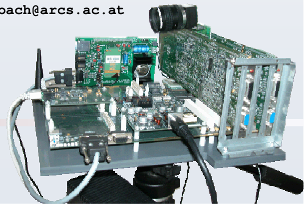
A Mobile Agent-based System for Dynamic Task Allocation in Clusters of Embedded Smart Cameras

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

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

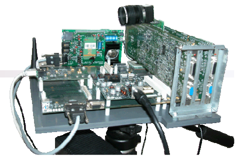
[2] Video & Safety Technology
ARC Seibersdorf Research
AUSTRIA
helmut.schwabach@arcs.ac.at

May 2o, 2oo5 Hamburg, Germany



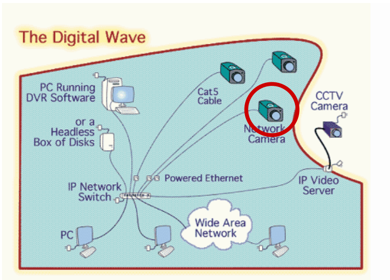
Overview

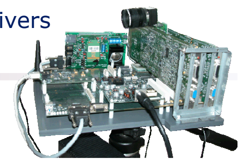
- Introduction
- The Smart Camera
- Surveillance System Architecture
- Task Allocation
- Implementation & Experiments
- Outlook



Surveillance Systems

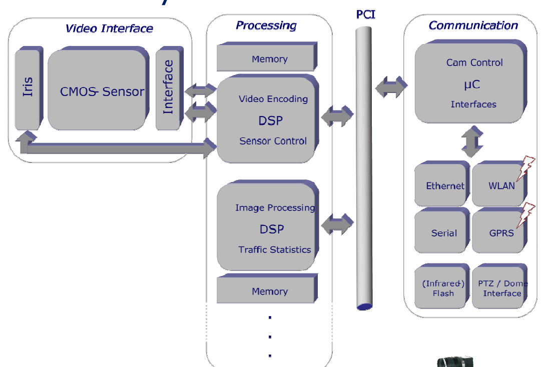
- 1st and 2nd generation
  - → Primarily analog frontends
  - → Backend systems are digital
- 3rd generation
  - → All-digital systems
- 3+ generation
  - → Smart cameras
  - → Surveillance tasks run on-site on smart cameras
    - Video compression
    - Accident detection
    - Stationary vehicles (tunnels)
    - Traffic statistics
    - Wrong-way drivers
    - Tracking

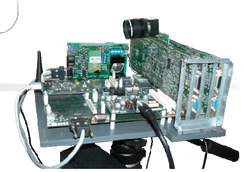




The Smart Camera - Hardware

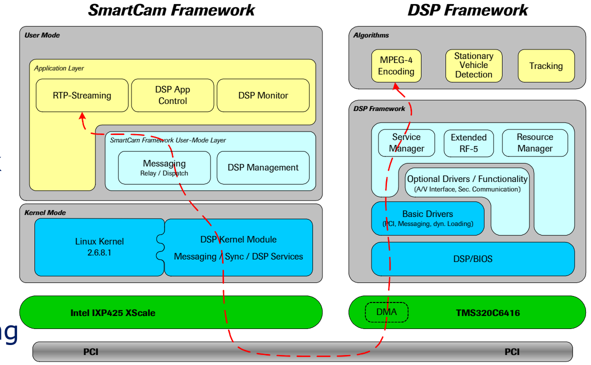
- Heterogeneous multi-processor system
  - →CMOS image-sensor
  - →TI TMS320C64x DSPs
  - → Intel IXP4xx network processor
- Scalable HW design
  - → Up to 10 DSPs
- Interfaces
  - → 100Mbit Ethernet, USB, Serial, GPRS, WLAN

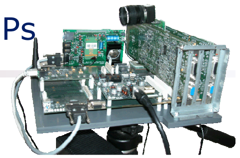




The Smart Camera - Software

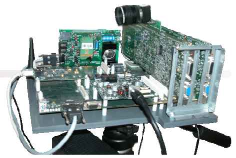
- ❑ Two-Fold Software Framework
  - → DSP-Framework
  - → SmartCam-Framework on Network Processor
- ❑ Provides services
  - → Data Management
  - → Performance monitoring
  - → Synchronization
  - → Message-based communication DSP⇔DSP / DSP⇔XScale
- ❑ Dynamic loading of modules (binaries) on DSPs





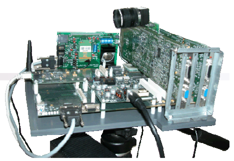
Surveillance System Architecture

- ❑ Improved surveillance due to smart cameras
  - → Decentralized, distributed surveillance
  - → Autonomous allocation & reconfiguration of tasks
- ❑ Functional groups of smart cameras
  - → Limited performance and resources
    - ■ Real-Time Constraints
  - → Distribute sets of surveillance tasks to groups of smart cameras
    - ■ *"Surveillance Clusters"*



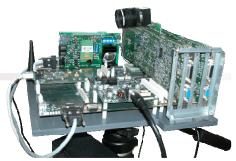
Task Allocation 1

- ❑ Determine optimal allocation of tasks to smart cameras
  - → According to required resources – primarily of the DSPs
    - ■ CPU, MEM (on-/off-chip), DMA (channels, IRQs), transferred data
  - → Overloading of a resource yields in violation of real-time constraints
- ❑ Reallocation is required **online** due to
  - → Changed resource requirements of tasks (raised by software)
  - → Changed resource availability of a camera (raised by hardware)
  - → Added/removed cameras or tasks
  - → Tracking tasks migrates between hosts



Task Allocation 2

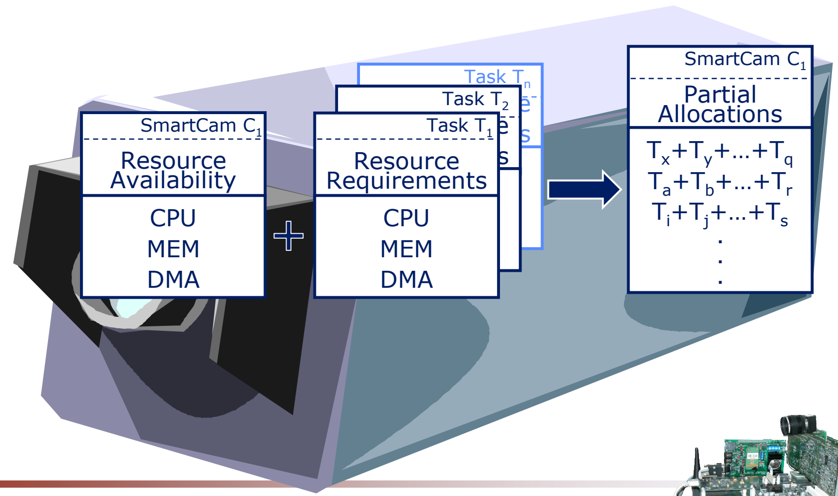
- ❑ Is a constraint-satisfaction-problem (CSP)
  - ■ Variables → Tasks $\{T_1,\dots,T_n\}$
  - ■ Domain → Cameras $\{H_1,\dots,H_m\}$
  - ■ Constraints → Allocation of all tasks
    No camera must be overloaded
  - → Complexity: $O(m^n)$
  - → Resources considered in constraints
    - ■ CPU, MEM (on-/off-chip), DMA (channels, IRQ), and transferred data
  - → State-of-the-art approaches
    - ■ provide **one** solution
    - ■ require a central host



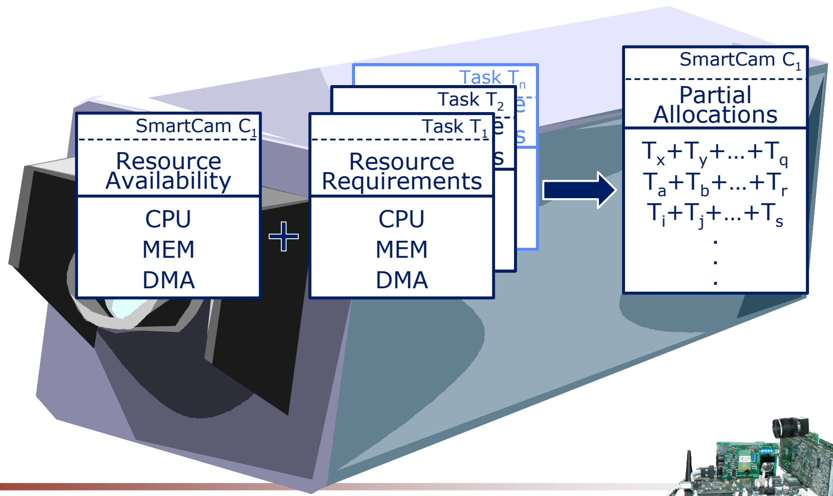
Task Allocation 2

- ❑ Is a constraint-satisfaction-problem (CSP)
  - → Variables, Domain, and Constraints
  - → Standard approaches provide **one** solution, require a central host
- ❑ Splitting the domain → Create a distributed CSP (DCSP)
  - → Determine partial allocation (solutions of sub-CSP) in parallel
  - → Prune set of partial allocations using a cost function
  - → Merge partial allocations
  - → Exploit distributed architecture

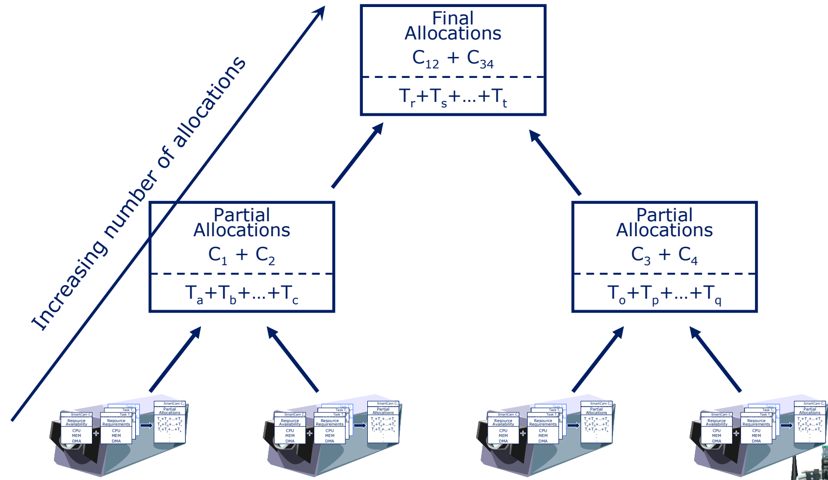
Determine Partial Allocations



Determine Partial Allocations

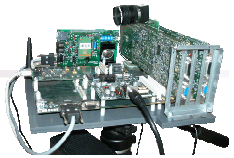


Determine Allocations



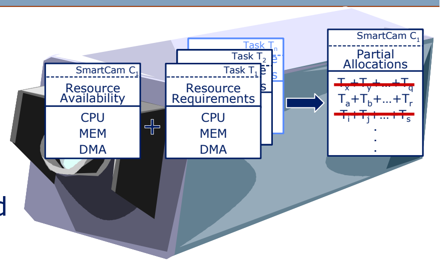
Cost Function

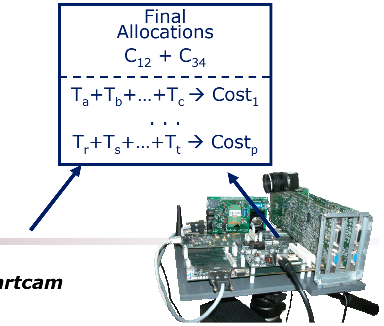
- ❑ Goal
  - → Reduce number of allocations (early pruning)
  - → Choose optimal allocation
- ❑ Five cost classes
  - → Resource Costs
  - → Migration Costs
  - → Quality-of-Service Costs
  - → Data-Transfer Costs
  - → Affinity Costs
- ❑ Costs are based on availability of resource



Integrating the Cost Function

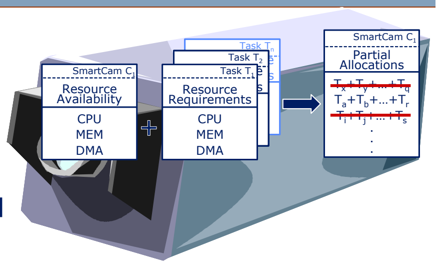
- ❑ Remove allocations
  - → Calculate Cost for each task on every camera
  - → Remove allocations which exceed an adaptive threshold
- ❑ Find optimal solution
  - → Determine the cost of each feasible allocation
  - → Choose the allocation which provides the lowest cost value

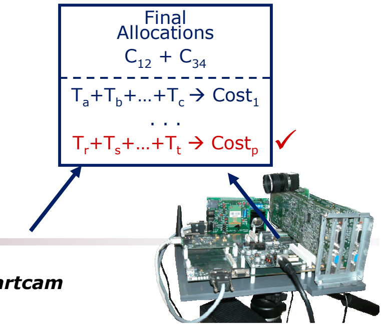




Integrating the Cost Function 1

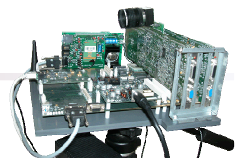
- ❑ Remove allocations
  - → Calculate Cost for each task on every camera
  - → Remove allocations which exceed an adaptive threshold
- ❑ Find optimal solution
  - → Determine the cost of each feasible allocation
  - → Choose the allocation which provides the lowest cost value





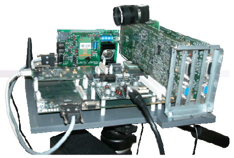
Integrating the Cost Function 2

- ❑ Cost integration reduces number of allocations
  - → Threshold is set according to lowest cost
    - ▪ A minimum number of feasible allocations has to be accepted
- ❑ Reallocation Scenarios
  - → Increased system load
    - ▪ Prune set of feasible allocations
    - ▪ Fast results required
  - → Reduced system load
    - ▪ Determine new set of feasible allocations
    - ▪ Timing to achieve results is relaxed



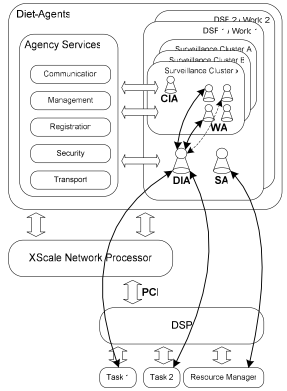
Implementation — System Setup

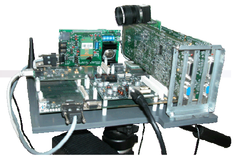
- ❑ Operational environment
  - → Network processor running Linux
  - → DSPs running DSP/BIOS (TI)
  - → SmartCam/DSP-Framework
- ❑ Mobile agent-based implementation
  - → Diet-Agents agent system
  - → JamVM as Java virtual machine
  - → Tight interface to DSPs



Implementation — Agent System

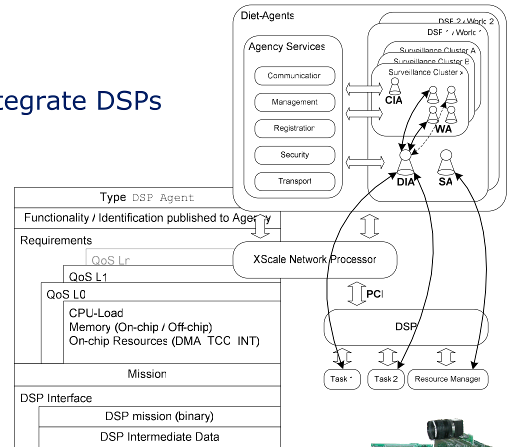
- ❑ DSP-Agencies
  - → Enhanced agencies to integrate DSPs
  - → Including
    - DSP Interaction Agent — DIA
    - Cluster Information Agent — CIA
    - System Agent — SA
    - Worker Agents — WA

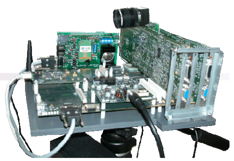




Implementation — Agent System

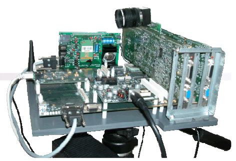
- ❑ DSP-Agencies
  - → Enhanced agencies to integrate DSPs
  - → Including
    - DIA, CIA, SA, and WA
- ❑ DSP-Agents
  - → DSP module (binary)
  - → Several QoS levels
  - → Store intermediate data





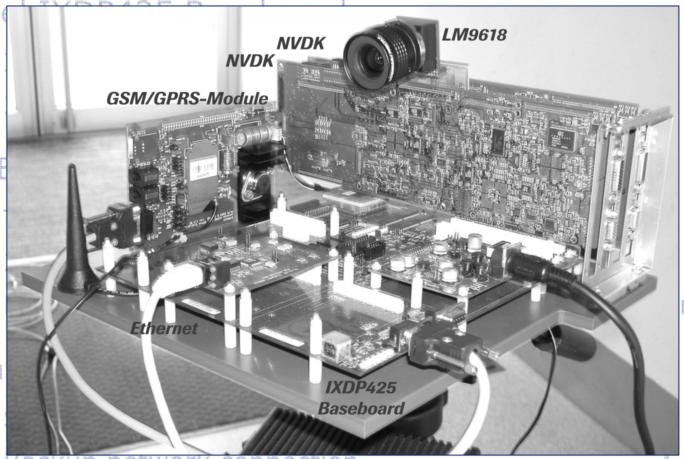
Prototype

- ❑ Intel IXDP425 Baseboard
  - → Intel IXP425 Network Processor @ 533 MHz
  - → 256 MB SDRAM, 2x 100BaseT Ethernet, PCI
- ❑ ATEME IEKC64 (2x)
  - → Texas Instruments TMS320C6416 @ 600 MHz
  - → 264 MB SDRAM
  - → National Semiconductor LM9618 CMOS image sensor
- ❑ WaveCom GSM/GPRS module
  - → Supports Class 10 data transfers
  - → Backup network connection



Prototype

- Inte
  - →
  - →
- ATE
  - →
  - →
  - → r
- Wa
  - →
  - → Backup network connection



Experimental Results

- Experiments conducted on
  - → a prototype of the smart camera,
  - → two 1 GHz Pentium-III PCs

Case	Executed by	PC JDK 1.4	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	4.852 ms	21.363 ms	2.360 ms
2b	Merge Solutions (3 Ag)	9 ms	5 ms	31 ms	4 ms
3a	Overall allocation determination (6 Ag)	786 ms	4.866 ms	21.442 ms	2.373 ms
3b	Overall allocation determination (3 Ag)	23 ms	12 ms	86 ms	13 ms
4	Reusing allocations (6 Ag)	14 ms	32 ms	172 ms	26 ms

Outlook

- Evaluate more complex scenarios
- Real-world tests
- Integrate learning agents to optimize the reallocation behavior
- Performance optimizations

?

http://www.iti.tugraz.at/smartcam