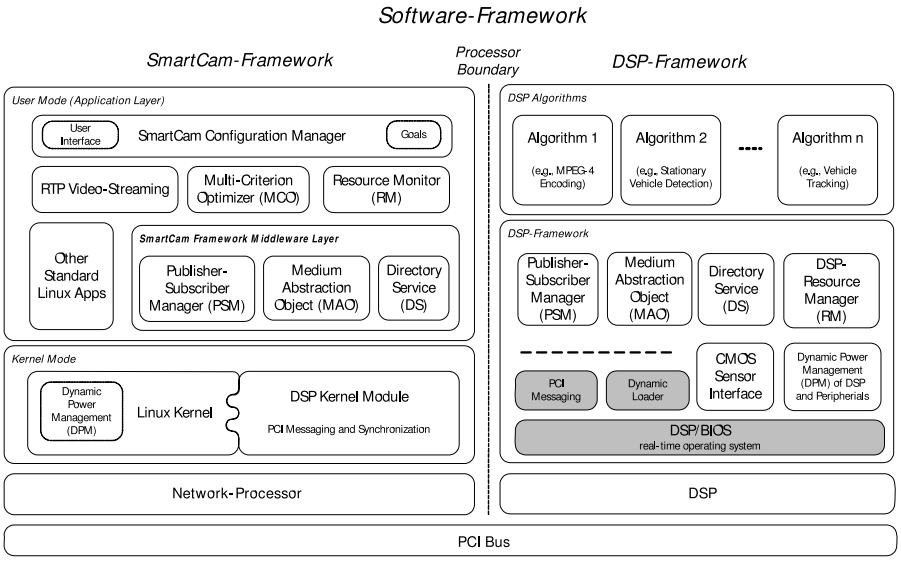

Figure 1. The overall software architecture of our smart camera.

provides an abstraction of the DSPs to ensure platform independence of the application layer. Second, the application layer uses the provided communication methods, i.e., internal messaging to the DSPs and external IP-based communication, to exchange information or offer data relay services for the DSP-FW.

Modules of this part of the software architecture support application development in that they provide high-level interfaces to DSP algorithms and functions of the DSP-FW. To further ease application development the SC-FW on the network processor is running on top of a standard LINUX kernel.

DSP Framework The DSP-FW, as indicated in the right part of Figure 1, runs on every DSP in the system. It is built upon the DSP/BIOS realtime operating system (RTOS) from Texas Instruments. The main purposes of the DSP-Framework are (i) the abstraction of the hardware and communication channels, (ii) the support for dynamic loading and unloading of application tasks, and (iii) the management of on-chip and off-chip resources of the DSP. Of course, the sensor interface module is only needed on the DSP to which the image sensor is connected. The key functionality in the DSP-Framework is the publisher-subscriber middleware that is described in Section 3.3 in more detail.

All IVS services (i.e., video analysis algorithms) and also some framework components can be loaded and unloaded at runtime by the *Dynamic Loader* module. Actually, only modules of the DSP-FW in dark shade in Figure 1 have to be available at startup. All other components can be dynamically loaded at runtime.

3.2. Online optimization and power-aware reconfiguration of camera configurations

Regardless if a camera configuration is set directly via a user interface, by a scheduled profile or even autonomously by the IVS system itself, it is desirable that only configurations with an optimal power- and service-level tradeoff are selected. The SC-FW, therefore, also includes an implementation of a multi-criterion optimizer for online optimization of the camera configuration. It employs a specially tailored online genetic algorithm that is suitable for solving the considered MCO problem with respect to a given cost model under soft realtime demands. The model contains individual costs for both quality-of-service (QoS) and its corresponding power-consumption for each service executed on a smart camera.

The MCO delivers Pareto-optimal feasible camera configurations that represent a full set of different tradeoffs in power consumption and service level. Thus, it allows the configuration manager to choose only optimal configurations among a pre-computed set of camera configurations.

For instance, whenever possible multiple services are run on a single DSP instead of utilizing two (or more) DSPs in order to reduce power consumption. The MCO, however, gets re-triggered whenever a (new) service is to be started or stopped or the capacity of available hardware resources (i.e., DSPs or network bandwidth) is changed.

The *SmartCam* configuration manager sets an onboard camera configuration due to configurable goals. It starts or stops IVS services on the DSPs but also triggers dynamic power management. Like this the DSPs get dynamically powered on and off due to their utilization in a camera configuration.

In case of low-energy, for instance, a camera configuration that utilizes less hardware resources is desired in order to gain more power savings by onboard DPM. Within the domain of the network processor DPM may only be partially applied because of the permanent use of affected components. An exception is the implementation of a suspend-mode of the entire camera.

Within the domain of the DSPs, more possibilities exist for DPM. Due to dynamic loading and unloading of application tasks on the DSPs, a DSP may get powered down completely by the *SmartCam* configuration manager if the camera is in an appropriate configuration. Furthermore, online DPM of the DSP cores and peripheral components such as video decoders or memories is applied. The DSP/BIOS RTOS provides so called 'hooks' that are called upon specific events such as task switches. The local power managers of the individual DSPs are called from such hooks at every task switch. These power managers maintain a data structure for each individual task, containing data about all corresponding components including the DSP core and peripheral components (e.g. video decoders) that are used by the task [7].

3.3. Publisher-subscriber middleware

The publisher-subscriber architecture is an integral part of the DSP-FW [4]. It aims at providing seamless and flexible connections between the algorithms running on the DSPs. Furthermore, it has to provide the basic means for supporting application reconfigurations aimed at reducing power consumption.

From the framework's point of view every video analysis algorithm is a separate entity that is executed in its own thread. Interconnections of the algorithms are defined by the application. For communication between algorithms on the same DSP an operating system mechanism called mailbox is employed. Mailboxes provide buffered communication and also allow for synchronization as tasks are blocked when they are waiting for data delivered by the mailbox. In video applications a large amount of data has to be handled. To use the limited memory of the DSPs efficiently image data is not copied when sent between algorithms on the same DSP. Only references to actual data are exchanged. Small messages like system commands or monitored performance information are directly posted to mailboxes. Figure 2 depicts the situation for two algorithms residing on the same DSP. The first algorithm provides a data service X that the second uses for further processing.

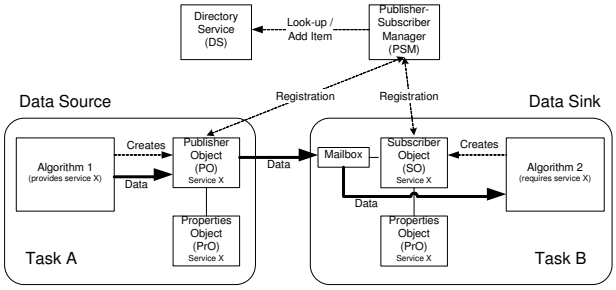

Figure 2. The publisher-subscriber architecture within a single DSP.

The *publisher-subscriber manager* (PSM) is the authority where algorithms can register as data providers or data consumers. That is, they register a publication or a subscription, respectively. There is one PSM running on each DSP and on the network processor. When an algorithm wants to register a service it first instantiates a publisher or subscriber object depending on whether a publication or subscription is needed. This object then registers itself with the PSM. The newly registered service is also added to the directory service (DS) where it can be looked up later on. As algorithms can reside on different DSPs within a *SmartCam* it is also necessary that each PSM can discover services that have registered with a different PSM. Therefore, the network processor also hosts a PSM that relays service requests between PSMs on different DSPs.

Properties objects (PrO) are used to describe published data and subscriptions, respectively. Each publisher and

subscriber object owns a PrO that represents its QoS configuration. Examples for properties include image resolution and frame rate. In the service discovery process the PrOs are used to match subscribers to appropriate publishers by comparing their properties.

A *publisher object* (PO) is instantiated for each message type an algorithm wants to publish to other tasks. On instantiation the PO handles the registration with the PSM. Every publisher keeps a PrO that contains a description of the provided service. When data is ready for transmission from the algorithm the PO posts this data as a message to the mailboxes of all subscribers registered for the service. If there are subscribers residing on different DSPs a proxy mechanism is used. This procedure is described in more detail in Section 3.3.

Tasks that require a data service of another algorithm instantiate a *subscriber object* (SO). This SO then registers with the PSM. In order to receive data a mailbox is created using operating system services. To define the required data quality also every SO owns a PrO. In the registration process the PSM looks up the appropriate service using the directory service DS (cf. Section 3.3). If a fitting service is found then the discovered publisher stores a reference to the mailbox of the requesting SO so that messages can be sent to it.

Medium abstraction and remote subscription In case of algorithms residing on different DSPs, i.e., a so-called *remote subscription*, an extension to the plain architecture described above is needed. A special object for abstracting from the communication medium is used to establish the connection. This *medium abstraction object* (MAO) is part of the middleware layer and is present on every processor of the platform. In general it is possible to use it for different communication media. But currently it is only used for providing abstract communication over the local PCI bus of the *SmartCam*. Figure 3 illustrates the case of two algorithms residing on two different DSPs in more detail.

It can be seen from Figure 3 that in contrast to the single DSP case (cf. Figure 2) each MAO creates a local proxy SO or PO on the sending and receiving DSP, respectively. These proxy objects behave like normal POs and SOs. The MAO, however, transfers the actual data over the local PCI bus instead of passing references.

Directory service and service discovery All publishings and subsciptions are listed in the directory service (DS) together with their properties. A search algorithm based on service names and QoS parameters is used to discover matching data service. If there is no matching PO or SO for a registering SO or PO, respectively, then a remote service discovery process is initiated by the local PSM. In a remote lookup the local PSM queries the PSM residing on the network processor that in turn keeps records of PSMs of all other DSPs. Therefore, all available services in the system are taken into account as all PSMs use their associated directory services in the search.

4. Implementation and experimental results

The *SmartCam* platform is based on an Intel IXDP425 development board comprising an Intel IXP425 XScale network processor running at 533 MHz. It is equipped with 16 MB of flash memory and 256 MB of SDRAM. Two to four ATEME NVDK PCI boards each comprising a Texas Instruments TMS320C6415 DSP running at up to 1 GHz are plugged into the base board. Each NVDK is equipped with 264 MB of SDRAM. The XScale is operated by a LINUX kernel version 2.6.10 and the DSPs run the Texas Instruments DSP/BIOS RTOS kernel as provided with the Code Composer Studio 3.0 development environment.

4.1. Performance analysis of the publisher-subscriber middleware

The overall memory footprint of the PS-MW is only 15.78 KB. Total memory consumption overhead, of course, depends on the number of publishings and subscriptions in the system as each of them requires a PrO and a PO or SO, respectively. In a typical setting with two algorithms per DSP and each algorithm providing one service and subscribes to one service this yields a total memory overhead of the middleware of 3.71 KB per DSP.

The PS-MW management overhead at system startup and for PO and SO creation/registration was measured to be about ten microseconds only. Message transfer overhead of our light-weight PS-MW is about 16.35% compared to a plain mailbox transfer. Note that in this scenario one publisher with exactly one connected subscriber on the same DSP was examined.

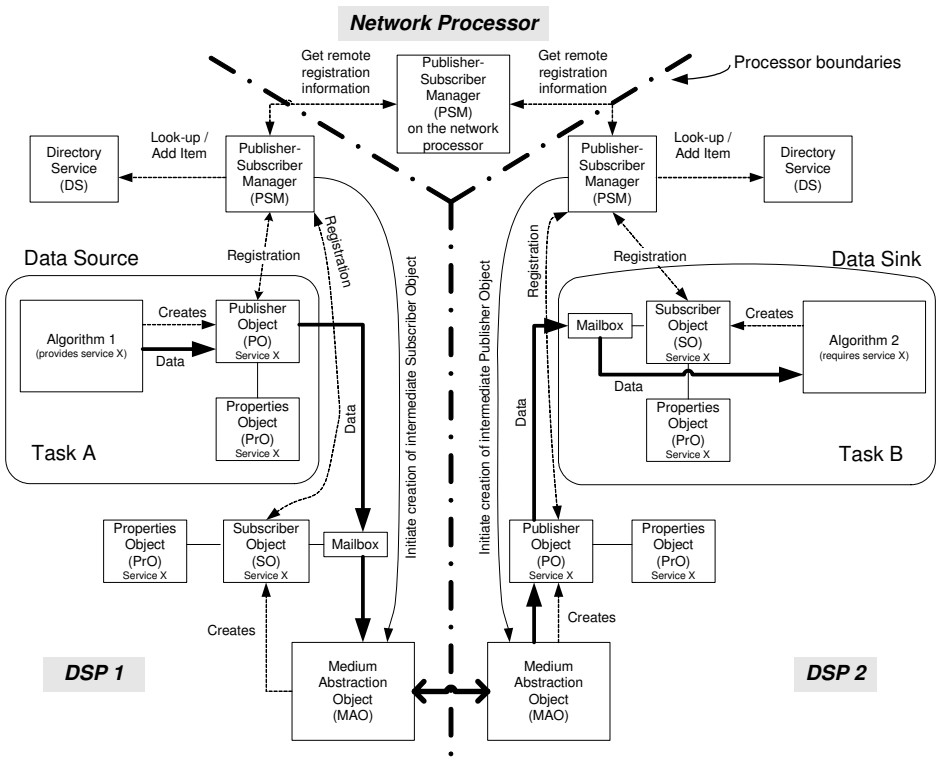

Figure 3. Extended publisher-subscriber architecture to connect algorithms on different DSPs.

In another scenario we examined a multicast communication scheme with one publisher and several subscribers connected to it on the same DSP. The significant time measure in this case is the overall time needed to transfer the published message to all subscribed tasks. In this scenario the transfer time increases almost linearly by approximately 1 μs for each subscriber.

Transfer overhead for communication of algorithms residing on different DSPs stems from the indirection in the involved MAOs and the proxy PO as well as the proxy SOs. It can be seen from Table 1 that multiple subscribers on the same remote DSP yield less overhead than if they all reside on different DSPs. This is due to less management overhead in the target MAO.

Number of SOs	Transfer overhead (μs)		
	2 DSPs	3 DSPs	4 DSPs
1	3.49	- -	- -
2	4.69	5.24	- -
3	5.91	6.44	7.49

Table 1. Message transfer overhead time for publisher and subscribers residing on different DSPs relative to direct PCI transfers.

4.2. Evaluation of the online multi-criterion optimizer

The considered MCO problem is evaluated for its use in IVS with an implementation of an genetic algorithm on the *SmartCam*. We therefore use an input data model with a maximal number of service tasks that is typical for IVS. The values for the power- and service costs of the tasks that form the cost model are based on previously measured values for the DSP's processor utilizations and corresponding power consumption including all different QoS-levels of all given services.

The genetic algorithm starts with a random initial population of individuals that represent camera configurations and improves its results up to a maximum number of iterations. The individuals form a population as a function of their costs for both optimization objectives, i.e., power

consumption and service quality. In the given evaluation, the MCO is executed for *s=6* different tasks in up to *q=5* QoS levels on *r=2* DSPs as listed in Table 2. Experimental results from previous performance evaluations of the genetic algorithm have indicated that setting a mutation-rate of *m=0.08*, and a crossover-rate of *c=0.14* respectively is suitable in order to obtain Pareto-optima as fast as possible.

DSPs	Services	Corresponding Number of QoS-Levels
$r = 2$	$s = 6$	$q_1 = 5$, $q_2 = 4$, $q_3 = 2$, $q_4 = 3$, $q_5 = 4$, $q_6 = 5$

Table 2. Parameter setup for the MCO.

For better evaluation of the MCO, the output data of the genetic algorithm has been compared with the Pareto-set of a separately implemented greedy optimization algorithm that considers every possible configuration of the whole search space. In particular, useful optimization results were obtained already after about 30% of the genetic algorithm's total execution time. By this time, the genetic algorithm has already found 90% of all theoretically existing Pareto-optimal camera configurations.

Figure 4 depicts the population of camera configurations that has been computed on the *SmartCam* by the genetic algorithm of the MCO in about 40% of its total execution time. It shows that the optimizer has generated a population that is already close to the theoretical Pareto-set that has been computed by the greedy algorithm that are plotted as *Optimal Individuals* in the figure.

Note that the leftmost point in Figure 4 represents a camera configuration with the most services executed in their best QoS levels but also with the highest power consumption. In contrast, the rightmost point is a configuration with the worst QoS but also with the lowest power consumption due to low device activity in the camera.

As also can be seen in Figure 4, there is a little gap in the computed Pareto-set of camera configurations. It marks the point where it can be switched between camera configurations that utilize one DSP or two DSPs, respectively. In the case of using the single-DSP instead of the two-processor camera configuration, a small degradation of the overall service quality leads to a decrease of power consumption of about 10%.

Because dynamic power management is applied to DSPs of the *SmartCam* and therefore they may get dynamically powered-down, the performance of MCO has been evaluated on the network processor of the *SmartCam*. For the computation of the population depicted in Figure 4, the MCO performed the optimization in $1377ms$ on the IXP425 processor. This turns out as suitable for the given soft real-time requirements in typical IVS-applications. Thus, online MCO is feasible for its implementation on embedded smart cameras.

5. Conclusion

In traffic surveillance there is a trend towards distributed intelligent surveillance cameras. These smart cameras provide on-site video analysis to detect dangerous traffic situations. High performance embedded computing platforms are required to provide enough computing power for the video analysis algorithms. In previous work [2] we developed the *SmartCam* that is a heterogeneous multi-processor prototype of an embedded smart camera. It comprises a network processor and up to ten DSPs.

Limited resources on the embedded platform prohibit to run all analysis algorithms simultaneously. Therefore, all algorithms are loaded and unloaded on demand at runtime. To support communication between dynamically changing algorithms on the DSPs a middleware layer that supports loose coupling of tasks is required. In this work a very light-weight real-time publisher-subscriber middleware (PS-MW) for the *SmartCam* platform is presented. Furthermore, we use a genetic algorithm that is specially tailored for embedded smart cameras for online multi-criterion optimization (MCO) with respect to the two desired objectives 'maximizing the service-level' and 'minimizing the power consumption'. Optimization results are then used to reconfigure services to get an optimal trade-off between power consumption and Quality-of-Service. Experimental results demonstrate the efficiency of the publisher-subscriber implementation and the feasibility of our approach for online computation of power-aware camera configurations.

In ongoing efforts middleware fault tolerance mechanisms are currently intensively investigated to make the *SmartCam* more robust. Future work will also focus on the implementation of a context-sensing and analysis unit

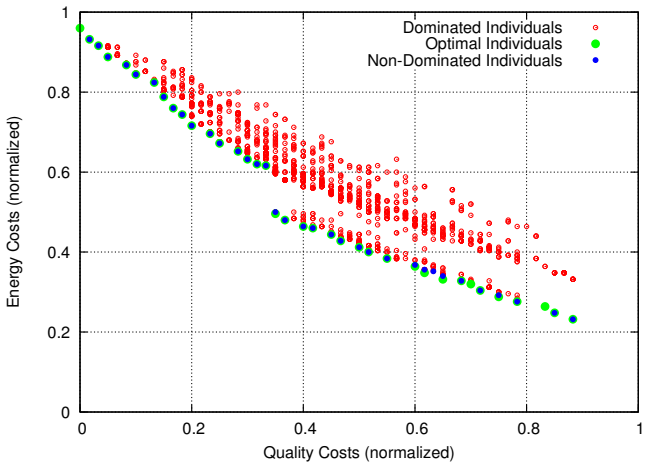

Figure 4. An approximated set of Pareto-optimal camera configurations computed by the MCO.

on the *SmartCam* SW-FW to further improve autonomous operation in a network of many distributed smart cameras.

References

[1] A. Bogliolo, L. Benini, and G. D. Micheli. A Survey of Design Techniques for System-Level Dynamic Power Management. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(3), June 2000.

[2] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach. Distributed embedded smart cameras for surveillance applications. *Computer*, 39(2):40–47, Feb. 2006.

[3] C. C. Coello. A Short Tutorial on Evolutionary Multiobjective Optimization. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, Zuerich, Switzerland*, pages 21–40, 2001.

[4] A. Doblander, B. Rinner, N. Trenkwalder, and A. Zoufal. A middleware framework for dynamic reconfiguration and component composition in embedded smart cameras. *WSEAS Transactions on Computers*, 5(3):574–581, Mar. 2006.

[5] C. Fetzer and K. Högstedt. Self*: A data-flow oriented component framework for pervasive dependability. In *Proceedings of the Eighth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems*, pages 66–73. IEEE, Jan. 2003.

[6] G. L. Foresti, C. Mähönen, and C. S. Regazzoni. *Multimedia video-based surveillance systems*. Kluwer Academic Publishers, 2000.

[7] A. Maier, B. Rinner, W. Schriebl, and H. Schwabach. Online Multi-Criterion Optimization for Dynamic Power-Aware Camera Configuration in Distributed Embedded Surveillance Clusters. In *Proceedings of the 20th IEEE International Conference on Advanced Information Networking and Applications*, Vienna, Austria, Apr. 2006.

[8] C. Mascolo, L. Capra, and W. Emmerich. Mobile computing middleware. In E. Gregori, G. Anastasi, and S. Basagni, editors, *Advanced Lectures on Networking: NETWORKING 2002 Tutorials*, volume 2497 of *Lecture Notes in Computer Science*, pages 20–52. Springer, 2002.

[9] K. Miettinen, editor. *Evolutionary Algorithms in Engineering and Computer Science*. John Wiley and Sons, 1999.

[10] C. Nitsch and U. Kebschull. The use of runtime configuration capabilities for network embedded systems. In *Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition*, page 1093. IEEE Computer Society, Mar. 2002.

[11] E. Pitt and K. McNiff. *Java.rmi: The Remote Method Invocation Guide*. Addison Wesley, June 2001.

[12] A. Pope. *The Corba Reference Guide: Understanding the Common Oject Request Broker Architecture*. Addison Wesley, Jan. 1998.

[13] R. Rajkumar, M. Gagliardi, and L. Sha. The real-time publisher/subscriber inter-process communication model for distributed real-time systems: Design and implementation. In *Proceedings of the Real-Time Technology and Applications Symposium*, pages 66–75. IEEE, May 1995.

[14] R. Sessions. *COM and DCOM: Microsoft's Vision for Distributed Objects*. John Wiley & Sons, 1997.

[15] W. Wolf, B. Ozer, and T. Lv. Smart cameras as embedded systems. *Computer*, 35(9):48–53, Sept. 2002.