Embedded Smart Cameras as Key Components in Reactive Sensor Systems

M. Bramberger[1], A. Doblander[1], M. Jovanovic[1], A. Klausner[1], A. Maier[1], B. Rinner[1] A. Tengg[1]

1: Institute for Technical Informatics, Graz University of Technology
Inffeldgasse 16, 8010 Graz, AUSTRIA
web: www.iti.tugraz.at

Abstract Smart cameras combine video sensing, video processing and communication onto a single embedded device. Due to their high onboard processing and communication power they are able to perform sophisticated video tasks in real time and can, therefore, react to changes in the camera's environment. Such embedded smart cameras are excellent candidates for reactive sensor systems.

In this paper we describe the necessary hard- and software components in order to perform high level video processing onboard. We then present a multi-camera setup which is able to autonomously react to significant changes in the observed environment. We further present ongoing research in extending our smart camera platform towards an autonomous distributed smart sensor system.

Keywords: smart camera; distributed embedded system; traffic surveillance; intelligent infrastructure; multi-sensor systems

1. Introduction

In many (embedded) applications, the computer system is connected to its environment via sensors and actuators. The data gathered from the sensors is exploited to derive information about the current status of the system's environment. Typically, this transformation from the sensor data to the derived information can be considered as a passive component. Active sensing, on the other hand, takes into account the evolution of the environment and adapts the actuators and the data processing in order to improve the quality of the sensory system. Active sensing mimics the human sensing system and is successfully applied in areas such as active vision.

Active sensing poses strong requirements on the sensory system and computing system. These requirements must be addressed both in hardware and software. Typically, data captured at several sensors must be processed and combined and the actuators must react within tight time frames. Thus, the main requirements include (i) high data rates from the sensors to the processing units, (ii) high computation performance, and (iii) sophisticated algorithms for data abstraction and fusion. A promising solution for solving these requirements is to integrate sensing and computation into a high-performance sensor platform.

In this paper we present a smart camera prototype [1] and discuss its potential for active sensing. Smart cameras combine video sensing, video processing and communication onto a single embedded device. Smart cameras not only capture and compress the grabbed video stream, but also perform sophisticated, real-time, on-board video analysis of the captured scene. These cameras help to migrate the video processing from central base stations to the sensors. In distributed video applications such as surveillance, smart cameras help to (i) reduce the communication bandwidth between camera and base station, (ii) decentralize the overall surveillance system and hence to improve the availability as well, and (iii) deploy more surveillance tasks than with traditional cameras [2].

The remainder of this paper is organized as follows: Section 2 briefly describes related work. Section 3 presents our smart camera platform and its prototype implementation. Sections 4 and 5 describe our ongoing work towards distributed smart cameras and intelligent multi-sensor systems Section 6 concludes this paper with a short discussion.

2. Related Work

In [3], Wolf et. al propose a smart camera, that is used to detect people and analyze their gestures in real-time. By using these cameras, intelligent environments and intelligent classrooms, respectively, can be created, which analyze the behavior of people, and initiates appropriate actions.

The proposed smart camera is optimized for the special application "gesture recognition". However, these cameras are not reconfigurable. The prototype of their smart camera is based on Philips Trimedia-processors, which are integrated into a host computer. The prototype is fed with analog video-data from external cameras. Subsequent research, however, is focused on the design and implementation of a specialized ASIC improving real-time video-analysis performance [4]. In order to reduce the size and cost of a smart camera, the designed ASIC is planned to be packaged with an image sensor into a system-in-package (SiP).

In [5], Caarls et. al aim at designing application-specific smart cameras, which integrate a sensor, single-instruction-multiple-data (SIMD) processors, and instruction-level parallelism (ILP) processors on a single chip. Therefore, smart cameras employing this application-specific chip are extraordinary small and simple in design.

Other research in specialized architectures for smart cameras has been performed by Atiquzzaman et. al [6] and Matsushita et. al [7]. These research projects concentrate on the design and implementation of low-level image analysis tasks (convolution, edge-detection) in field-programmable gate arrays (FPGA). Research by Pinz et al. [8] and Berry et al. [9] also focus on the development of dedicated hardware using CMOS sensors and FPGAs for high-performance image processing. They have demonstrated their cameras among others in fast tracking applications.

3. Smart Embedded Camera

Smart cameras are key components of novel vision-based systems. Surveillance is an excellent example for such applications. The increased functionality, flexibility and scalability of novel surveillance systems require very high computing and communication capabilities of an embedded platform for smart cameras.

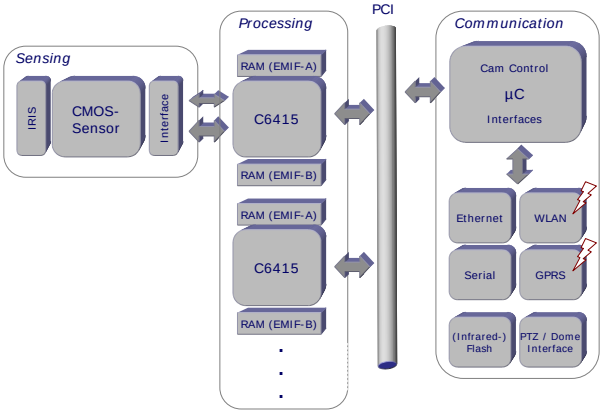

Figure 1: SmartCam architecture.

3.1 Hardware Architecture

Our smart camera (SmartCam) [2] has been designed as a low-power, high-performance embedded system. Figure 1 depicts the camera's main units: (i) the *sensing unit*, (ii) the *processing unit*, and (iii) the *communication unit*. A high-dynamic, monochrome CMOS image sensor is the heart of the sensing unit. It delivers images with VGA resolution and up to 30 frames per second and transfers the captured images via a FIFO memory to the processing unit. The processing unit can be equipped with up to ten TMS320C64x DSPs from Texas Instruments. The computing performance of this scalable architecture can be adapted to the requirements of the real-time video analysis and compression tasks targeted for the smart camera. An aggregate performance of up to 80 GIPS can be delivered by the DSPs while keeping the power consumption low.

The DSPs are coupled via a local PCI bus which also serves as the connection to the network processor (Intel XScale IXP425). The network processor establishes the connection between processing and communication unit and controls internal and external communication. Internal communication is performed via the PCI bus between either the DSPs or the DSPs and the network processor. The communication unit currently supports two interfaces for IP-based external communication: wired Ethernet and wireless GSM/GPRS.

3.2 Software Architecture

The software architecture of our smart camera is designed for flexibility and reconfigurability. It consists of several layers which can be grouped into (i) the *DSP-Framework*, running on the DSPs, and (ii) the *SmartCam-Framework*, running on the network processor. This architecture is based on the abstraction that the application logic is running on the network processor and loads and unloads the actual analysis algorithms onto the DSPs as needed.

The DSP-Framework runs on every DSP in the system. 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. Service management facilities are needed to allow algorithms on different DSPs to establish connections to each other dynamically. That is important because all algorithms can be loaded and unloaded at runtime by the "Dynamic Loader" module. Thus, most framework's components can be dynamically loaded at runtime. The DSP-Framework is built upon the DSP/BIOS operating system from Texas Instruments.

The SmartCam-Framework serves two main purposes. First, it 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 DSP algorithms. The SmartCam-Framework is running on top of a standard Linux kernel.

3.3 Prototype Implementation

A prototype of the SmartCam has been developed using components-off-the-shelf (COTS) hardware components in order to test and evaluate the distributed surveillance system [11]. This prototype serves as feasibility demonstration of our approach. It provides the required computing and communication performance. However, the prototype implementation lacks in providing the desired form factor and reliability for real world employment in harsh environments.

The IXDP425 development board from Intel serves as camera platform. This baseboard is equipped with an Intel IXP425 XScale network processor running at 533 MHz, 256 MB external memory and four PCI slots. The IXP425 provides on-chip support for Ethernet access, multiple serial ports, and most importantly, an on-chip PCI host controller.

Network Video Development Kits (NVDK) from ATEME serve as the DSP platform. These PCI boards are plugged into the base board and are comprised of Texas Instruments TMS320C6416 DSPs running at 600 MHz. Each NVDK board is equipped with a total of 264 MB of

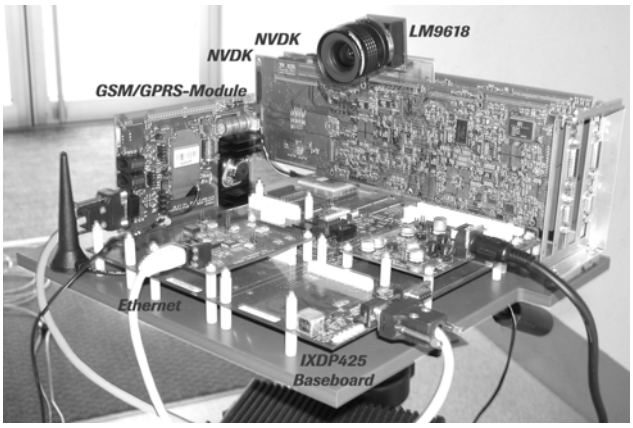

Figure 3: SmartCam prototype.

memory accessible via two different external memory interfaces of the DSP.

The monochrome sensor LM-9618 from Kodak Eastman captures the images in our SmartCam prototype. This image sensor provides a high-dynamic range of up to 110 dB at VGA resolution. It is connected to one of the DSPs via FIFO memory.

The network processor is operated by Linux (Kernel 2.6.x) which enables access to a broad variety of open-source software modules. For the interoperability with the DSPs the SmartCam-Framework is used. This software framework is executed on top of Linux. Java is also running on top of Linux supporting platform-wide applications.

The DSPs are operated by the DSP/BIOS real-time operating system (RTOS). The DSP-Framework runs on top of this RTOS and serves as the counterpart of the SmartCam-Framework on the network processor.

A mobile agent system (MAS) is applied to support the development of a distributed surveillance system consisting of loosely coupled smart cameras. Mobile agents are most suitable for this distributed application since each surveillance task can be encapsulated within a mobile agent which is then able to migrate between cameras. The MAS supports autonomous operation of the surveillance tasks. Additionally, the mobile agent approach is highly scalable and flexible.

4. Distributed Smart Cameras

An important feature in our distributed smart camera system is *dynamic reconfiguration*. The aim of dynamic reconfiguration is to allow modifications of system's configuration at runtime. Thus, (surveillance) tasks executed on individual cameras can be modified, initialized, terminated and migrated to different cameras while the overall surveillance system is running. By means of this dynamic reconfiguration the surveillance system can be adapted to changes in its environment without restarting the overall system. This is essential in applications which must be operational 24 hours a day.

In order to offer dynamic reconfiguration our SmartCam platform must support dynamic task allocation and task migration. This is achieved by two software components. First, our lightweight software framework on the SmartCams supports dynamic allocation and de-allocation of tasks to individual processors within a camera. Second, the MAS allows dynamic allocation/migration of tasks among various cameras. In our system of distributed smart cameras we have introduced logical groups of cameras called *clusters*. This cluster organization is very helpful in the allocation of tasks to individual cameras [14].

In our research on distributed smart cameras we have implemented two case studies. The first study is on autonomous task allocation and it describes the autonomous distribution of high-level tasks to a cluster of smart cameras. The second study is on multi-camera object tracking.

4.1 Autonomous Task Allocation

The goal of the task allocation system is to autonomously find a mapping of tasks to smart cameras which satisfies all requirements and is optimal with respect to a specified metric, i.e., some cost function. This system is designed to operate fully distributed; hence no central host is required for operation. Since the surveillance tasks have firm real-time requirements, the task allocation system has to take care that no deadlines are missed due to an overload of a camera or a camera's subsystem, respectively.

The re-allocation of tasks may be necessary due to events, raised by hardware or software: (i) Hardware events usually originate from changed resource availability due to added or removed cameras, hardware breakdown, or re-usability of recovered resources. (ii) Software events are caused by changes to resource requirements due to changes in the task set of the surveillance cluster, or because of changes in the quality-of-service level (QoS) of tasks, i.e., due to detected events in the observed scene.

The allocation of tasks to smart cameras is done in two steps. In the first step, all feasible allocations of tasks to smart cameras (allocations where no real-time requirements are violated) are determined. In the second step, the optimal allocation of tasks is chosen by using a cost function. This complex cost function is composed from parts representing (i) the hardware resources, (ii) the required data transfer, (iii) the migration overhead and (iv) the affinity to the observed scene [12, 14].

We have evaluated our task allocation system on several surveillance clusters comprised of different numbers of smart camera prototypes and PCs serving as SmartCam emulations, i.e., only the SmartCam framework has been executed on the PC but not the DSP framework. Our experimental results show that an autonomous task allocation can be performed within a few seconds in a cluster of 3 to 6 cameras [14].

4.2 Multi-Camera Tracking

The basic idea of our tracking approach is that only a single tracker is instantiated in our multi-camera system. This tracker (agent) then "follows" the tracked object, i.e., migrates to the SmartCam which should next observe the object.

In this case study, the task migrating is implemented using diet-agents running under Java as MAS. The tracking algorithm is based on a Kanade-Lucas-Tomasi (KLT) feature tracker [13] which is implemented as an individual agent in the MAS. The KLT tracker has a short initialization time which makes it very applicable for multi-camera object tracking by mobile agents.

The hand-over process is controlled by the tracking agents, which use pre-defined regions in the observed scenes, so-called *migration regions*. When the tracked object enters a migration region the hand-over to the "next" SmartCam is initiated.

4.3 Ongoing Work in Distributed Smart Cameras

Related work [15] shows that dynamic reconfiguration including dynamic configuration is the best solution for multimedia streaming applications due to the optimal use of the available resources. Other approaches show that a mobile agent system is most wide spread as a framework for this dynamic and distributed system.

Currently the SmartCam project advances the existing work in the field of QoS using policies as a structure for managing the dynamic reconfiguration. These policies are dynamic structures and should make self-learning possible. The self-learning algorithm is based on records of accompanied events. Processing this data enables event prediction and the improvement of real time tasks with rigorous requirements as well as the improvement of the QoS.

The ongoing work considers the lack of resources in the most critical situation. In those situations dramatical degradation of QoS parameters or even cancellation of some inessential tasks is inevitable. The aim is to minimize the calculation for new task allocation as much as possible. Furthermore, the self learning should play the key role in this process.

Complex structure of policies define decisional tree for reconfiguration, which is N-decisional tree diagram. Root of this tree is basic event recognition algorithms and those algorithms should have low ballast costs. Every next system state is a result of dynamic reconfiguration and presents the target state of reconfiguration. After event is completely accompanied, the next target state for reconfiguration is global event recognition state (simple motion detection).

Future case studies will focus on the task migration but with the accent on the code migration. The algorithm repository database will be distributed which will enable the complete use of camera memory for extreme cases. This approach can help in situations when memory requirements are very high. Virtual reality could be used as a case study which will provide a good test for our system, but only with the appropriate human behavior recognition algorithm.

5. Towards an Intelligent Multi-Sensor System

Fusing data from various sensors helps to improve the robustness and confidence, to extend the spatial and temporal coverage as well as to reduce ambiguity and uncertainty of the processed sensor data. In the I-SENSE project we exploit these characteristics to improve the quality of various applications such as traffic surveillance. The goal of this project is to extend our SmartCam-platform to a distributed high-performance multi-sensor architecture. Our multi-sensor architecture combines data from different sensors at two stages. First, intra-node fusion takes place at a single sensor node where raw sensor data or abstracted features are combined. Second, inter-node fusion combines abstracted data from various geographically distinct sensor nodes.

In our I-SENSE project not only methods for abstracting and fusing data but also architectural issues must be solved. These important architectural issues include both hardware and software features such as (i) computing and communication performance, (ii) scalability, autonomous operation and configuration, (iii) fault tolerance and graceful degradation as well as (iv) data abstraction and communication.

5.1 Multi-Sensor Data Fusion

As the name multi-sesnor data fusion implies, it is a technique by which data from several sensors are combined through a data processor to provide comprehensive and accurate information. Although the provision of a single data stream from multiple inputs is advantageous, the powerful potential of this technology stems from its ability to track changing conditions and anticipate impacts more consistently than could traditionally be done with a single data source. Thus, multi-sensor data fusion makes it possible to create a synergistic process in which the consolidation of individual data creates a combined resource with a productive value greater than the sum of its parts [16].

In our I-SENSE project three basic categories (or levels) of data fusion will be used. These fusion levels are differentiated according to the amount of information they provide. The most basic level involves the fusion of multi-sensor data to determine the position, velocity, and identity of a tracking object. At this level, however, only raw, uncorrelated data is provided to the user, and therefore this level is called *raw-data fusion*. In comparison, level two data fusion provides a higher level of inference and delivers additional interpretive meaning suggested from the raw data and data will be fused on feature level. Therefore this level is called *feature-level fusion*. Level three data fusion is designed to make assessments and provide recommendations to the user, much as occurs in knowledge-based expert systems (KBES) and is therefore called *decision fusion*. Thus, each jump between data fusion levels represents a corresponding leap in technological complexity to produce increasingly valuable informational detail.

Data fusion is performed on the individual sensor nodes using data from the local sensors as well as abstracted sensor data from the adjacent sensor nodes. In order to realize a flexible and scalable solution our multi-level fusion is based on the following constraints: (i) Sensor fusion is performed distributedly in our I-SENSE

platform, i.e., there is no (single) central fusion node. (ii) Communication of the abstracted sensor data is restricted to adjacent sensor nodes, i.e., the sensor data is not broadcasted over the entire network. (iii) The sensor nodes have no global knowledge about the network topology; the sensor nodes require only knowledge about their neighborhood, or the so called cluster.

5.2 Multi-Level Fusion Architecture

The I-SENSE platform consists of a two-level architecture which is an extension of our SmartCam architecture (cp. Section 3). The top-level is composed of geographically distributed sensor nodes which are connected via a common communication medium (cp. Figure 5). The sensor nodes represent the bottom-level of our I-SENSE platform. They are equipped with high-performance processing and communication elements and can be attached to several heterogeneous and homogeneous sensors. Thus, the sensor nodes are the main processing components of our I-SENSE platform.

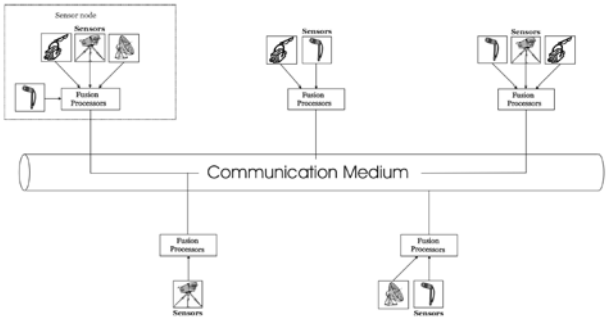

Figure 5: I-SENSE platform: a scalable, distributed embedded system.

Scalability is supported in several ways in our I-SENSE project. First, the number and type of sensors can be adapted at the individual sensor nodes. Second, the processing and communication performance of each sensor node can be easily modified. Third, communication can be performed via wired and wireless connections and finally, the number of sensor nodes can be easily adapted.

The fusion nodes are logical entities on a sensor node that perform the sensor fusion algorithms at the individual fusion levels. These nodes receive (raw) data from the sensors and abstracted data from other fusion nodes, and provide data for other sensor nodes and data consumers. Data consumers represent remote fusion nodes as well as visualization nodes where the combined sensor data can be monitored. The presence of multiple data sources and fusion nodes provides many choices for the architecture, i.e., how the sensors or data sources report to each fusion node and the connectivity among the nodes.

The multi-level data fusion architecture for the I-SENSE project is presented in Figure 6. The three fusion methods (i) raw-data fusion, (ii) feature-level fusion and (iii) decision fusion are the heart of this architecture. The output of these fusion strategies are then combined to derive the current state of the fusion node. Typical methods for deriving this state are filtering, classification, situation detection and prediction. The state of the fusion node can be sent to other (remote) sensor nodes or a visualization node via the communication interface.
The two bottom level methods combine data from sensors directly attached to the fusion node; the top level method uses state information from remote fusion nodes.

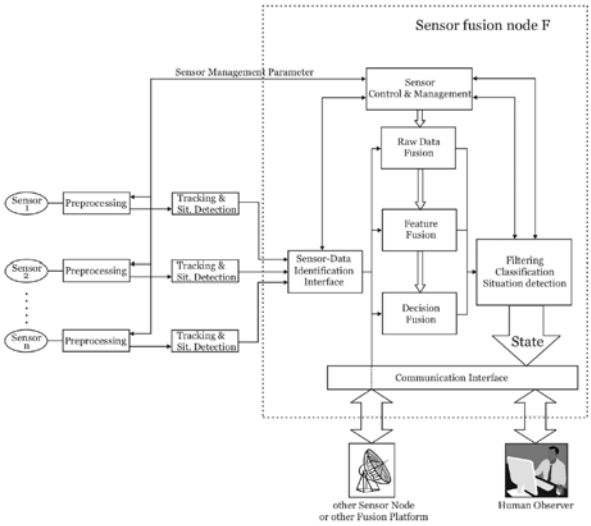

Figure 6: Multi-level data fusion architecture.

5.3 Current State in I-SENSE project

The ePCI-101 Kontron board serves as I-SENSE multi-sensor platform. This baseboard is equipped with Intel´s latest low power processor, the Pentium-M 1,6GHz, with the 855GME chipset delivers outstanding performance, allowing to set the pace for innovative applications today by offering enough headroom for the emerging next generation´s requirements. Network Video Development Kits (NVDK) from ATEME serve as the DSP platform for visual sensing. At this board video data from different visual sensors are captured (e.g., video sensor and infrared visual sensor) and prepared for raw-data fusion. Current work is to integrate image calibration, registration and fusion algorithm to the I-SENSE platform to perform raw-data fusion of visual data. Currently, a case study is started to ascertain the possibilities of integrating high performance audio cards to our proposed system.

6. Conclusion

Our SmartCam prototype is a major step in the development of a fully embedded distributed surveillance system. By migrating most of the computation from a central server to the smart embedded cameras, the system architecture becomes more flexible and scalable, the overall communication bandwidth is reduced and the entire surveillance system is able to autonomously and dynamically react to detected events in the supervised scenes. This novel surveillance architecture poses, however, strong requirements on the camera's hard- and software. Recent technological advances support this trend towards smart cameras. In a subsequent step we currently augment the smart cameras with additional

sensors transforming them into a high-performance multi-sensor system. By combining visual, acoustic, tactile or location-based information, the smart cameras become more ``sensitive'' and are able to deliver more accurate results which make them even wider applicable.

Embedded smart cameras have the potential for the successful deployment in many different applications such as smart environments, intelligent infrastructures and pervasive computing.

Developing this SmartCam prototype has given us insights that may be interesting for many other distributed embedded systems as well. We believe that the key to successful deployment of smart cameras is (i) the integration of sensing, computing and communication in a small, power-aware embedded device; (ii) the availability of high-level image/video processing algorithms/libraries for the embedded target processors (DSPs); (iii) a light-weight software framework supporting glueless intra- and inter-camera communication; and (iv) the availability of various system-level services such as task mapping and QoS adaptation to enable autonomous and dynamic operation of the overall multi-camera system.

7. References

[1] M. Bramberger, B. Rinner, and H. Schwabach. An Embedded Smart Camera on a Scalable, Heterogeneous Multi-DSP System. In Proc. of the European DSP Education and Research Symposium (EDERS 2004), Birmingham, U.K. Nov. 2004

[2] M. Bramberger, A. Doblander, A. Maier, B. Rinner and H. Schwabach. Distributed Embedded Smart Cameras for Surveillance Applications. IEEE Computer, to appear

[3] W. Wolf, B. Ozer, and T. Lv. Smart Cameras as Embedded Systems. IEEE Computer, 35(9):48-53, September 2002.

[4] T. Lv, B. Ozer, S. T. Chakradhar, J. Xu, W. Wolf, and J. Henkel. A methodology for architectural design of multimedia multiprocessor SoCs. IEEE Design & Test of Computers, 22(1):18-26, 2005.

[5] W. Caarls, P.P Jonker, and H. Corporaal. SmartCam Design Framework. In Proceedings of the 4th Progress Symposium on Embedded Systems, Oct 2003.

[6] M. Atiquzzaman, M.G. Hartley, and M.Y. Siyal. A novel multiprocessor architecture for low-level image processing applied to road-traffic data capture and analysis. In Proceedings of the Second Asian Conference on Computer Vision, pages 504-508, Dec 1995.

[7] N. Matsushita, D. Hihara, T. Ushiro, S. Yoshimura, J. Rekimoto, and Y. Yamamoto. Id cam: a smart camera for scene capturing and id recognition. In Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality, pages 227-236. IEEE, Oct 2003.

[8] U. Muehlmann, M. Ribo, P. Lang, and A. Pinz. A new high speed cmos camera for real-time tracking applications. In IEEE International Conference on Robotics and Automation, pages 5195-5200, 2004.

[9] P. Chalimbaud, F. Berry, P. Martinet In Proceedings of the 33rd International Symposium on Robotics, ISR'02, Stockholm - Sweden, October 2002.

[10] R. Steinmetz and K. Nahrstedt. Multimedia Systems. Springer, 2004.

[11] A. Maier, B. Rinner, H. Schwabach,. A Hierarchical Approach for Energy-Aware Distributed. In Proceedings of the IEEE/IFIP International Workshop on Parallel and Distributed Embedded Systems. Fukuoka, Japan, July 2005

[12] M. Bramberger, B. Rinner, and H. Schwabach. A Method for Dynamic Allocation of Tasks in Clusters of Embedded Smart Cameras. In Proc. IEEE Conference of Systems, Man and Cybernetics, Hawaii, U.S.A., Oct. 2005

[13] J. Shi and C. Tomasi. Good features to Track. In Proc. of the International IEEE Conference on Computer Vision and Pattern Recognition, pages 593-600, Seattle, U.S.A. June 1994

[14] M. Bramberger. Distributed Dynamic Task Allocation in Clusters of Embedded Smart Cameras. PhD thesis. Graz University of Technology, 2005

[15] Reconfiguration-based QoS management in multimedia streaming applications Layaida, O.; Atallah, S.B.; Hagimont, D.; Euromicro Conference, 2004. Proceedings. 30th 2004

[16] J.K. Hackett, M. Shah. Multi-sensor Fusion: A Perspective. Proceedings IEEE International Conference on Robotics and Automation, May 1990