EMBEDDED MIDDLEWARE ON DISTRIBUTED SMART CAMERAS

Bernhard Rinner[1], Milan Jovanovic[2], Markus Quaritsch[2]

[1]Pervasive Computing Group
Institute of Networked and Embedded Systems
Klagenfurt University, AUSTRIA

[2]Institute for Technical Informatics
Graz University of Technology, AUSTRIA

Agenda

1. (Distributed) Smart Cameras
   - introduction
   - challenges in application development
2. System-level software for smart camera networks
   - SmartCam HW/SW architecture
   - SmartCam middleware services
3. Case study on multi-camera tracking
   - autonomous camera handover
4. Conclusion

Introduction

- Smart cameras
  - combine image sensing, processing and communication on single embedded device
  - perform (high-level) image analysis onboard
  - collaborate in networks of cameras

- These smart image sensors are very attractive for various apps, eg.
  - surveillance & security
  - traffic monitoring
  - health care
  - entertainment

Distributed Smart Cameras (DSCs)

- May help to overcome some hard problems, eg
  - occlusion and low “pixels-on-target” by exploiting multiple views
  - high communication bandwidth by data abstraction and local processing
  - limits in real-time behavior by avoiding round-trip delays
  - failures of individual cameras by exploiting redundancy
- Challenges for DSCs
  - architecture, network
  - collaboration
  - design process for distributed, embedded vision sensors

Application Development for DSCs

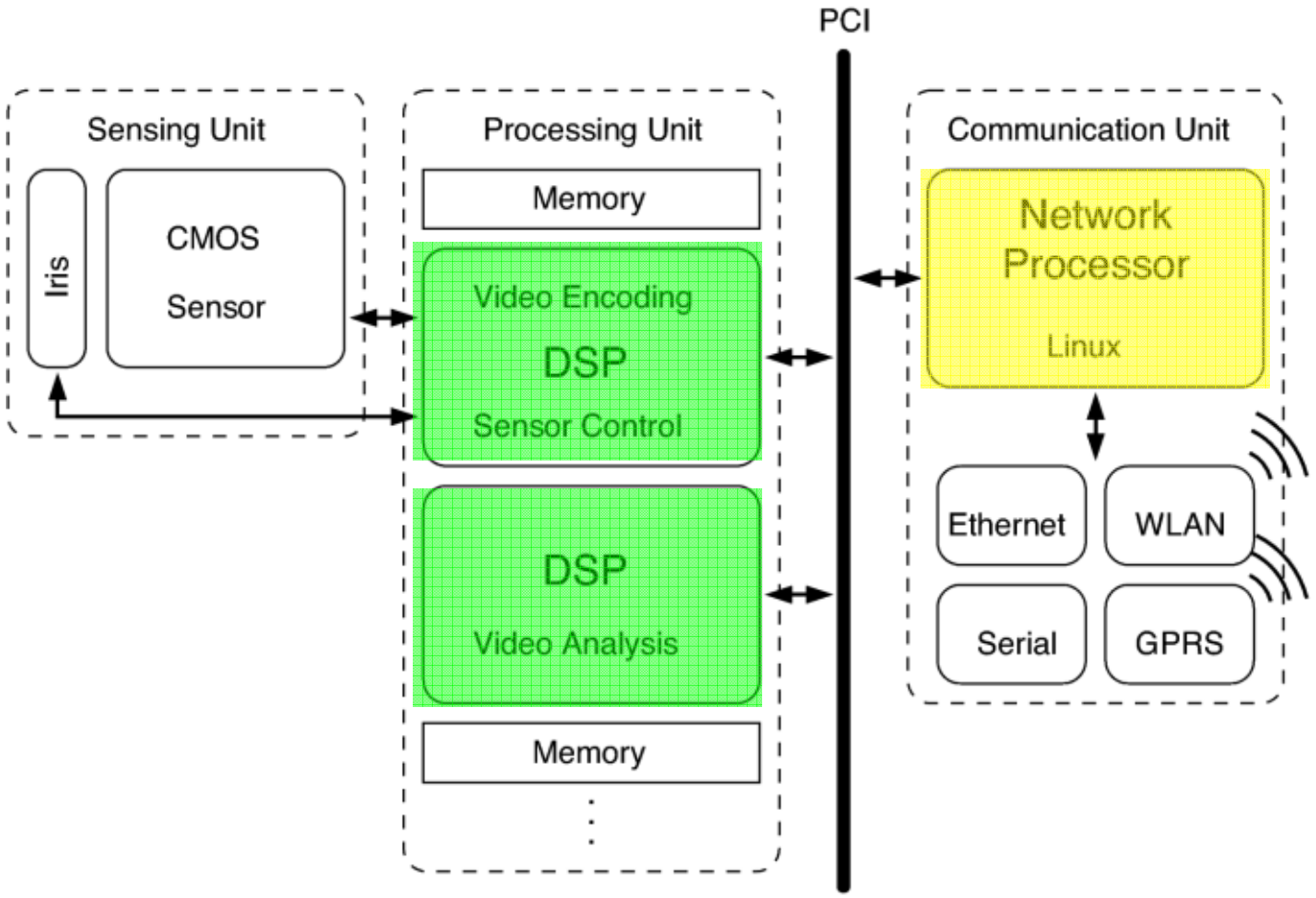
- Much more difficult than single-camera applications
  - collaboration requires communication & control
  - dynamic network (QoS adaptation, scalability, ad-hoc networking)
  - distributed computing (concurrent threads of control)
- Middleware would help throughout the development …
  - design
  - deployment
  - operation/reconfiguration
- … but available MW does not fit
  - general “CORBA-like” MW are too heavy
  - WSN MW focus on ad-hoc networking, power awareness

Middleware Services we'd like to have

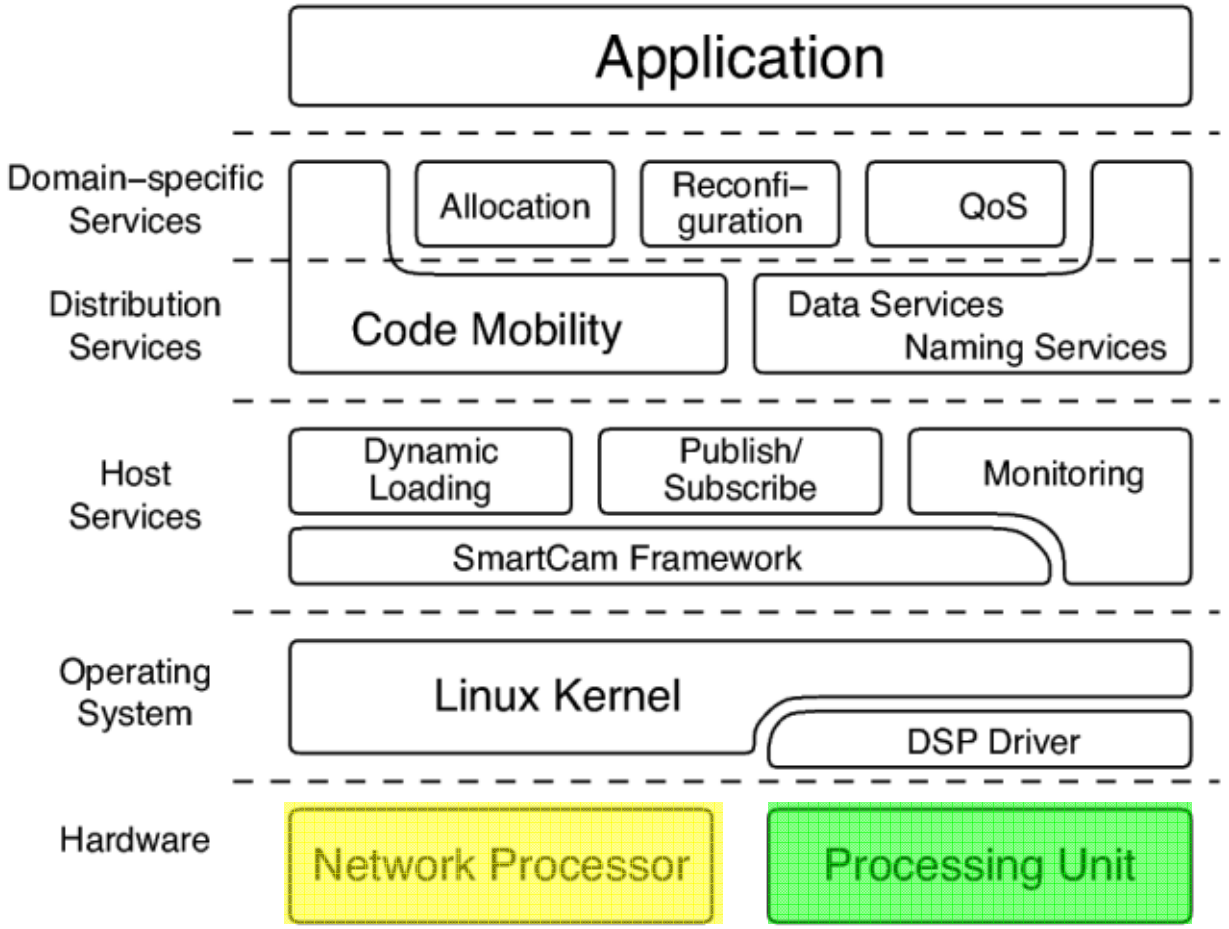
- Deployment services
  - help to initiate the DSC network (loading, allocation, update …)
- Operational services
  - responsible for efficient coordination and configuration
- Networking services
  - establish transparent communication, data transfer and resource management
- Application-specific services
  - specific to image processing applications, e.g., calibration, registration, QoS adaptation
- On embedded, resource-limited, image networks

Our SmartCam Architecture

[IEEE Computer 2/2006]



SmartCam Middleware



SmartCam MW: Host Services

- Dynamic Loading
  - change functionality on SmartCam during runtime (“DLL-like”)
  - basis for many other services
- Publish/Subscribe
  - provides transparent (inter-processor) communication on SmartCam
- Monitoring
  - observes dynamic resource utilization on SmartCam
  - focuses on critical resources on embedded platform (CPU, memory, DMA, PCI bus)

SmartCam MW: Distribution Services

- Mobile agent system (MAS)
  - foundation for distributed applications
  - provides mechanisms for code and data migration among SmartCams
  - abstracts image processing as tasks (executed on DSP)
- Mobile agent
  - contains application logic and controls image processing tasks
  - improves scalability
- Data and naming services
  - distinguish between control messages and image data

SmartCam MW: Domain-spec. Services

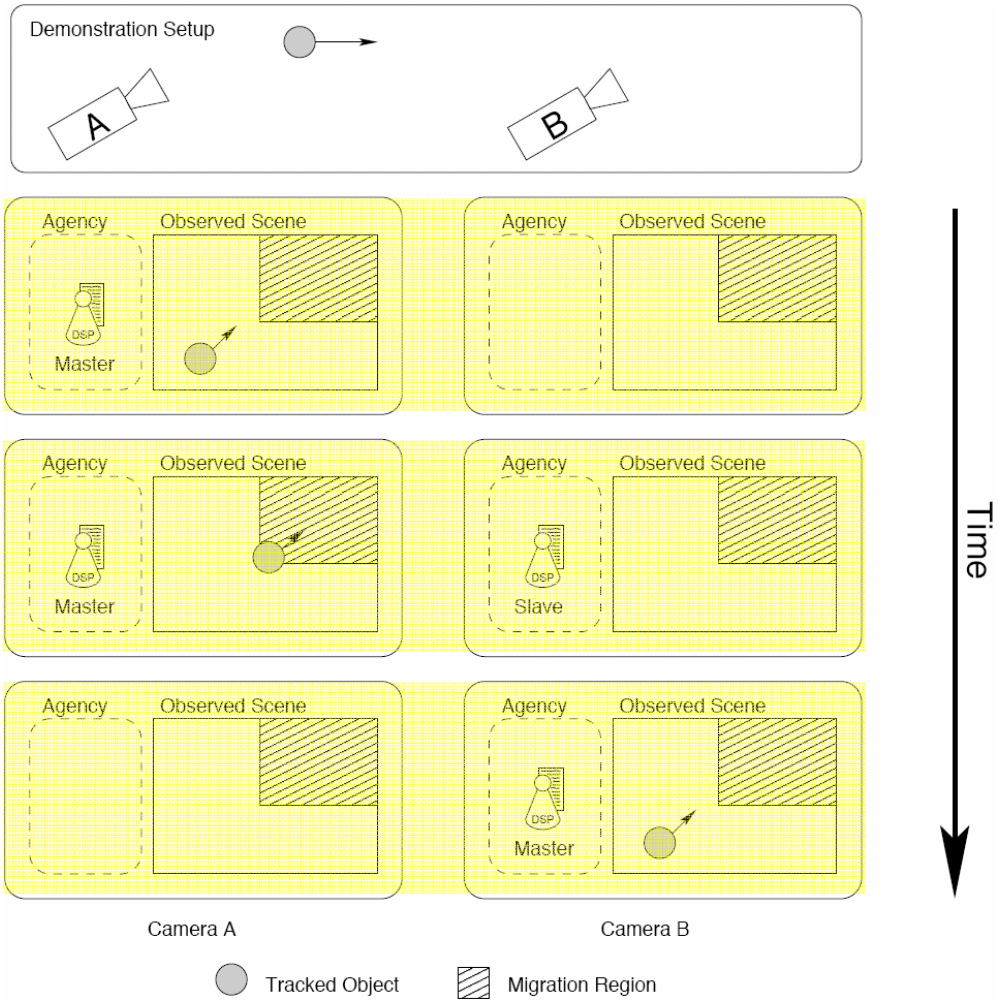
- Allocation
  - assigns image tasks to individual cameras
- Dynamic reconfiguration
  - modifies allocation and functionality (tasks) during runtime
  - requires reasoning about current configuration and resource utilization
- QoS adaptation
  - eg, combined power and QoS

Autonomous Multi-Camera Tracking

[EURASIP JES 1/2007]

- Develop autonomous multi-camera tracking
  - on embedded smart cameras
  - using an arbitrary tracking algorithm
  - without central coordination
- Tracking algorithm
  - standard (“color-based”) CamShift tracker
  - tracker encapsulated in mobile agent
  - one tracking agent for each tracked object/person
- Camera handover
  - based on pre-defined “migration region” in camera’s FOV
  - tracking agent autonomously migrates to “next” camera(s)

Multi-Camera Handover Strategy



Master/Slave handover

1. camera A tracks object
2. whenever object enters migration region tracking agent is cloned on “next” camera (slave)
3. slave starts tracking when slave identifies object master gets terminated

Tracker initialization

- color histogram a initialization data

Multi-Camera Tracking Demo

Supporting Middleware Services

- Abstraction of image processing
  - Mobile Agent: application logic
  - Tracking algorithm: identify position of object
- Code Migration and dynamic loading
  - Tracker is executed only on the camera observing the object
- Transparent messaging
  - Communication between neighboring cameras
  - Communication between tracking agent and visualizer

Conclusion

- Distributed Smart Cameras are likely to become an enabling technology for various applications
- Exploit and advance methods from related fields
  - vision, sensor networks, embedded systems, distributed computing, multimedia, …
- Open Research Challenges
  - architecture & networking
  - collaborative multi-camera vision, sensor fusion
  - (semi-)automatic deployment, eg. calibration, synchronization
  - design support, tools etc.
- Most of these challenges have a strong influence on system-level software (middleware)

Acknowledgements

This work has taken place at Graz University of Technology.
Further information available at

www.iti.tugraz.at/smartcam

This research has been partially supported by the Austrian Promotion Agency under grant 810072.

See you at ICDSC-07 (www.icdsc.org)