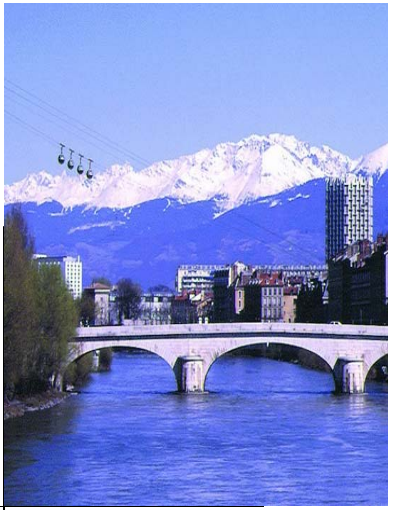
Smart Cameras and Visual Sensor Networks



EMBEDDED SYSTEMS WEEK

Part 4

Visual Sensor Networks

Bernhard Rinner

ALPEN-ADRIA UNIVERSITÄT KLAGENFURT

LASMEA

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

Le2i

cnrs dépasser les frontières

Tutorial Agenda

1. Introduction
2. Smart imager and smart cameras
3. Embedded image processing
    - Heterogeneous Platforms (FPGAs, DSPs ...)
    - Dedicated Processors (GPU and cell)
4. **Visual Sensor Networks**
    - Distributed Sensing and Processing
5. Conclusion
    - Research Challenges

Distributed Smart Cameras

Smart Cameras collaborate

- Connect autonomous cameras in a network
  - exploit smart cameras' capabilities (eg. avoid raw data transfer)
  - relax centralized/hierarchical structure of MC networks
  - introduce dynamic configuration (structure and functionality)

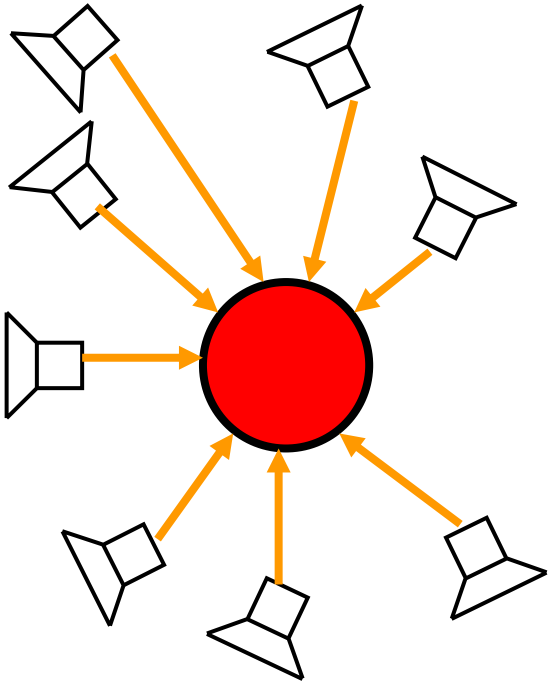
- Challenges for distributing sensing & processing
  - camera selection and placement
  - calibration & synchronization
  - data distribution and control, protocols and middleware
  - distributed computer vision (distributed signal processing)
  - real-time, energy-awareness, …

(Potential) Advantages of DSC

- Scalability
  - no central server as bottleneck
- Real-time capabilities
  - Short round-trip times; “active vision”
- Reliability
  - High degree of redundancy
- Energy and Data distribution
  - Reduced requirements for infrastructure; easier deployment?
- Sensor coverage
  - Many (cheap) sensors closer at “target”; improved SNR
- …

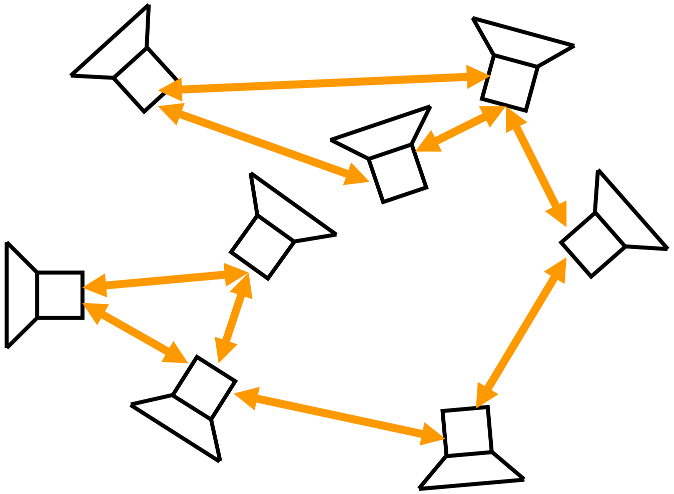
Networking

Traditional Camera Networks



Cameras stream images/
videos to „server“

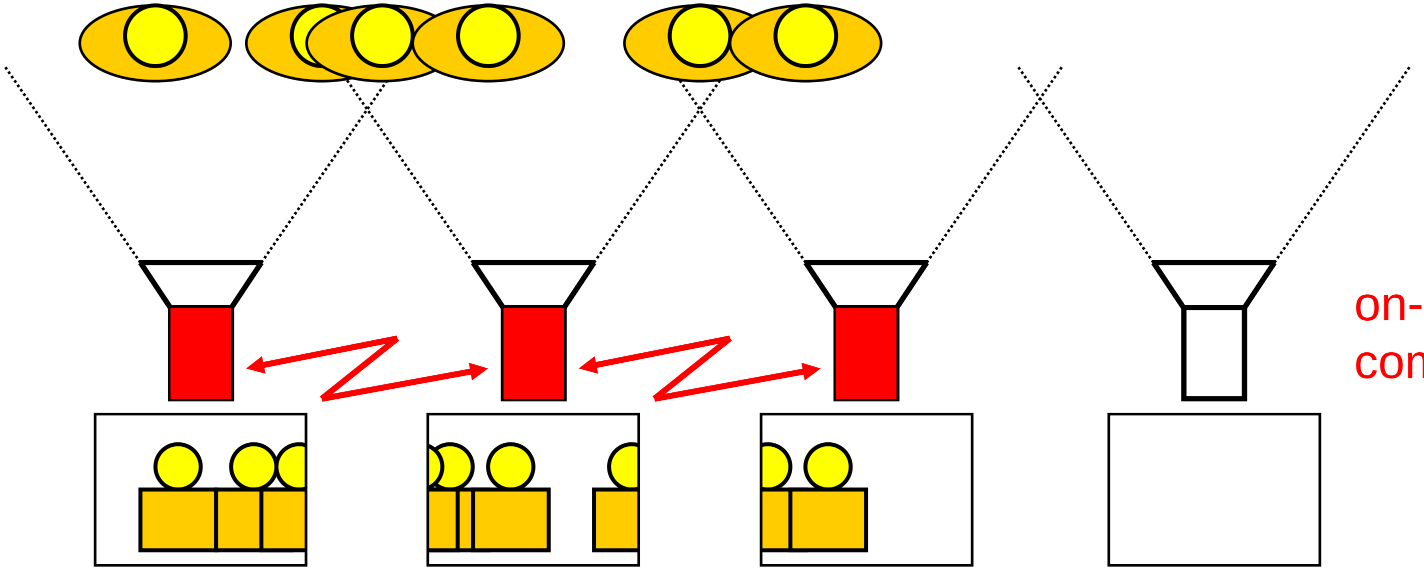
Smart Camera Networks



Cameras collaborate directly
(spontaneous, p2p, ad-hoc)

Distributed Processing in Network

- Example: autonomous tracking of mobile objects among multiple cameras



on-board analysis
communication

- Computation follows (physical) object
  - requires spontaneous communication; distributed control & data

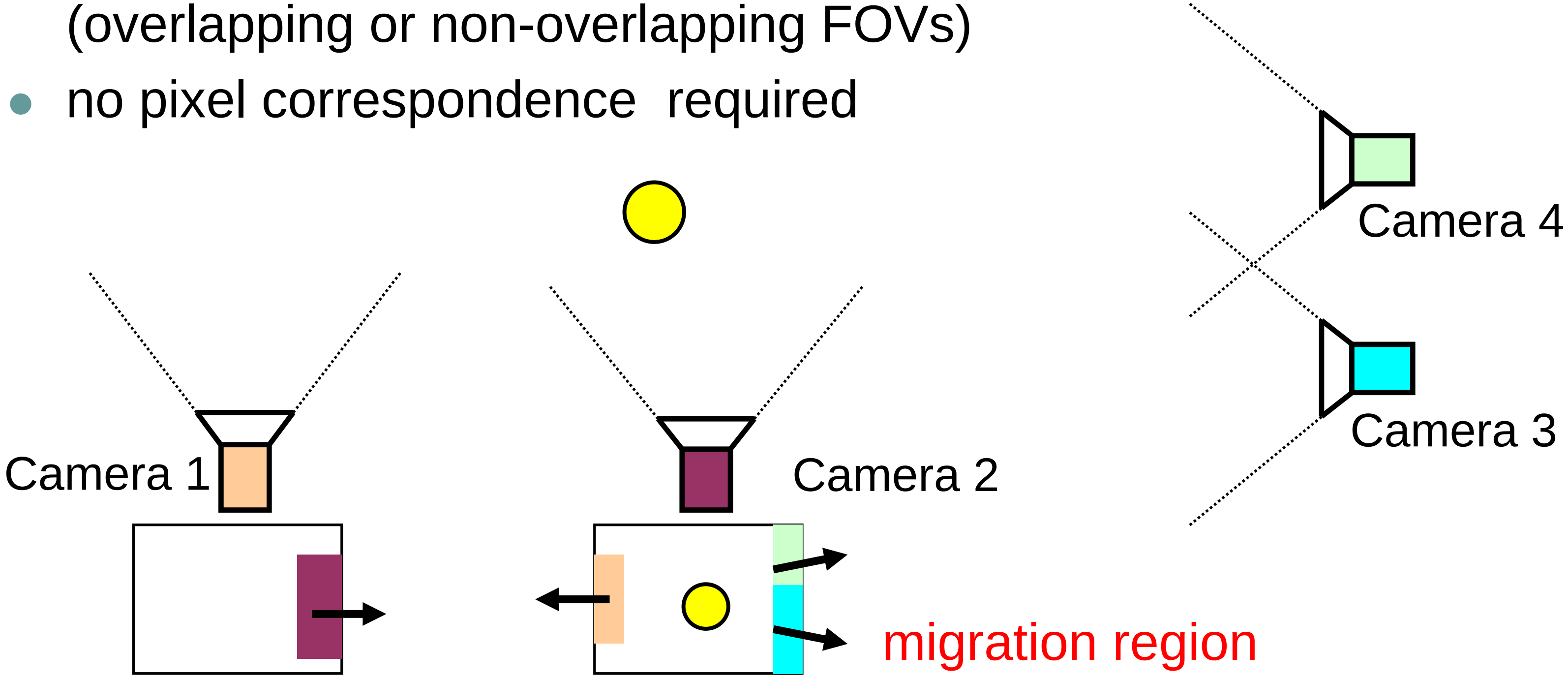
Autonomous Multi-Camera Tracking

[] Quaritsch et al., Autonomous Multicamera Tracking on Embedded Smart Cameras EURASIP JES 1/2007]

- Assumptions for multi-camera tracking
  - implement on distributed embedded smart cameras
  - avoid accurate camera calibration
  - do not rely on central coordination
- Important design questions
  - What (single-camera) tracking algorithm to use?
  - How to coordinate the cameras?
    i.e., distributed control, exploit locality
  - How to hand over tracking from one camera to next?
- Treat questions independently
  - standard (“color-based”) CamShift tracker
  - focus on hand over strategy

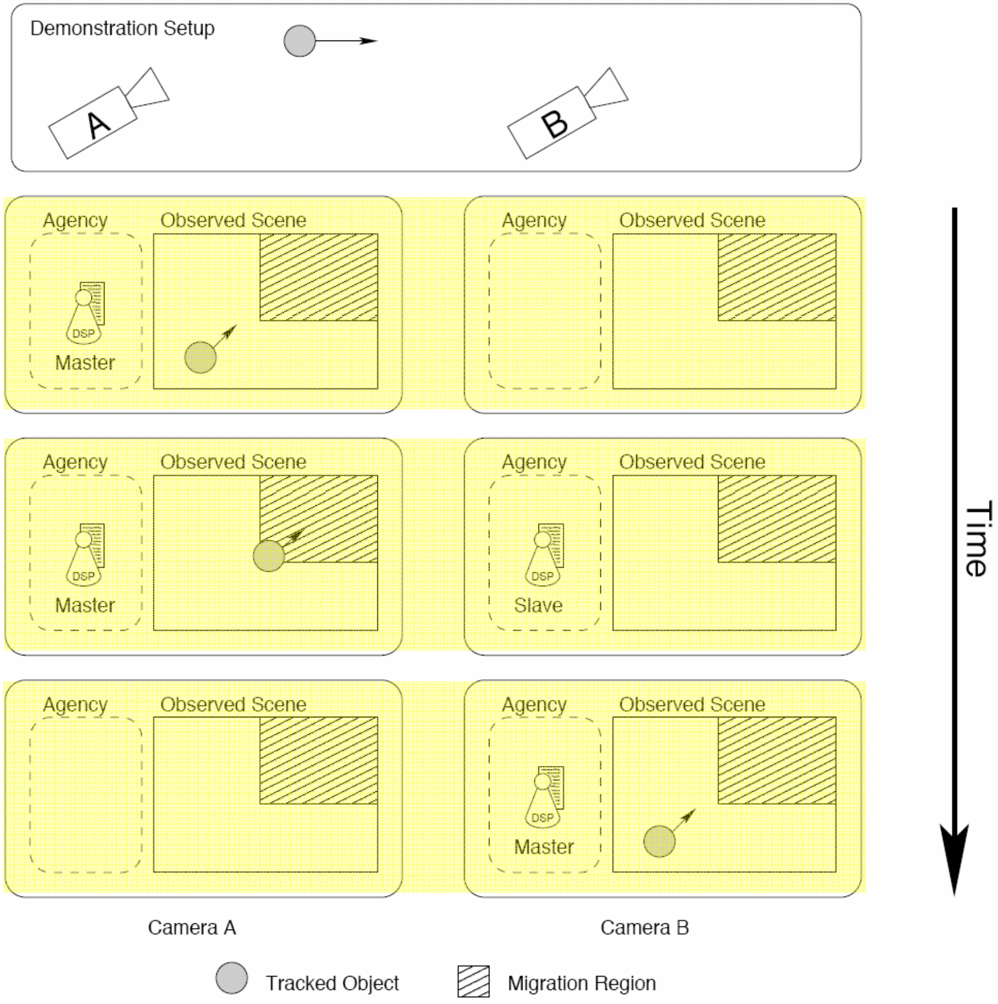
Spatial Relation among Cameras

- Camera neighborhood relation
  - important for determining “next camera(s)”
  - based on pre-defined “migration region” in camera’s FOV (overlapping or non-overlapping FOVs)
  - no pixel correspondence required



migration region with motion vector

Multi-Camera Handover Protocol



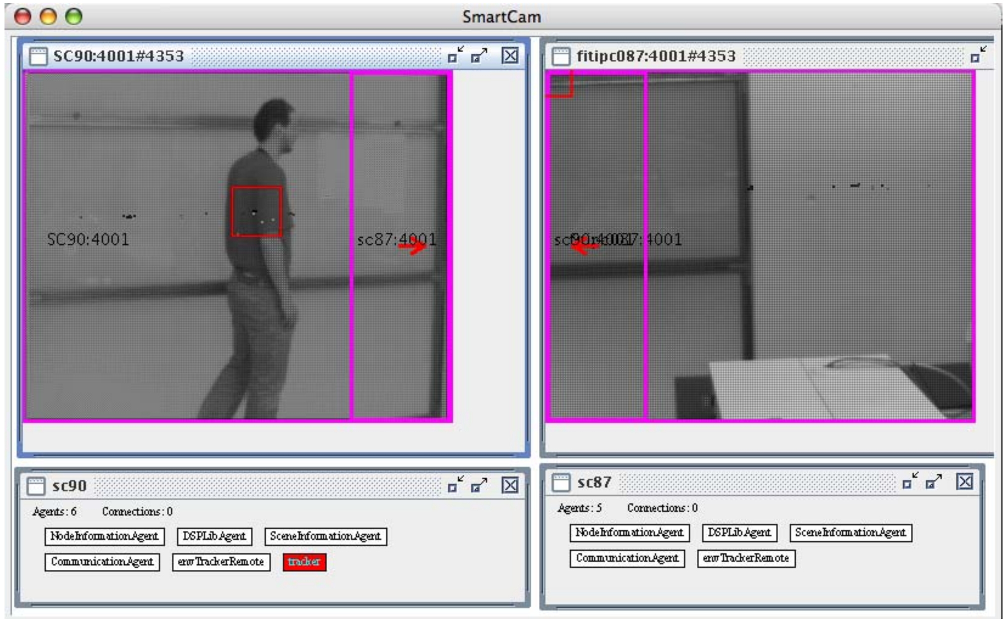
Master/Slave handover

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

Implementation & Results



Visualization

- migration region (magenta)
- tracked object (red rectangle)

- tracking agent (red box)

Code size	15 kB
Memory requirement	300 kB
Internal state	256 B
Init color histogram	< 10 ms
Identify object	< 1ms

CamShift (single camera)

Loading dynamic executable	8 ms
Initializing tracking algorithm	250 ms
Creating slave on next camera	18 ms
Reinitializing tracker on slave	2 ms
Total	**278 ms**

Multi-camera performance

Toward Visual Sensor Networks

Characteristics of VSN

- In-network image sensing & processing
- Data streaming as well as eventing
- Resource limitations (power, processing, bandwidth …)
- Autonomy & service-orientation
- Ease of deployment

Multi-view Calibration

- Standard calibration methods are tedious
  - performed offline
  - require physical appearance of reference objects
  - limited scalability in large networks
- Automatic methods are necessary in visual sensor networks
  - Limited knowledge about initial position and orientation of cameras
  - Mobility of camera nodes
  - No human/expert available
- Estimation methods
  - Vision Graph
  - Calibration of neighboring cameras

Estimating the Vision Graph

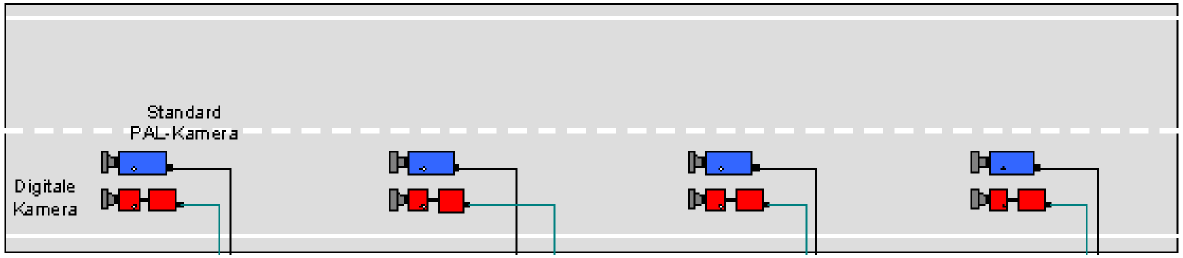
- Identify cameras with overlapping FOV
  - Also referred to as topology of the network
  - Exploit spatiotemporal tracks of moving objects
  - Often assume common ground plan
- Determine the “area” of overlap
  - Compute offline (if cameras are fixed)
  - Model camera projection (if parameters are known)

Multi-Camera Calibration

- Focus on calibration only among neighboring cameras
  - Determine reliable corresponding points
  - Estimate parameters of neighboring cameras
- Distributed calibration algorithms
  - Avoid transferring images
- Exploit information about position and orientation of cameras
  - Often available in sensor networks
  - Calibration not exclusively based on captured images

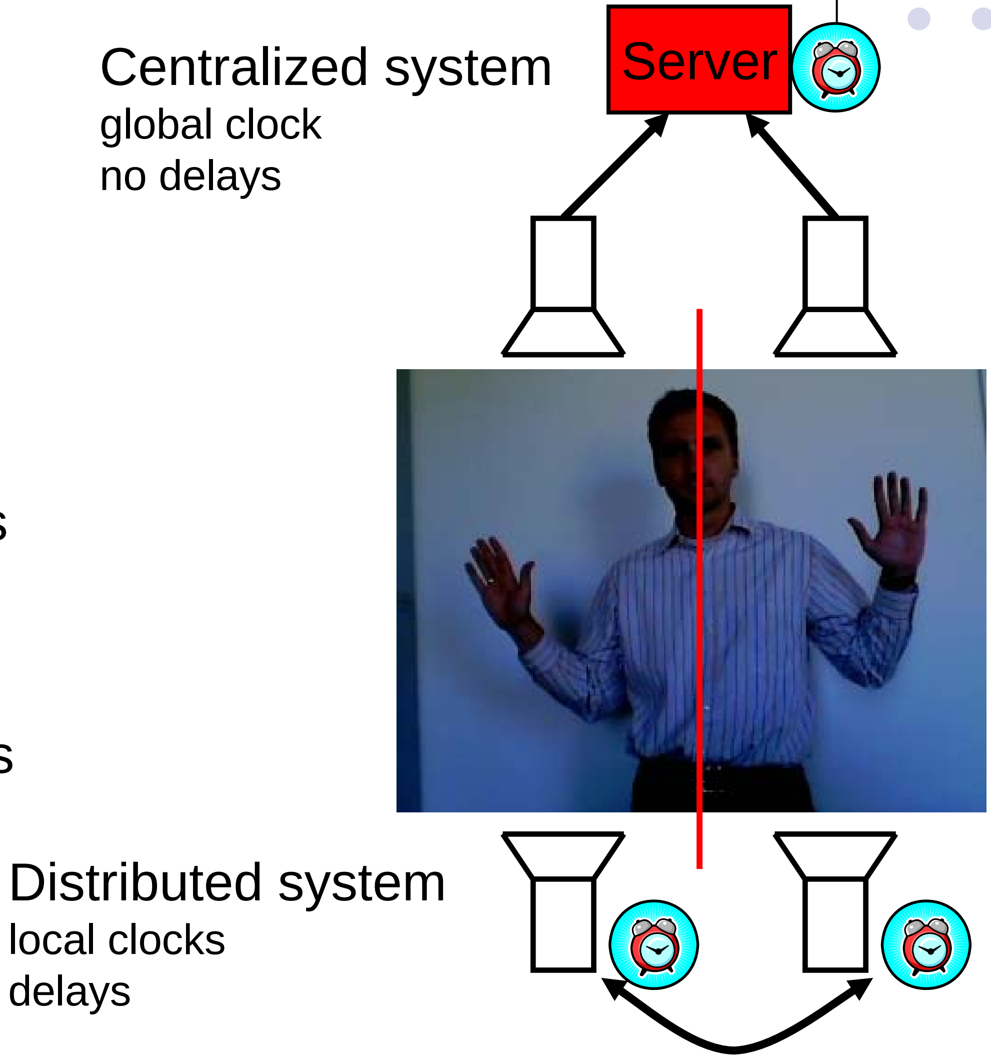
Multi-Camera Calibration (2)

- Relaxing calibration requirements
  - What to do when there is no overlap (cp. epipolar geometry)?
  - Accurate calibration not required for some applications
- Example: Camera Hand-off in MC-Tracking
- Camera network topology
  - Applications pose strong constraints (traffic, buildings etc.)



Temporal Calibration / Synchronization

- Cameras need to be synchronized for distributed analysis. Problems
  - No global clock
  - Communication delays (unknown, jittering)
- Example
  - Fusing individual views from two cameras



Centralized system
global clock
no delays

Distributed system
local clocks
delays

Synchronization

- Synchronization accuracy
  - Depends on application and on level of local processing
  - Often “frame-accurate” synchronization sufficient
- Apply methods from sensor networks
  - Distributed and resource-aware

Resource-Awareness

- Visual sensor nodes have limited resources
  - Embedded platform
- Critical resources
  - Sensing
  - Computing and memory capacity
  - Communication
  - Power
- Manage resources effectively
  - Switch off unused components: dynamic power management
  - Trade performance, quality, time etc: reconfiguration

Quality of Service

- Cameras and VSN provide different quality levels
- Low-level QoS
    - Image resolution
    - Communication bandwidth, delay
- More abstract QoS
    - Different detection performances

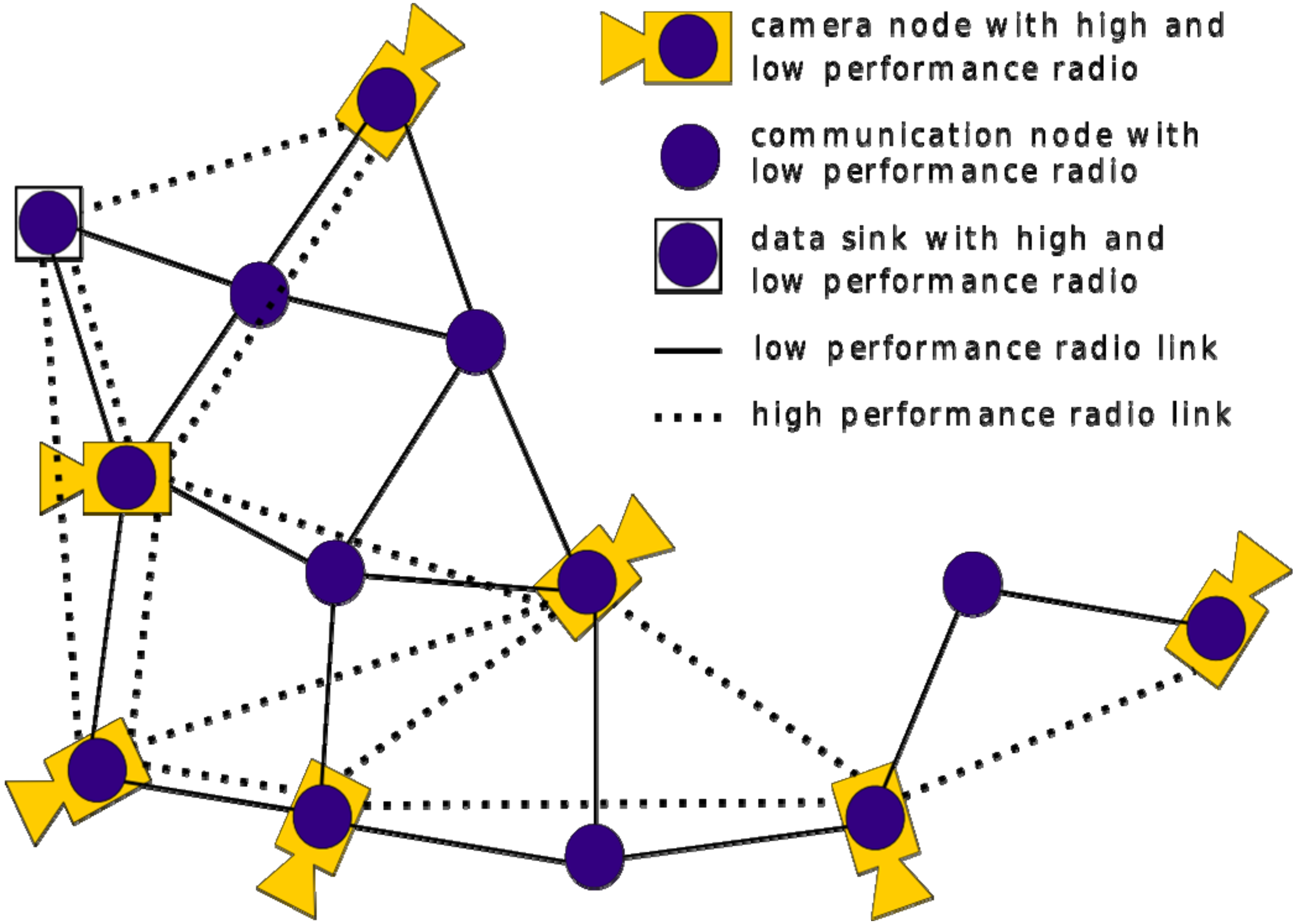
Applications & Case Studies

Pervasive Smart Camera Network

- Tradeoff among bandwidth, power consumption and streaming requirements in VSN
- One approach: dual radio networks
- Equip (some) nodes with two radios: low-bandwidth & high-bandwidth
- Use low-bandwidth radio for normal operation
  - coordination, eventing,
  - transfer of low-resolution (still) images
- Use high-bandwidth radio for streaming

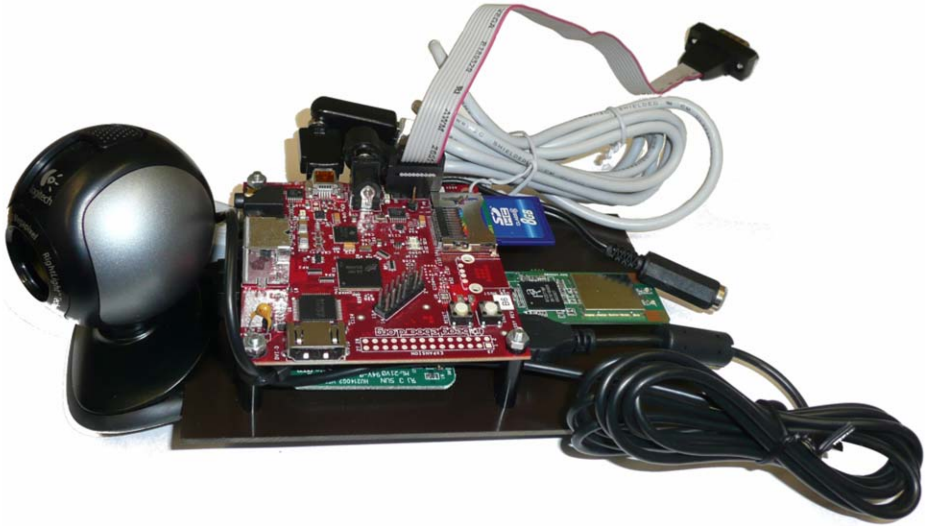
[] Winkler, Rinner. Pervasive Smart Camera Networks exploiting heterogeneous wireless Channels. In Proc. PerCom 2009

PSC Network Architecture



PSC Camera Network

- Visual Sensor Network Platform
- Sensor Nodes
  - Embedded board with USB connected peripherals
  - TI OMAP3530 processor: ARM Cortex A8 @ 600MHz, TI C64x DSP @430MHz
  - 128MB RAM, 256MB Flash
  - SD-Card, USB, DVI, audio-i



PSC Demo: Tracking

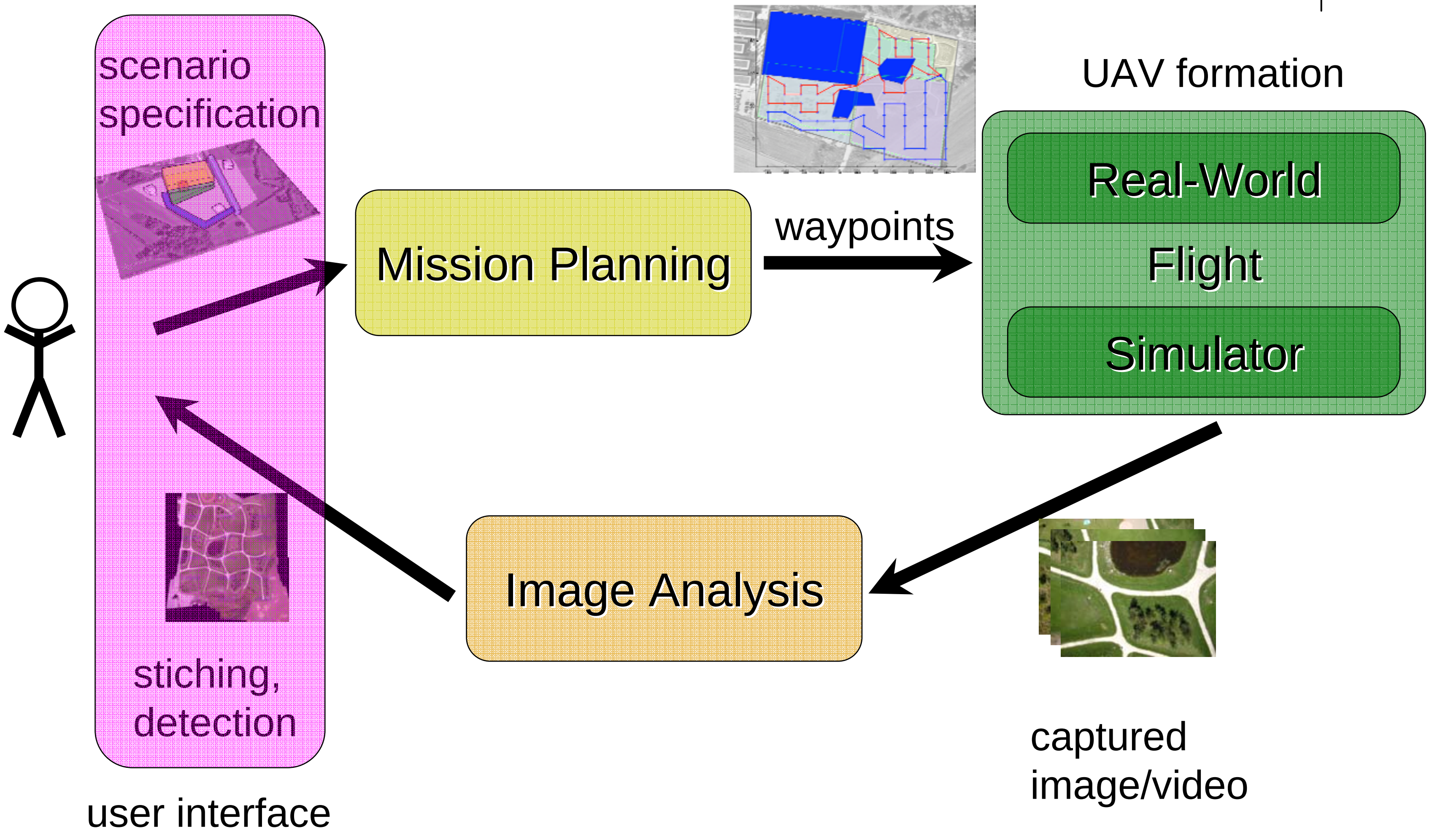
- Demonstrate tracking by using only low-bandwidth radio
  - initially transfer background image
  - perform tracking onboard
  - transfer tracking result (bounding box);
    8 bytes/frame

Collaborative Microdrones

- UAVs for disaster management
  - deploy a group of small UAVs for disaster management applications
  - fly over the area of interest in structured way (formations)
  - sense the environment
  - analyze the sensor data (image stiching, object detection etc.)
- Provide “bird’s eye view” to special task forces in real-time
- Support high autonomy and an intuitive user interface

[] Quaritsch et al., Collaborative Microdrones: Applications and Research Challenges. In Proc. Autonomics 2008

High-level “Processing Loop”



scenario specification

Mission Planning

waypoints

UAV formation

Real-World

Flight

Simulator

Image Analysis

stiching, detection

captured image/video

user interface

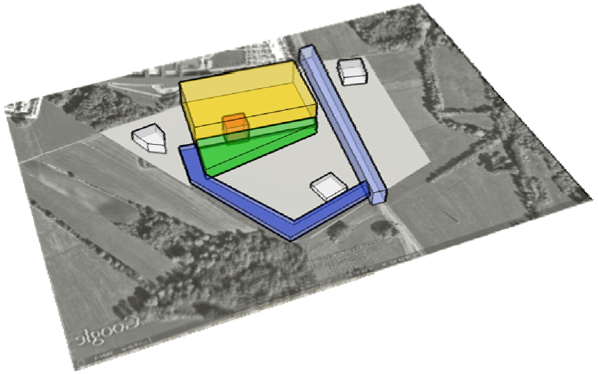
UAV Platform

[www.microdrones.com]

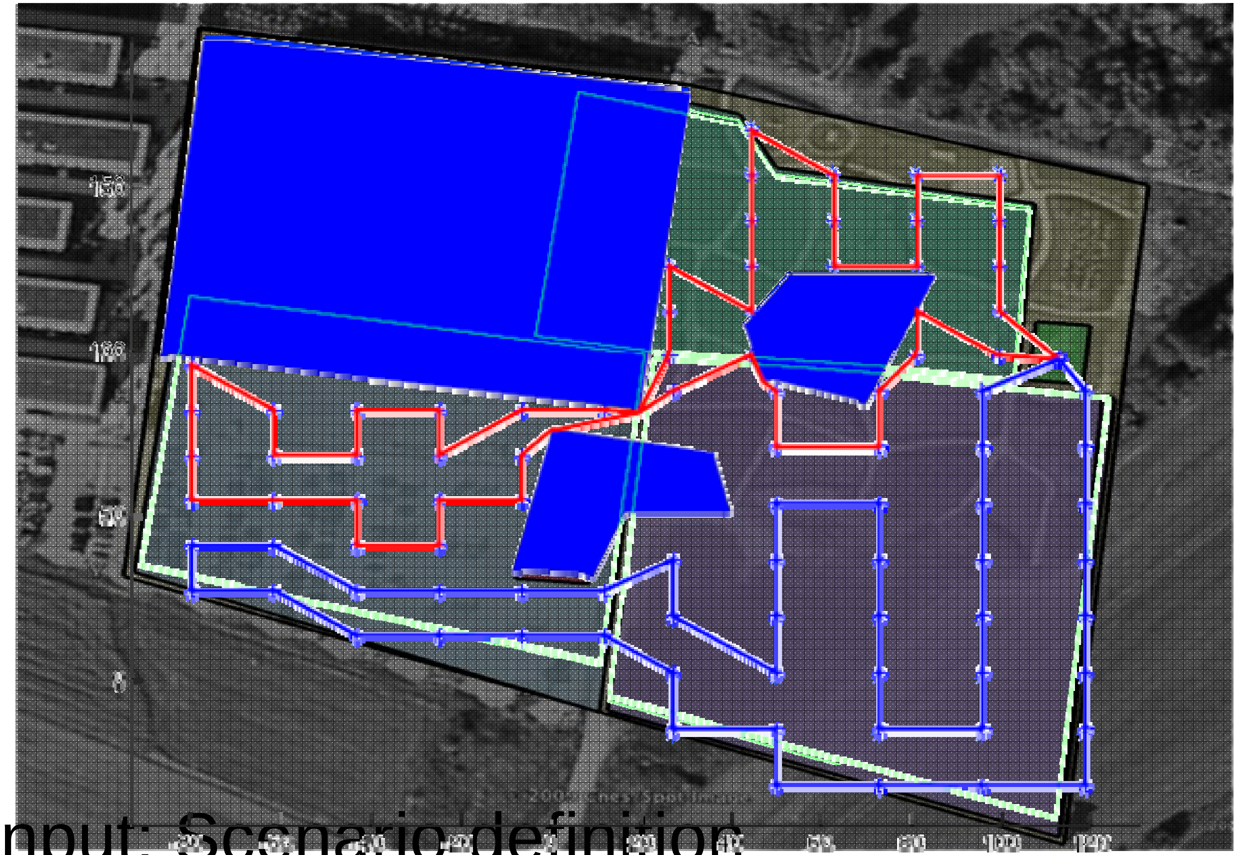
- Battery-powered quatrocopter
  - about 1 m size, 200g payload
  - 20 minutes operation time
  - onboard camera 10MPixel
- GPS-based waypoint navigation
- Communication
  - Uplink (RC channel): remote control;
  - Downlink (2.4 GHz channel): flight data, (low-resolution) images/video

cDrones: Mission Planning

- Find the optimal routes & formation for a small group of UAVs
  - Sequence of waypoints & actions
- Given the scenario description
  - Simplified 3D representation
  - Areas of interest, no-fly zones
- Considering various constraints
  - Power, flight time
  - Target resolution, update rate etc.
- Current approach
  - CSP-based planning



cDrones: Mission Planning (2)



Input: Scenario definition

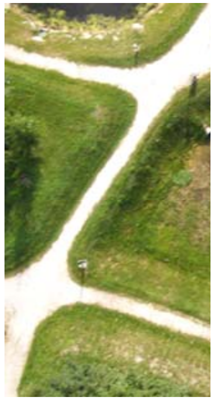
Output: route generation (optimize coverage)

Picture: routes for multiple drones

cDrones: UAV Formation

- Build and maintain a formation
  - e.g. “parallel”, “triangle” (of 3-5 UAVs)
  - Follow the waypoint routes given by mission planning
- Exploit GPS and IMU data of UAVs
  - Guarantee safe flight routes for individual UAVs
  - No online obstacle detection
- Provide real and simulation environment
  - Simplify testing
  - Modeling the UAV dynamics

cDrones: Aerial Imaging



Mosaicking individual images to „overview image“

cDrones: Aerial Imaging (2)

- Video analysis
  - Alignment of frames (ego motion compensation)
  - Object detection & tracking (relative movement within aligned frames)

raw video

analysis

(Potential) further Applications

- Entertainment (computer games)
  - in 3D environments
- „Smart Rooms / Smart Environments
  - detection gestures, sign language, room occupancy …
- Environmental monitoring
  - sensor fusion, habitat monitoring
- Security
  - Safety enhancement (trains, cars), access control, surveillance
- „Virtual Reality“
  - augment real world with digital information
- …