A Mobile Agent-based System for Dynamic Task Allocation in Clusters of Embedded Smart Cameras

Michael Bramberger[1], Bernhard Rinner[1] and
Helmut Schwabach[2]

[1]Institute for Technical Informatics,
Graz University of Technology, Graz, Austria
{bramberger,rinner}@iti.tugraz.at

[2]Video & Safety Technology,
ARC seibersdorf research, Seibersdorf, Austria
helmut.schwabach@arcs.ac.at

Abstract — *This paper presents a dynamic task allocation method for smart cameras targeting traffic surveillance. Since our target platforms are distributed embedded systems with limited resources, the task allocation has to be light-weight, flexible as well as scalable and has to support real-time requirements. Therefore, surveillance tasks are not allocated to smart cameras directly, but to groups of smart cameras, so called surveillance clusters. We formulate the allocation problem as a distributed constraint satisfaction problem (DCSP) and present a distributed method for finding feasible allocations. Finally, a cost function is used to determine the optimal allocation of tasks. We have realized this dynamic task allocation using heterogeneous, mobile agents which utilize their agencies and our embedded software framework to find the most appropriate mapping of tasks in a distributed manner. The dynamic task allocation has been implemented on our smart cameras (SmartCam) which are comprised of a network processor and several digital signal processors (DSPs) and provide a complex software framework.*

1 Introduction

Surveillance systems currently undergo a dramatic shift. Traditional surveillance systems of the first and second generation have employed analog CCTV cameras, which transferred the analog video data to digital base stations where the video analysis, storage and retrieval takes place. Current semiconductor technology enables surveillance systems to leap forward to third generation systems which employ digital cameras with on-board video compression and communication capabilities. *Smart cameras* [1] [2] even go one step further; they not only capture and compress the grabbed video stream, but also perform sophisticated, real-time, on-board video analysis of the captured scene. Smart cam-

eras help in (i) reducing the communication bandwidth between camera and base station, (ii) decentralizing the overall surveillance system and hence to improve the fault tolerance, as well as (iii) realizing more surveillance tasks than with traditional cameras.

Typical tasks to be run in a surveillance system targeting traffic surveillance include MPEG-4 video compression, various video analysis algorithms such as accident detection, wrong-way drivers detection and stationary vehicle detection. Additionally, traffic parameters such as average speed, lane occupancy and vehicle classification are often required. Since computing power is limited we may not allocate all tasks to the cameras.

This paper presents a resource-aware dynamic task allocation system for smart cameras targeting traffic surveillance. Since our target platforms are distributed embedded systems with limited resources, the task allocation has to be light-weight, flexible, scalable, and has to support real-time requirements. Therefore, surveillance tasks are not allocated to smart cameras directly, but to groups of smart cameras, so called surveillance clusters. We formulate the allocation problem as a distributed constraint satisfaction problem (DCSP) and present a distributed method for finding feasible allocations. Finally, a cost function is used to determine the optimal allocation of tasks. We have realized this dynamic task allocation using heterogeneous, mobile agents which utilize their agencies and our embedded software framework to find the most appropriate mapping of tasks in a distributed manner

The main contributions of this ongoing research project include (i) the distributed agent-based approach for determining all feasible allocations of a DCSP, (ii) the evaluation of a complex cost function considering the limited resources of the embedded platform in detail, (iii) the integration of a mobile agent system in our embedded software framework. The dynamic task allocation has been implemented on our smart cameras (*SmartCam* [3]) which are comprised of a network processor and several digital signal processors (DSPs) and provide a complex software framework.

The remainder of this paper is organized as follows: Section 1.1 sketches related work. Section 2 describes the DCSP approach and focuses on finding feasible allocations of tasks and the cost function. Section 3 and 4 present the implementation and the experimental results, respectively. Section 5 concludes the paper with a summary and an outlook on future work.

1.1 Related Work

Agents-based load distribution has been an active research in the last decade. An overview of agent standards and available platforms is presented in [4]. WYA (While You're Away) [5] is based on the NOMADS mobile agent system. It provides dynamic load balancing by mobile agents, which utilize a central coordinator to move between workstations. Qin et al [6] identified the amount I/O-traffic as an important issue for dynamic load balancing beside CPU-load and memory. Chow and Kwok [7] introduced the affinity of an agent to its machine and used a credit-based scheme to determine the agents to be migrated by using a central host station. Agents used in surveillance systems were proposed by the MODEST consortium [8], where mobile agents were used to track vehicles using PC based platforms.

Distributed constraint satisfaction problems have been analyzed by Yokoo et al. in [9] for variable-split CSPs. The presented asynchronous backtracking algorithm used

autonomous agents, however, the high communication effort between the agents is impractical for real-time systems.

2 Task Allocation

The goal of the task allocation system is to 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. 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: (1) Hardware events usually originate from changed resource availability due to added or removed cameras, hardware breakdown, or re-usability of recovered resources. (2) 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. (1) In the first step, all feasible allocations of tasks to smart cameras (allocations where no real-time requirements are violated) are determined. (2) In the second step, the optimal allocation of tasks is chosen by using a cost function.

2.1 Find Feasible Allocations

The determination of feasible allocations of tasks to smart cameras is a distributed constraint satisfaction problem (CSP) [9]. CSPs are defined by a set of n variables $T = \{T_1, ..., T_n\}$, which hold values of a finite domain D, and a set of k constraints $C = \{C_1, ..., C_k\}$. In our case, the surveillance tasks to be allocated to the cameras of the surveillance cluster are the variables, and the values they hold is the identifier of the allocated smart camera $D = \{1, ..., m\}$. Therefore, $T_i = d$ indicates, that task i is currently allocated to smart camera d.

Equation 1 determines all permutations with repetitions A, denoted by $\prod^R$, from the m cameras and n tasks. The subsets of A are allocations of tasks to cameras, which have to be checked for feasibility. The constraints (cp. eq. 4 are formulated using two functions which determine the resource requirements and availability of the tasks and the cameras. $req(Res, i)$ determines the requirements of task i concerning resource Res, while $avail(Res, d)$ determines the availability of the resource Res on smart camera d.

$$A = \prod^R (m, n) = \{\ A_1, ..., A_p\ \};\ p = m^n \tag{1}$$

$$A_j = \{\ a_1, ..., a_n\ \} \mid_{a_i \in D} \tag{2}$$

$$R = \{CPU, MEM, DMA\} \tag{3}$$

$$C = \{\ \forall_{A_j \in A} : \ \forall_{d \in D} : \ \forall_{r \in R} : \ \forall_{a_i \in A_j} \forall_{a_i = d} : \ (\sum_i req(r, i)) < avail(r, d)\} \tag{4}$$

Equations 1 to 4 define a feasible allocation, i.e., any allocation A which does not violate any resource constraint C. Note that in our implementation we also considered the DSPs

memory hierarchy and several parameters of the DMA subsystem (cp. resource costs in section 2.2.1).

However, this approach requires a central host. A more scalable solution is to distribute the CSP to several cameras in the surveillance cluster.

2.1.1 The Distributed CSP

In order to distribute the determination of feasible allocations to all smart cameras, we split the domain D into m sub-domains $D_1, ..., D_m$, each comprised of a single value $D_l = l$. The set of tasks T remains the same, while the complexity of the constraints is reduced slightly (cp. equation 5).

To achieve all possible allocations of tasks, we choose $\forall_{0<i\leq n}$ tasks out of the set of all tasks $\forall_{0<i\leq n} : A_i = choose(i, n)$. Every set A_i is comprised of ν allocations $A_i = \{\ A_1^i, ..., A_\nu^i\ \}$, each of which consisting of i tasks $A_\nu^i = \{\ a_1, ..., a_i\ \} \mid \forall_{a_\eta \in T}$. All allocations A_ν^i, which meet the constraints (cp. equation 5) for all resources R, are chosen as feasible, partial allocations.

$$C = \{\ \forall_{A_i \in A} : \forall_{A_\nu^i \in A_i} : \forall_{a_j \in A_\nu^i} : \forall_{r \in R} : (\sum_j req(r, a_j)) < avail(r, d)\ \} \tag{5}$$

Note that the constraints only consider tasks allocated to a single smart camera, so the CSP is split into m sub-CSPs, which can be solved in parallel on the m smart cameras in the surveillance cluster.

Normally, CSPs require that all variables are included in the solutions of the problem, however, in our case this rule would imply that only partial allocations are valid, which include all n tasks of the surveillance cluster. In order to accept task allocations which do not include all tasks, the CSP rules have to be weakened. Even the allocation of no task is a valid allocation, which is reasonable, if a surveillance cluster contains less tasks than smart cameras.

2.1.2 Merging of Allocations

Finally, m sets of partial task allocations are available (cp. eq. 6), which have to be merged in order to find all feasible task allocations.

$$P = \{\ P_1, ..., P_m\ \}; P_j = \{\ P_1^j, ..., P_\nu^j\ \} \tag{6}$$

Consequently, valid task allocations (1) do not allocate tasks to more cameras concurrently, and (2) include all surveillance tasks.

The merging of partial allocations can be done in parallel, as long as rule 1 is not violated. However, the final merging process has to obey rule 2 too.

The merging step of the partial allocations finally provides a set of f feasible allocations F, from which the most appropriate allocation has to be chosen.

2.2 Find Optimal Allocation

For finding the most appropriate allocation of tasks a cost-scheme is used. This cost-calculation scheme includes five cost classes: (1) Resource costs (C_R), (2) data-transfer

costs (C_T), (3) migration costs (C_M), (4) affinity costs (C_A), and (5) quality-of-service costs (C_{QoS}). These costs are calculated for every task on every smart camera in the surveillance cluster. Finally, the overall costs are accumulated as enlisted in the set of feasible allocations F. The following section presents the five cost classes, however, a more detailed description of the cost calculation can be found in [10].

2.2.1 Cost Calculation

To compute the overall cost C_{Tot} of a surveillance task, the cost values of the five cost classes are weighted and added:

$$C_{Tot} = k_R * C_R + k_T * C_T + k_M * C_M + k_A * C_A + k_{QoS} * C_{QoS} \tag{7}$$

Resource Cost The resource costs are composed of four resources: (1) CPU load, (2) memory usage (including memory hierarchy on DSPs), (3) DMA utilization (including channels, reload-tables, and transfer complete codes), and (4) memory bus utilization.

Data-Transfer Cost We have defined two different types of data transfers: (1) Data entering or leaving the system will usually be transmitted using a network interface, while (2) intermediate data transfers operate internally via PCI bus or directly to memory. To consider the bandwidth of the used interface, the data-transfer cost C_T is calculated as the amount of megabytes (MB) transferred, multiplied by a slowness factor depending on the bandwidth of the connection.

Migration Cost The migration of tasks between smart cameras is accounted for by the migration costs depending on (1) required data transfer, and (2) the algorithms downtime due to migration.

Affinity Cost Affinity costs express the priority of tasks inside a surveillance cluster (cluster affine tasks), and additionally provide the possibility to bind a surveillance task to a specific camera (scene affine task), as it is required for tracking tasks, for example.

Quality-of-Service Cost In oder to allow a more fine-grained control over the system, surveillance tasks may feature several quality-of-service (QoS) levels with different quality levels and, therefore, different resource requirements. Usually it is desired to run all tasks with highest possible quality, therefore, this cost adds penalty costs if the task should be run with lower QoS-levels (e.g. due to insufficient resource availability for running at best quality). These penalty costs and the number of QoS-levels are defined by the task designer.

2.3 Reallocation Scenarios

As mentioned before, two scenarios are possible, where a reallocation of tasks is necessary. However, we handle the scenarios differently to improve the performance of the system.

Increased Requirements / Decreased Resource Availability Increased requirements or decreased resource availability indicates a higher system load. Therefore, the number of feasible allocations will decrease, since less combinations of tasks will satisfy the constraints of the CSP. Therefore, the value of the changed resource or availability is updated in the set of feasible allocations F. Finally, the feasible allocations are re-checked and infeasible allocations are removed from F. This re-checking process does not require to recalculate all possible allocations, therefore, the time required for determination of the new allocation is reduced dramatically.

Decreased Requirements / Increased Resource Availability In contrast to the upper scenario, the decrease of requirements or increase of resource availability can be seen as a reduced load of the system. Consequently, the number of both, partial and complete, allocations will increase. Therefore, all feasible allocations have to be re-computed. This approach leads to longer execution times, however, as the service quality will rise by this reallocation, the real-time deadlines are not as firm as in the upper scenario.

3 Implementation

For the implementation of our surveillance system we have chosen to use mobile agent technology. Mobile agents are most suitable for our system, since they support mobility, autonomy, and platform independence.

3.1 Agent System

We have chosen to use the diet-agents[1] system since it includes all required features like mobility, autonomy, and platform independence and is reasonable small, which is inevitable for embedded systems.

However, extensions to the agent system were necessary to add the support for the DSPs, and the decentralized management of the surveillance clusters.

DSP Agencies Agencies provide the environment for agents to live in. An extension to standard agencies is required to support multiple surveillance clusters. Therefore, a logical grouping of agents within the agency is desirable. We have implemented a cluster-information agent (CIA), which hosts local knowledge of other cameras belonging to the surveillance cluster, and provides lookup services for agents within the surveillance cluster. The integration of the DSPs into the agent system is done by one DSP integration agent (DIA) per DSP in the system. This DIA is a stationary agent, which manages message registrations, message dispatching, and binary downloads to the DSP.

Two basic agent types are deployed within the agent system: (1) Management-oriented SmartCam-agents, which are not included in the task allocation system, and (2) DSP agents, which implement algorithmic functionality to be executed on a DSP. Consequently, DSP-agents represent the tasks allocated by the presented allocation system. Therfore the next paragraph focuses on these agents.

[1] see http://diet-agents.sourceforge.net

DSP Agents DSP agents are an enhancement of standard agents, as they are the base of the task allocation system. Therefore, DSP agents include their requirements to the system and their cost parameters. Additionally, the cost calculation functionality is added to these agents.As the computational intensive parts of these agents are executed on the DSPs, the agents contain a binary, which is downloaded to the DSP and dynamically integrated into the system. Therefore, the agents maintains communication channels to the DSP by using the DSP integration-agent (DIA).

In case of migration the agent notifies the DSPs task to stop computation and to transmit the persistent intermediate results to the agent. After migration the agent loads the binary to the DSP and transmits the stored intermediate results to the DSP. This enables the algorithm to continue computation or to use the intermediate results as new starting values, respectively.

3.2 Task Allocation System

This section outlines the actions taken during a reconfiguration of a surveillance cluster. Since our goal is a flexible, scalable and distributed implementation we have chosen to realize the task allocation system using mobile agents.

A new allocation is always initiated by a smart camera, either due to a hardware event (changes of availability) or a software event (changes of requirements). This camera initiates the reallocation by broadcasting the *request-for-requirements* to all agents, that encapsulate the surveillance tasks, within the surveillance cluster. After the agents receive this request, they broadcast their requirements to *all* smart cameras in their surveillance cluster. Additionally, all agents calculate their costs for every smart camera and transmit the cost values to the initiating camera. The smart cameras determine in parallel the *partial allocations*, which have to be merged in the next step. As the smart cameras are comprised of more DSPs, the partial allocations for the DSPs are merged. To further merge the partial allocations, the half of the smart cameras (determined during the creation of the surveillance cluster) create *allocation-merging agents (AMA)*, which pick up the merged partial allocation and migrate to predetermined smart cameras (which comprise the other half of the surveillance cluster), where they grab the partial allocation, and merge both, the local and the included partial allocations. After the merging process, the agents migrate to the next smart camera, as defined by the fixed itinerary, and merge their partial allocation with the partial allocation provided by another AMA. This way the partial allocations are merged to a final allocation, whereas the last allocation merge is done on the initiating smart camera. The merging process corresponds to a binary tree, therefore $ld(m)$ (where m is the number of smart cameras) sequential steps have to be done.

The initiating smart camera has to determine the most appropriate task allocation, therefore, the costs, as submitted by the agents, are accumulated for every task allocation. Consequently, the task allocation with the lowest cost is selected as the new task allocation. Since the selection of the most appropriate task allocation is based on the costs of the agents, the desired optimization goal like the number of migrations, degradation of quality-of-service, or balanced use of resources can be achieved by adapting the scaling factors of the cost classes. Finally, the new task allocation is broadcast to all smart cameras and agents, which may update their QoS level, or migrate to another smart camera

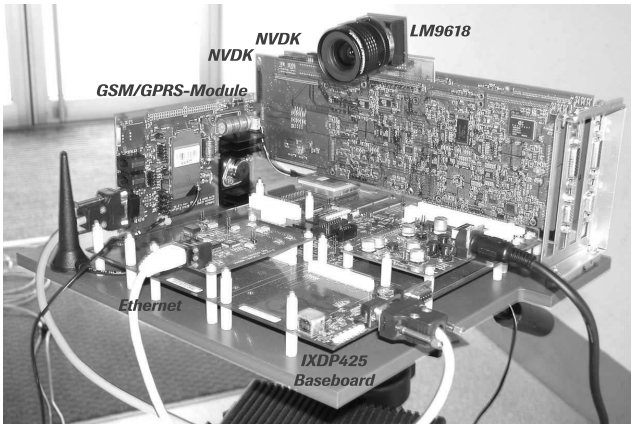

Figure 1: The prototype of the smart camera

then. To enable the fast reconfiguration by removing feasible task allocations, also the set of feasible task allocations is broadcast to all smart cameras in the surveillance cluster after the system has settled.

3.3 Hardware Setup

To verify, test and evaluate the presented agent system, we have used two hardware platforms. A prototype of our smart camera and PCs equipped with DSP boards.

The prototype of our smart camera consists of an *Intel IXDP425 Development Board*, which is equipped with an *Intel IXP425* network processor running at 533 MHz, as well as two *Network Video Development Kits (NVDK)* from ATEME. Each board is comprised of a TMS320C6416 DSP from Texas Instruments, running at 600 MHz, with a total of 264 MB of on-board memory. Image acquisition is done using the *National Semiconductor LM9618* monochrome CMOS image sensor, which is connected to one of the DSP boards.

Due to the lack of additional smart camera prototypes, we are using two Pentium-III personal computers (PCs) running at 1 GHz, which are equipped with one Network Video Development Kit (NVDK) from ATEME.

4 Experiments

In order to verify and evaluate the presented task allocation system, we used the smart camera prototype and two PCs. Consequently, we have setup a surveillance cluster containing these three “cameras”. We have used four different agent types for our experiments: (1) A MPEG-4 video compression agent, (2) a stationary-vehicle detection (StVD) agent, (3) a vehicle count agent, and (4) a vehicle classifier agent. The MPEG-4 agent and the stationary-vehicle detection agent [2] are well tested and evaluated agents, however, the vehicle count and vehicle classify agents only simulated real behavior. A total of six agents have been instantiated from these four classes, where we used three MPEG-agents (as every camera has to transmit a live video stream), and one agent of every other agent-type.

Table 1 enlists the execution times of the two stages of the task allocation. On the PCs

Case	Executed by	PC JDK 1.4.2	PC JamVM	SmartCam JamVM	SmartCam Native/C++
1a	Partial Allocations (6 ag.)	20 ms	14 ms	79 ms	13 ms
1b	Partial Allocations (3 ag.)	14 ms	7 ms	55 ms	9 ms
2a	Merge Solutions (6 ag.)	766 ms	4852 ms	21363 ms	2360 ms
2b	Merge Solutions (3 ag.)	9 ms	5 ms	31 ms	4 ms
3a	Overall running time (6 ag.)	786 ms	4866 ms	21442 ms	2373 ms
3b	Overall running time (3 ag.)	23 ms	12 ms	86 ms	13 ms
4	Pruning of allocations (6 ag.)	14 ms	32 ms	172 ms	26 ms

Table 1: Running times of task allocation

we have used two Java virtual-machines: (a) Sun's JDK 1.4.2, which uses a just-in-time (JIT) compiler, and therefore reaches better performance values, while (b) the JamVM virtual machine [2] is designed as an interpreter. JamVM was also used on the prototype of the smart camera (c), and finally we have implemented the core calculation functions (determine partial allocations, allocation merging) in C++ to estimate the maximum performance (d).

Lines 1a, 2a, and 3a represent the execution times if all six tasks were considered. In a second scenario we have removed the MPEG-4 encoding agents from the surveillance cluster, and allocated them statically to the cameras. The results of the dynamic allocation of the resulting three agents is enlisted in lines 1b, 2b, and 3b in table 1.

Lines 1a and 1b show that the determination of partial allocations only requires a low amount of time. However, lines 2a and 2b enlist that the merging process requires most of the computation time. It can be seen, that merging using the JamVM requires up to 21 seconds, which is unacceptable. The C++ implementation, however, performs better, but also requires about 2 seconds for computation. The running time of the merging stage rises exponentially with the number of agents and cameras, since the number of feasible allocations rises exponentially, too (e.g. there are 2880 feasible allocations in case 2a). Therefore, a reduction of feasible allocations would yield in better performance.

Lines 3a and 3b enlist the overall execution time, as the sum of finding the partial allocations and the merging process.

Finally, we have also evaluated the performance of the task allocation system at increasing system load, in which case the previously determined allocations a rechecked for feasibility. Line 4 enlists the times required to recheck all 2880 allocations. Note that even running JamVM on our prototype delivers a new allocation within 172 ms, which is sufficient.

5 Conclusion

In this paper we have presented a resource-aware dynamic task-allocation system targeting embedded smart cameras. Surveillance tasks are not allocated to the smart cameras directly, but to groups of smart cameras, surveillance clusters, within the tasks are allocated by the smart cameras in a distributed manner. We show that the task allocation can

[2] http://jamvm.sourceforge.net

be formulated as a distributed constraint satisfaction problem (DCSP) and present a solution for this DCSP. Therefore we combine the results of the DCSP with a cost function to retrieve the optimal allocation of tasks. Finally, we discuss the mobile-agent based implementation of the system and present results achieved by evaluation of the implemented system.

Future work includes (1) the further evaluation of the system, using more complex scenarios, (2) the tighter combination of finding feasible allocations and calculating the costs for an allocation in order to delete expensive allocations at an early stage, (3) the test and evaluation of the system in real-world scenarios, and (4) the integration of learning agents into surveillance clusters, which influence the behavior of the system by adapting the cost function based on previous behavior, events, and actions.

References

[1] W. Wolf, B. Ozer, and T. Lv. Smart Cameras as Embedded Systems. *IEEE Computer*, 35(9):48–53, Sep 2002.

[2] M. Bramberger, J. Brunner, B. Rinner, and H. Schwabach. Real-Time Video Analysis on an Embedded Smart Camera for Traffic Surveillance. In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 174–181, 2004.

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

[4] M. K. Perdikeas, F. G. Chatzipapadopoulos, I. S. Venieris, and G. Marino. Mobile agent standards and available platforms. *Elsevier Computer Networks*, (31), 1999.

[5] N. Suri, P.T. Groth, and J.M. Bradshaw. While You're Away: A System for Load-Balancing and Resource Sharing based on Mobile Agents. In *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 470–473, 2001.

[6] X. Qin, Q. Zhu, and D. Swanson. A Dynamic Load Balancing Scheme for I/O-Intensive Applications in Distributed Systems. In *Proceedinges of the 2003 International Conference on Parallel Processing Workshops*. IEEE, 2003.

[7] K.-P. Chow and Y.-K. Kwok. On Load Balancing for Distributed Multiagent Computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(8):787–801, Aug 2002.

[8] B. Abreu, L. Botelho, A. Cavallaro, D. Douxchamps, T. Ebrahimi, P. Figueiredo, B. Macq, B. Mory, L. Nunes, J. Orri, M. J. Trigueiros, and A. Violante. Video-Based Multi-Agent Surveillance System. In *Proceedings of the 2000 Intelligent Vehicles Conference*, Oct 2000.

[9] M. Yokoo, E.H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):Sep/Oct, 1998.

[10] M. Bramberger, B. Rinner, and H. Schwabach. Resource-aware dynamic task-allocation in clusters of embedded smart cameras by mobile agents. In *Proceedings of the IEE Intenational Workshop on Intelligent Environments*. IEE, June 2005. to appear.

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

[12] C.S. Regazzoni, V. Ramesh, and G.L. Foresti. Introduction of the special issue. *Proceedings of the IEEE*, 89(10), Oct 2001.