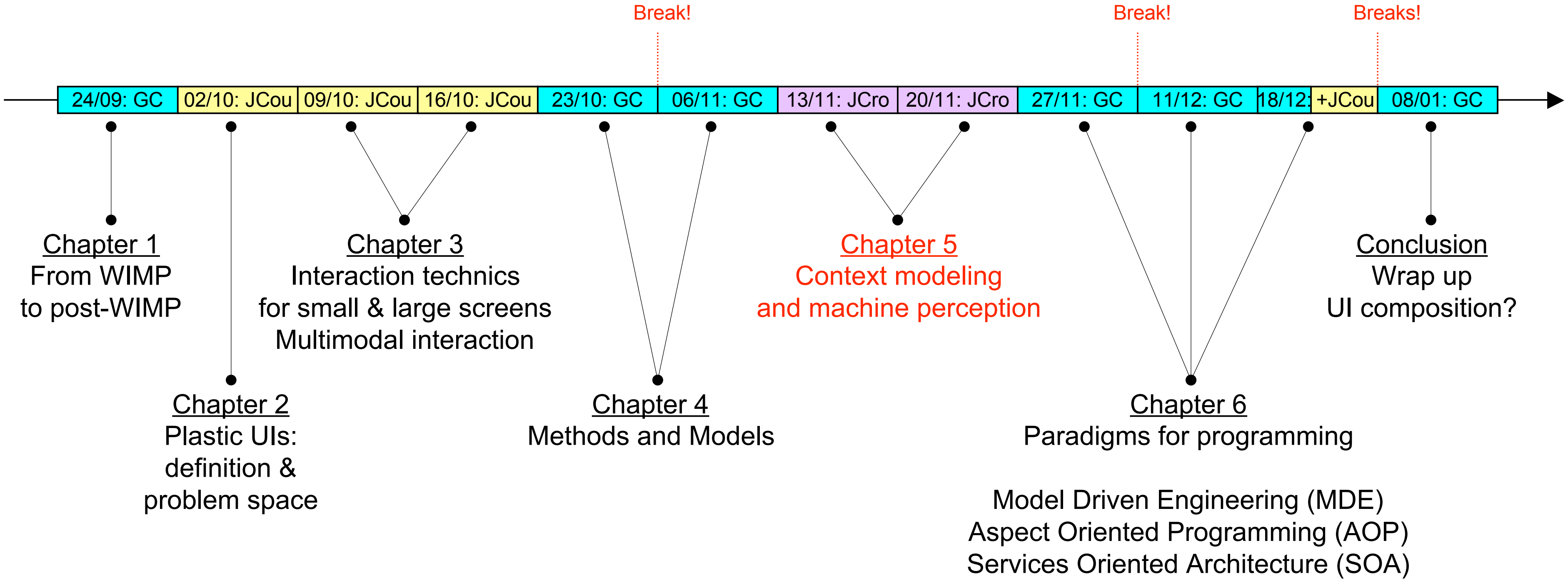
Mobile and Context-aware Interactive Systems

Gaëlle Calvary, Joëlle Coutaz and James L. Crowley

Master of Science in Informatics at Grenoble
Université Joseph Fourier (Grenoble I)
ENSIMAG / INP Grenoble

Outline and schedule



GC: Gaëlle Calvary
JCou: Joëlle Coutaz
JCro: James Crowley

Lesson Plan

1) Introduction: Context Aware Systems and Services
2) Software components for perception, action and interaction
3) Situation Models: a formal foundation for context modeling
4) Acquiring situation models
5) Autonomic methods for software components

Lesson Plan

1) Introduction: Context Aware Systems and Services
2) Software components for perception, action and interactive
3) Situation Models: a formal foundation for context modeling
4) Acquiring situation models
   - Rappel: Definition of Situation Models
   - The acquisition problem
   - Hand-Crafting situation models
   - Off-line: Learning a Generic Situation Model
   - On-line: Accommodating Individual Preferences
5) Autonomic methods for software components

Situation Models:
An analytical tool for describing interactions

P. Johnson-Laird 1983 - Situation Model

An analytical tool to allow Human Psychologists to model human to human interaction.

Situation:	Relations between entities
Entities:	People and things;
Relations:	An N-ary predicate (N=1,2,3 …)
Example:	John is facing Mary. John is talking to Mary.

Situation Models for Interaction

Proposal: Use situation models as a software framework for systems and services that interact with humans

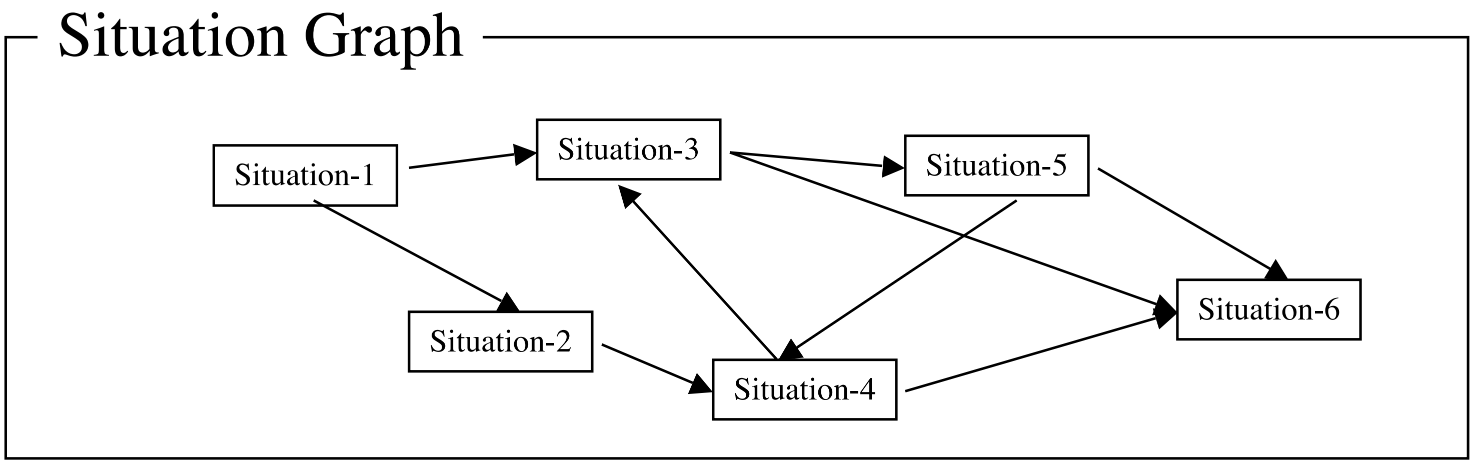
Situation:

- A configuration of relations between entities, with
- The appropriateness of actions for the situation.

Context:

- A situation network composed from
- A set of entities, relations, actions, and situations

Situation Graph



A situation graph describes a state space of situations

A Situation determines:

System Attention: entities and relations for the system to observe

System Behaviours: List of actions that are allowed or forbidden

Situation Models for Interaction

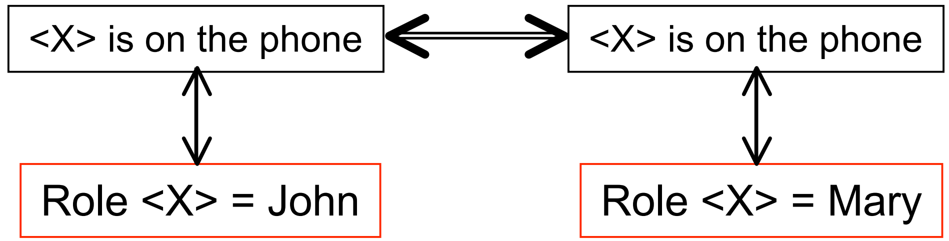
John is on the phone ⟷ John is facing Mary ⟷ John is talking to Aibo

John is on the phone	John is facing Mary	John is talking to Aibo
AIBO Behaviours	AIBO Behaviours	AIBO Behaviours
Be Quite (10)	Be Quite (5)	Be Quite (-10)
Play (-5)	Play (0)	Play (5)
Talk to John (-10)	Talk to John (-5)	Talk to John (10)
Dance (-10)	Dance (-10)	Dance (5)
Sing (-20)	Sing (-20)	Sing (-5)

<u>Each situation indicates:</u>

- Transition probabilities for accessible situations
- The appropriateness or inappropriateness of actions.

Roles and Situations



A <u>role</u> is a "variable" for entities.

Roles allow generalizations of situations.

Roles enable <u>learning by analogy</u>

Roles and Situations

Role: An abstract person or thing

A role predicts the actions that might be taken by an actor or the actions enabled by an object.

Entity: A correlated set of observed properties.

Two kinds of entities:

Actor: An entity that can spontaneously act to change a situation.

Prop: An entity that can not spontaneously act.

Situation Models as Scripts for Services

Many human activities follow scripts, but with variations.

Proposal: script services as a network of situations.

Formal Definitions:

Situation: A configuration of entities playing roles.

Configuration: A set of relations (predicates).

Relation: A predicate on properties of one or more entities.

Situation Models as Scripts for Services

Fundamental Problem

The Knowledge Barrier:

The extreme complexity of human activity and individual preferences

Proposed Solution

Machine Learning

Off-line: Learning of prototype scripts

On-line: Adaptation of scripts to accommodate preferences

Learning Situation Models

<u>Four Learning Problems</u>

Learning Service Behaviour

⇒ Supervised on-line Learning

Learning Situations Graphs

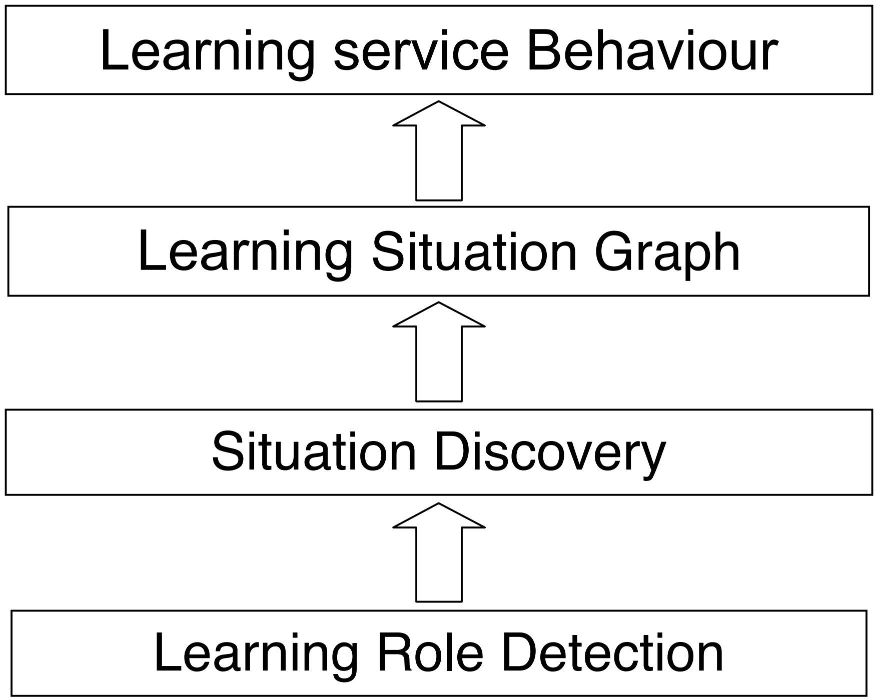
⇒ Supervised off-line Learning

Situation Discovery

⇒ Unsupervised off-line Learning

Role Detection

⇒ Off-line Statistical Learning



Learning Role Assignment

<u>Four Learning Problems</u>

Learning Service Behaviour

⇒ Supervised on-line Learning

Learning Situations Graphs

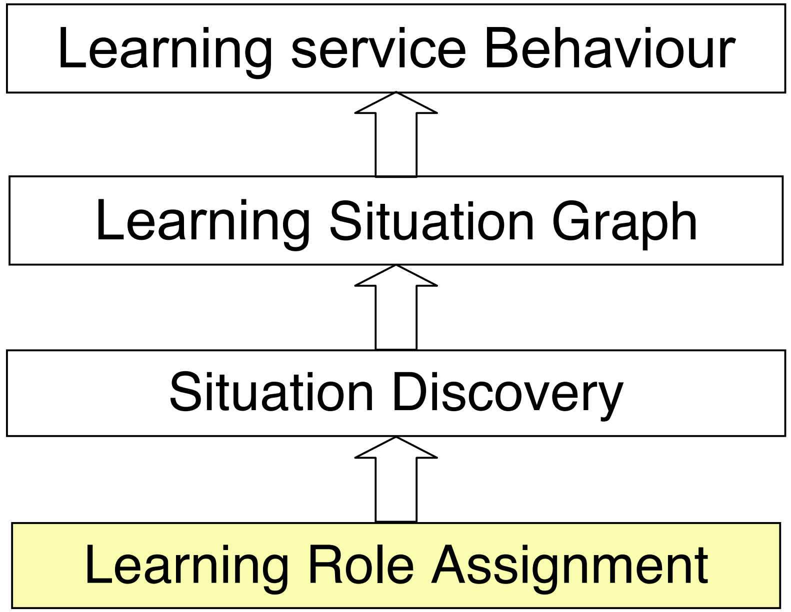
⇒ Supervised off-line Learning

Situation Discovery

⇒ Unsupervised off-line Learning

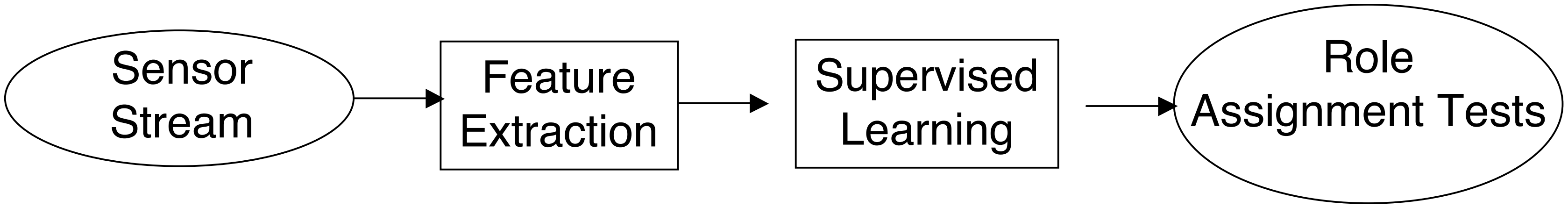
Role Assignment

⇒ Off-line Statistical Learning



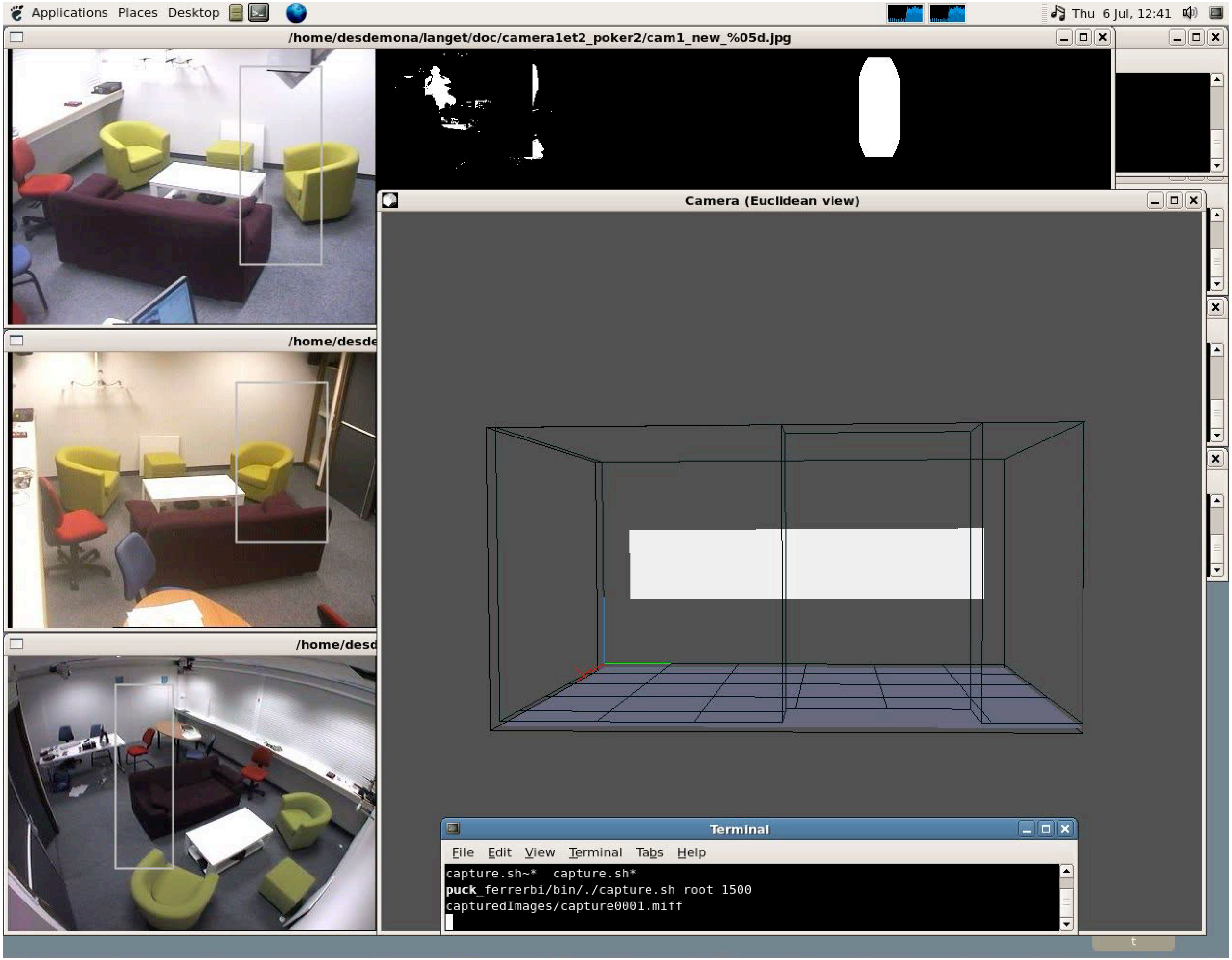
Role Detection: Supervised learning using SVMs

Learning Role Assignment



Approach:

1) Visual and acoustic tracking
2) Compute feature stream
3) Hand Label examples
4) Train Role Recognition Classifiers


Applications Places Desktop
Thu 6 Jul, 12:41
/home/desdemona/langet/doc/camera1et2_poker2/cam1_new_%05d.jpg
/home/desde
/home/desd
Camera (Euclidean view)
Terminal
File Edit View Terminal Tabs Help
capture.sh~* capture.sh*
puck_ferrerbi/bin/./capture.sh root 1500
capturedImages/capture0001.miff

Simple Features from Tracker

1. Acoustic and Visual Blob tracking
2. Features for P persons

Visual Features:

- 3D Position
- 3D Blob Speeds
- Lengths, orientation of 2nd moments
- Face Orientation

Acoustic Features:

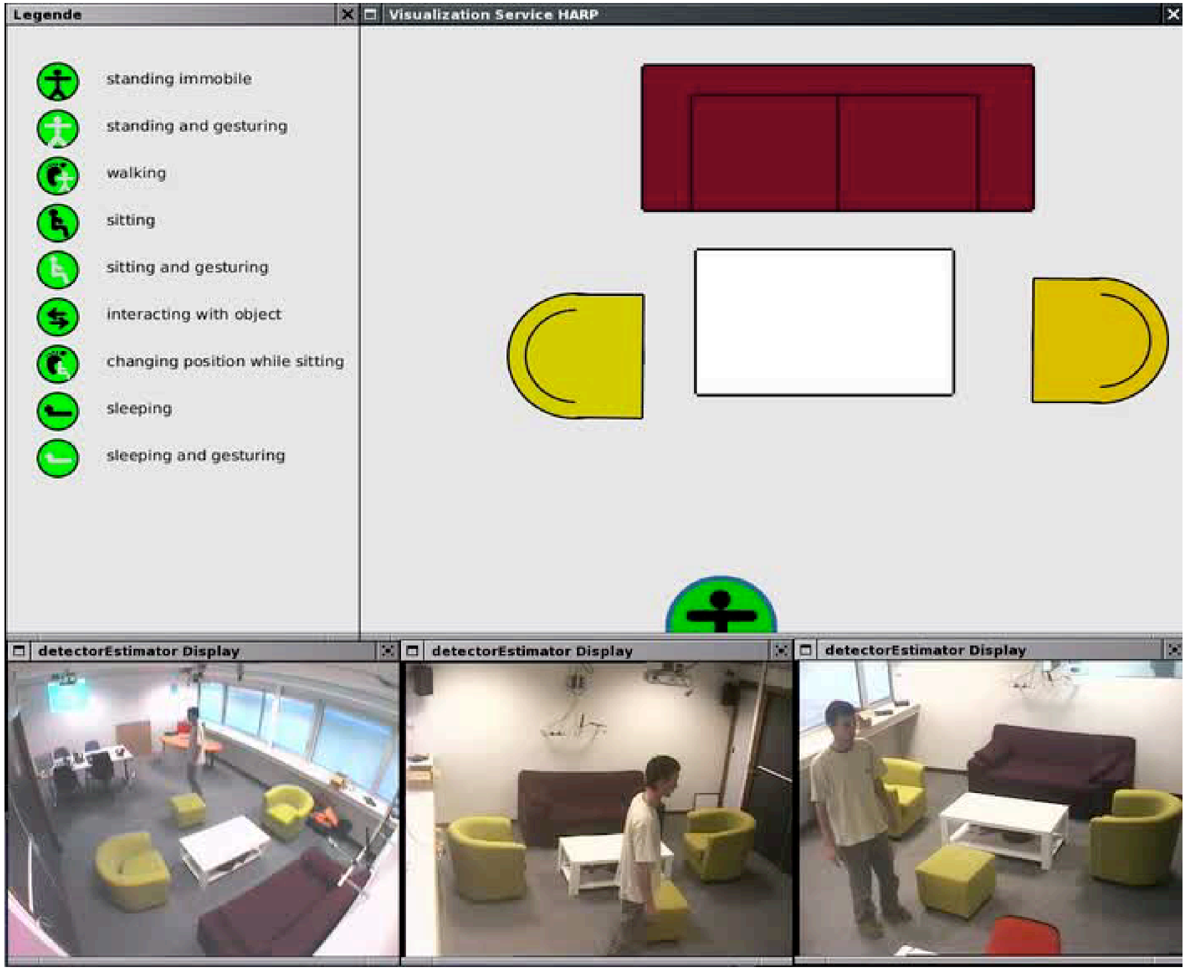
- Speech activity detection

Also available: Statistical appearance features (not used here).

Role Assignment

Each blob is tested using statistical classifier.

Most likely blob is assigned to role.



Learning Situation Models

<u>Four Learning Problems</u>

Learning Service Behaviour

⇒ Supervised on-line Learning

Learning Situations Graphs

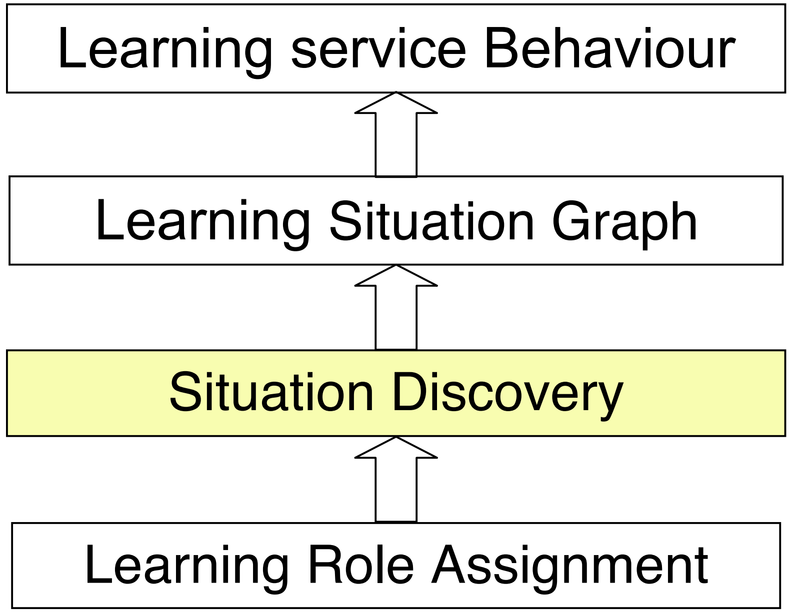
⇒ Supervised off-line Learning

Situation Discovery

⇒ Unsupervised off-line Learning

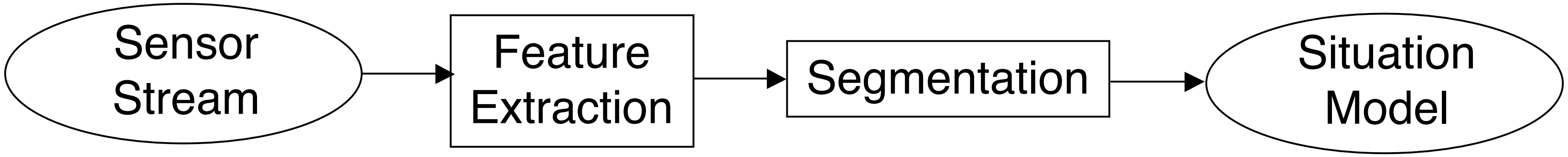
Role Assignment

⇒ Off-line Statistical Learning



Initial Situation Model: Learned by Segmenting Feature Stream

Learning an Initial Situation Model

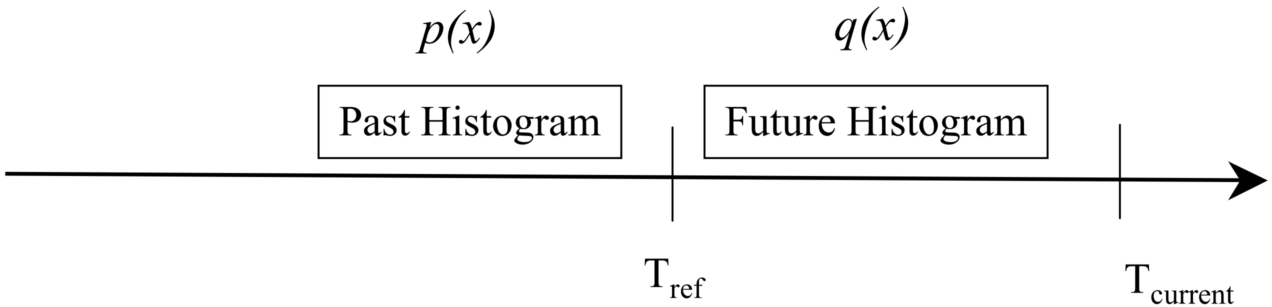


Approach:

1) Visual and acoustic tracking
2) Compute feature stream
3) Calculate running histograms of features
4) Calculate Jeffrey Divergence between past and future
5) Detect rupture in Jeffrey Divergence

segmenting feature stream

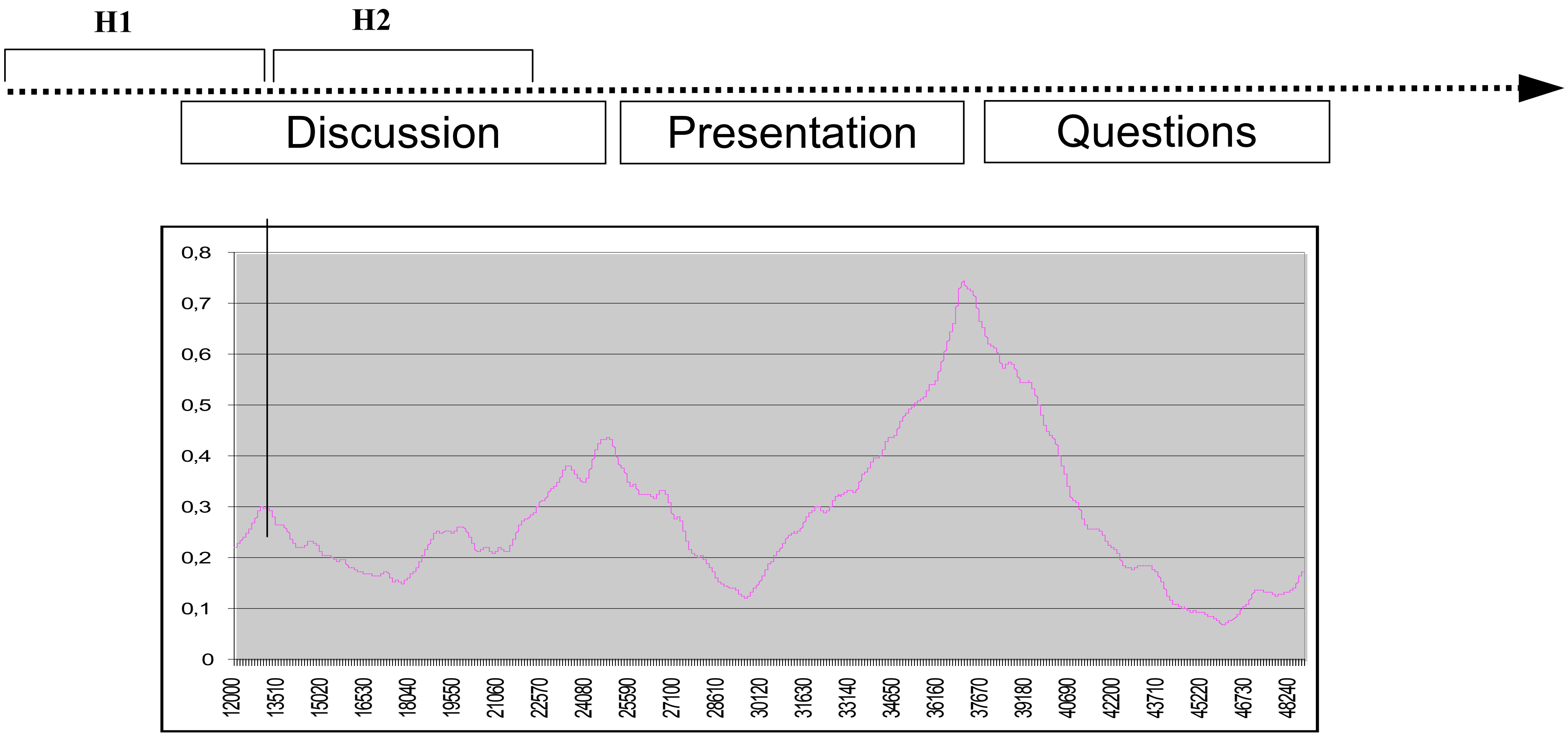
3. Compute D-Dimensional Histogram from last N frames
4. Compute Jeffery Divergence between Past and Future



$$J_{p,q} = \sum_{x \in X} p(x) \cdot \log \frac{p(x)}{\frac{p(x)+q(x)}{2}} + q(x) \cdot \log \frac{q(x)}{\frac{p(x)+q(x)}{2}}$$

Segmenting Feature Stream

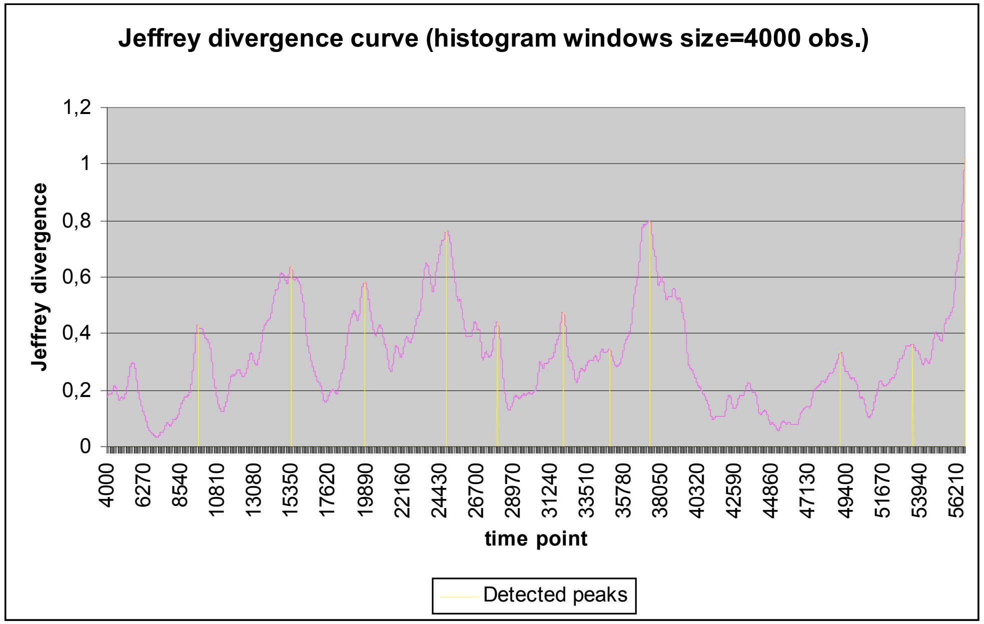
Slide two adjacent histograms from the beginning to the end of a recording, while calculating Jeffrey divergence

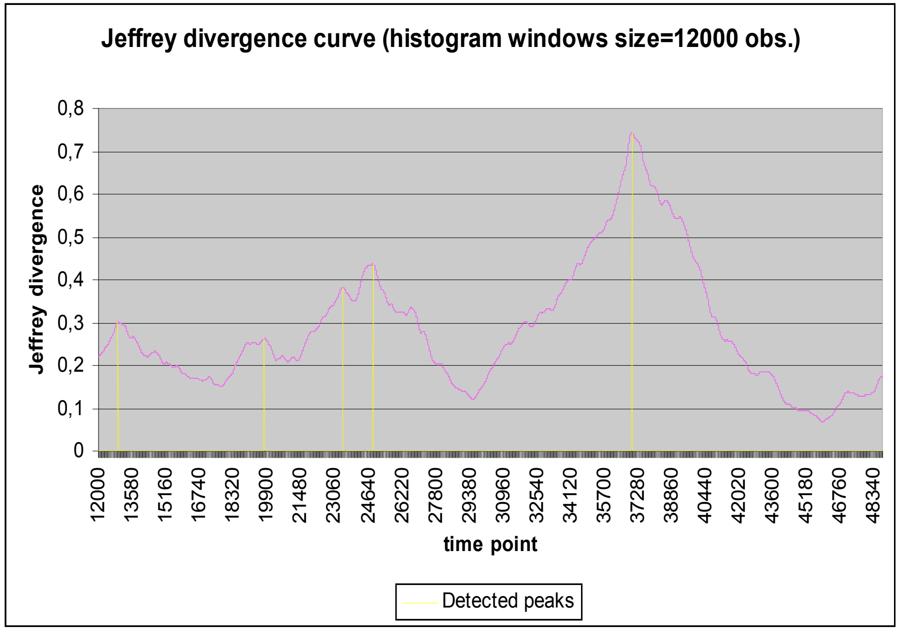


Segmenting Feature Stream

multi-scale analysis: Jeffrey divergence curves for different window sizes

4000-16000 observations (between 64sec and 4min 16sec)





Learning Situation Models

<u>Four Learning Problems</u>

Role Detection

⇒ Statistical Learning

Situation Discovery

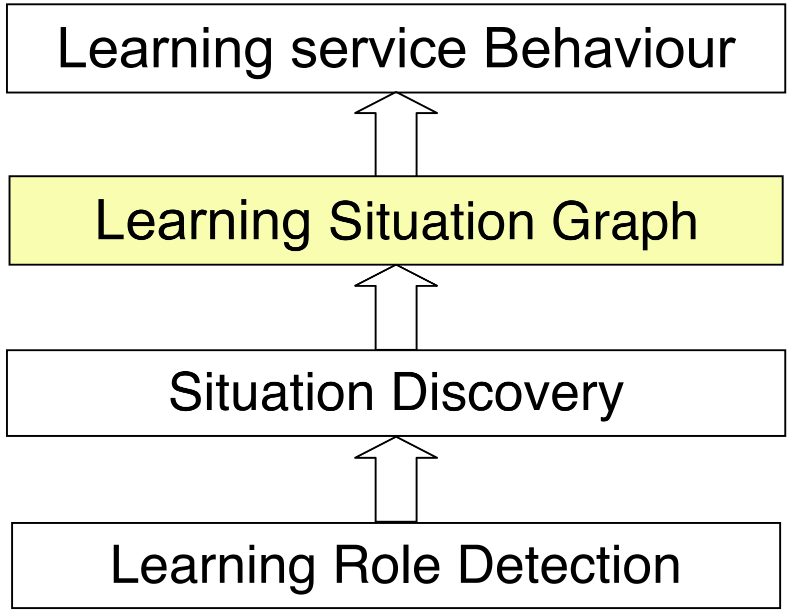
⇒ Unsupervised Learning

Learning Situations Graphs

⇒ Supervised Learning

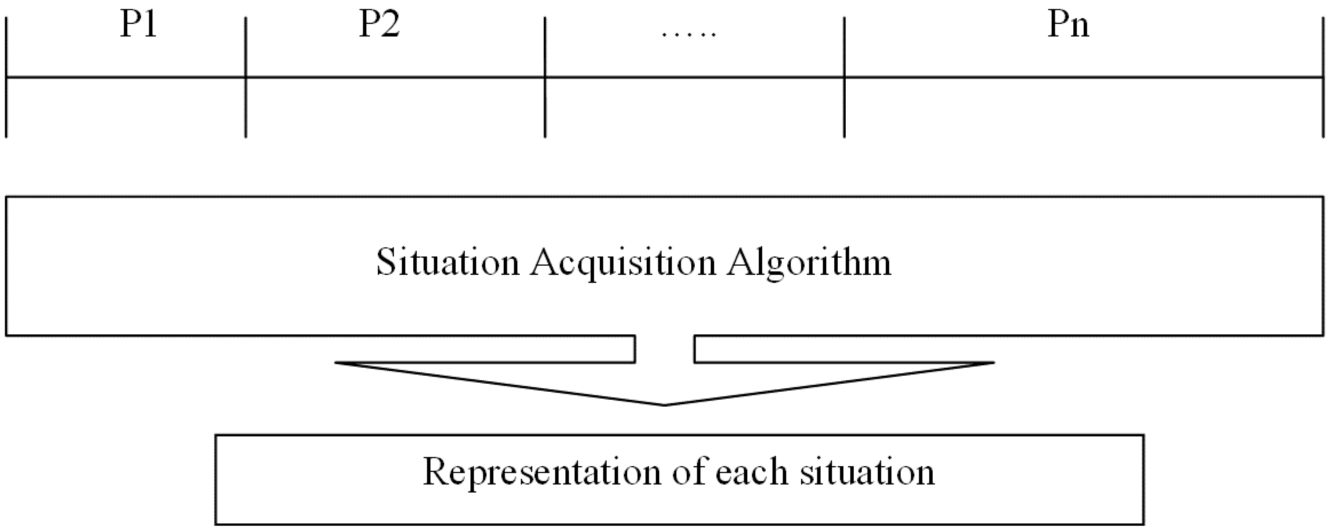
Learning service Behaviour

⇒ Supervised Learning



Supervised Situation Learning

n observation sequences associated to *m* situation labels ($m \leq n$)



Each sequence corresponds to one situation

Two or more sequences can have the same situation label

Supervised Situation Learning (2/2)

learner L: $\{P_1, P_2, \ldots, P_k \mid k > 0\} \rightarrow$ situation representation S

```
A. For each learner class do:
    a. {optimization step}
       For each situation label do:
          •  Select learner/set of learners
          •  Apply learner to given observations
    b. {validation step}
       Calculate quality of obtained situation
       representations
    c. Repeat a.-b. until best quality is obtained

B. Choose learner class with best quality of situation
   representations
```

Supervised Situation Learning (2/2)

```
A. For each learner class do:
     a. {optimization step}
        For each situation label do:
          •  Select learner/set of learners
          •  Apply learner to given observations
     b. {validation step}
        Calculate quality of obtained situation
        representations
     c. Repeat a.-b. until best quality is obtained

B. Choose learner class with best quality of situation
   representations
```

-- Iterate over learner classes --

Supervised Situation Learning (2/2)

```
A. For each learner class do:
     a. {optimization step}
        For each situation label do:
            •  Select learner/set of learners
            •  Apply learner to given observations
     b. {validation step}
       Calculate quality of obtained situation
       representations
     c. Repeat a.-b. until best quality is obtained

B. Choose learner class with best quality of situation
   representations
```

-- Iterate over situation labels to be learned --

Supervised Situation Acquisition Algorithm

```
A. For each learner class do:
    a. {optimization step}
       For each situation label do:
          •  Select learner/set of learners
          •  Apply learner to given observations
    b. {validation step}
       Calculate quality of obtained situation
       representations
    c. Repeat a.-b. until best quality is obtained

B. Choose learner class with best quality of situation
   representations
```

Quality measure – principle: *maximize the distance between the means of the classes while minimizing the variance within each class [Fisher1938]*

On-line: Adapting to User Preferences

<u>Four Learning Problems</u>

Learning Service Behaviour

⇒ Supervised on-line Learning

Learning Situations Graphs

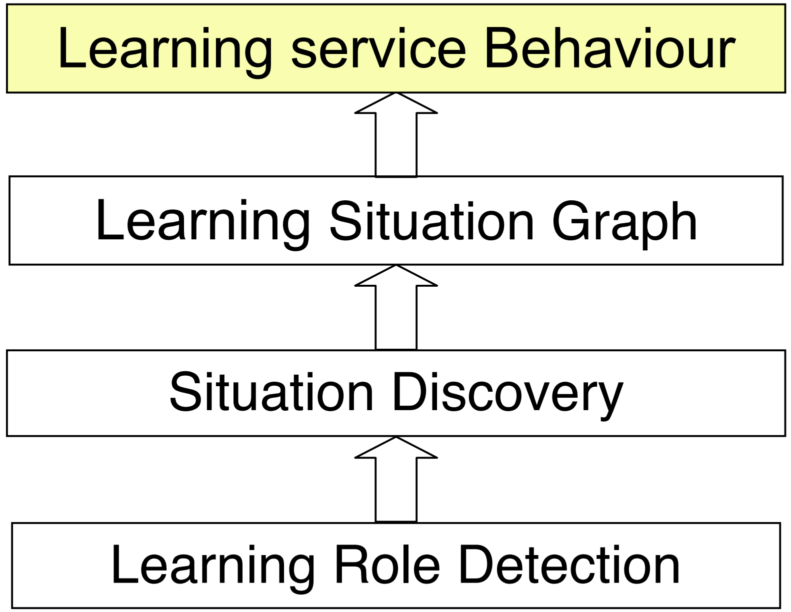
⇒ Supervised off-line Learning

Situation Discovery

⇒ Unsupervised off-line Learning

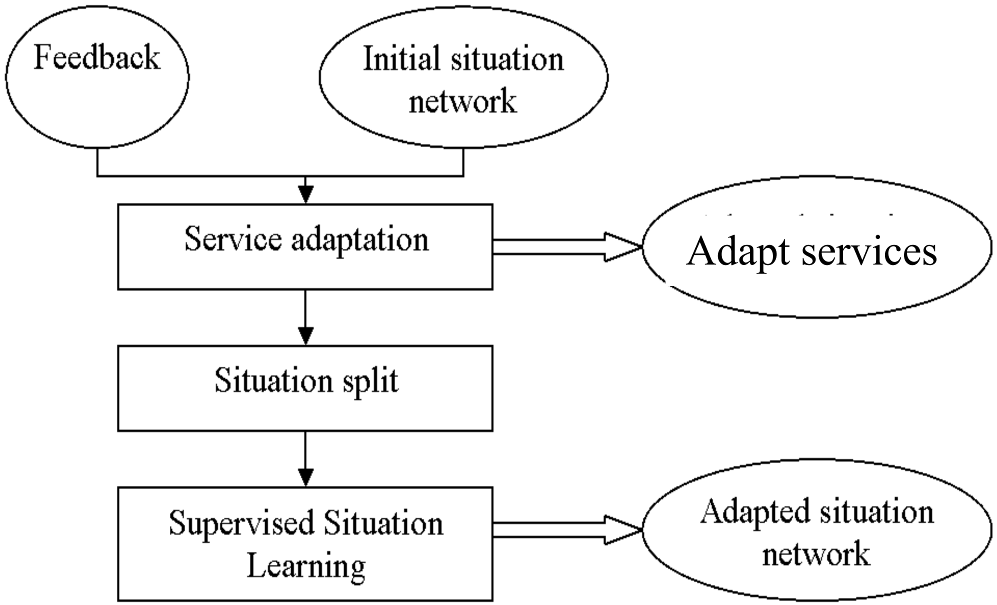
Role Assignment

⇒ Off-line Statistical Learning



Split and Merge Situations from User Feedback

On-line: Adapting to User Preferences

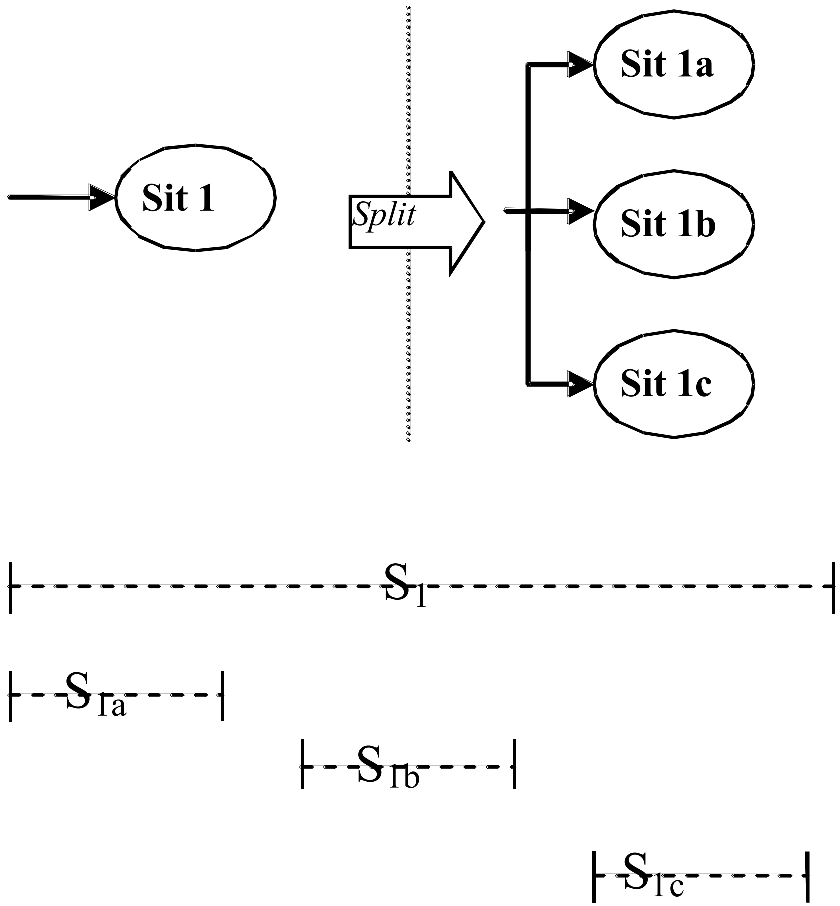


1. Modify association of service with situations
2. Split and Merge for Situations based on Human Feedback

Situation Split

different disjunctive services for one situation → situation split

supervised situation acquisition algorithm for learning sub-situations



Multimodal observation of the scene

head set microphones + speech activity detection

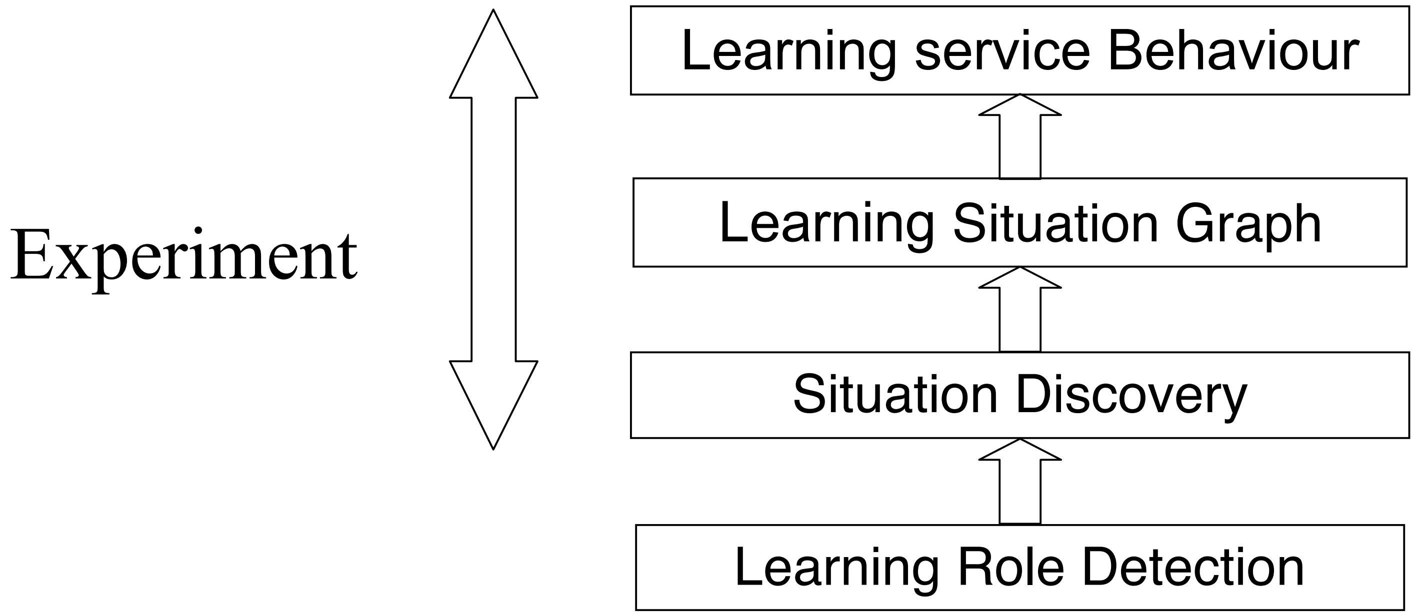
ambient sound detection

multimodal entity observation codes:

0 : entity does not exist
1 : standing immobile
2 : standing and interacting with table
3 : standing and gesturing
4 : standing and interacting with table (in movement)
5 : walking
6 : sitting
7 : sitting and interacting with table
8 : sitting and gesturing
9 : sitting and interacting with table (in movement)
10 : changing position while sitting
11 : lying down
12 : lying down and gesturing
13 : detection error

14-26 : entity is speaking
27-39 : there is noise in the environment
40-52 : entity is speaking and there is noise

Experimental Demonstration

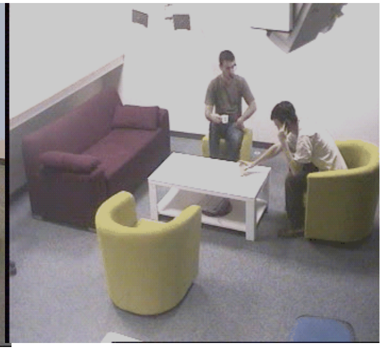


Experimental Demonstration

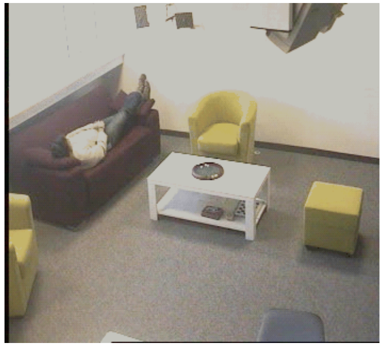
3 scenarios recordings

Situations: “introduction”, “aperitif”, “siesta”, “presentation
“game”

Introduction



Aperitif



Siesta

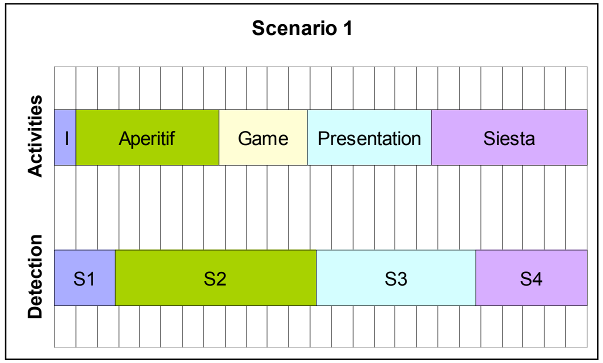
Presentation

Game

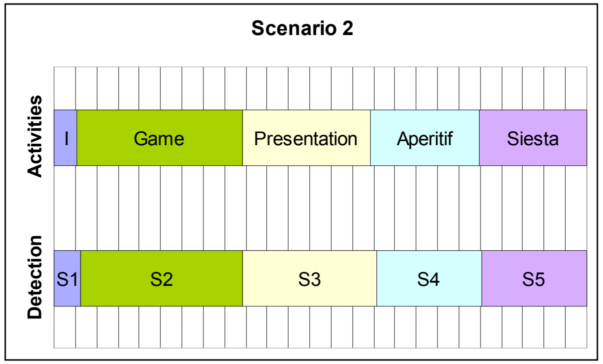
“online” in-scenario situation recognition

Experimental Demonstration

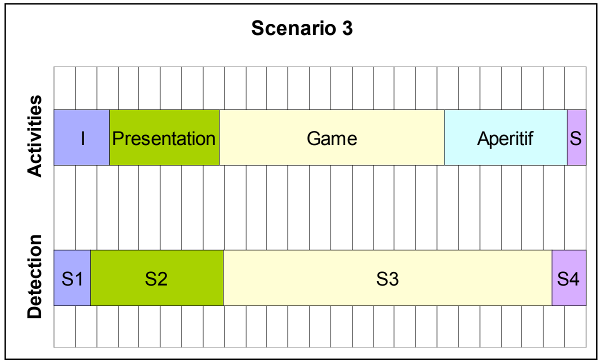
Off-line: unsupervised situation discovery for protype situations



Q = 0.68



Q = 0.95

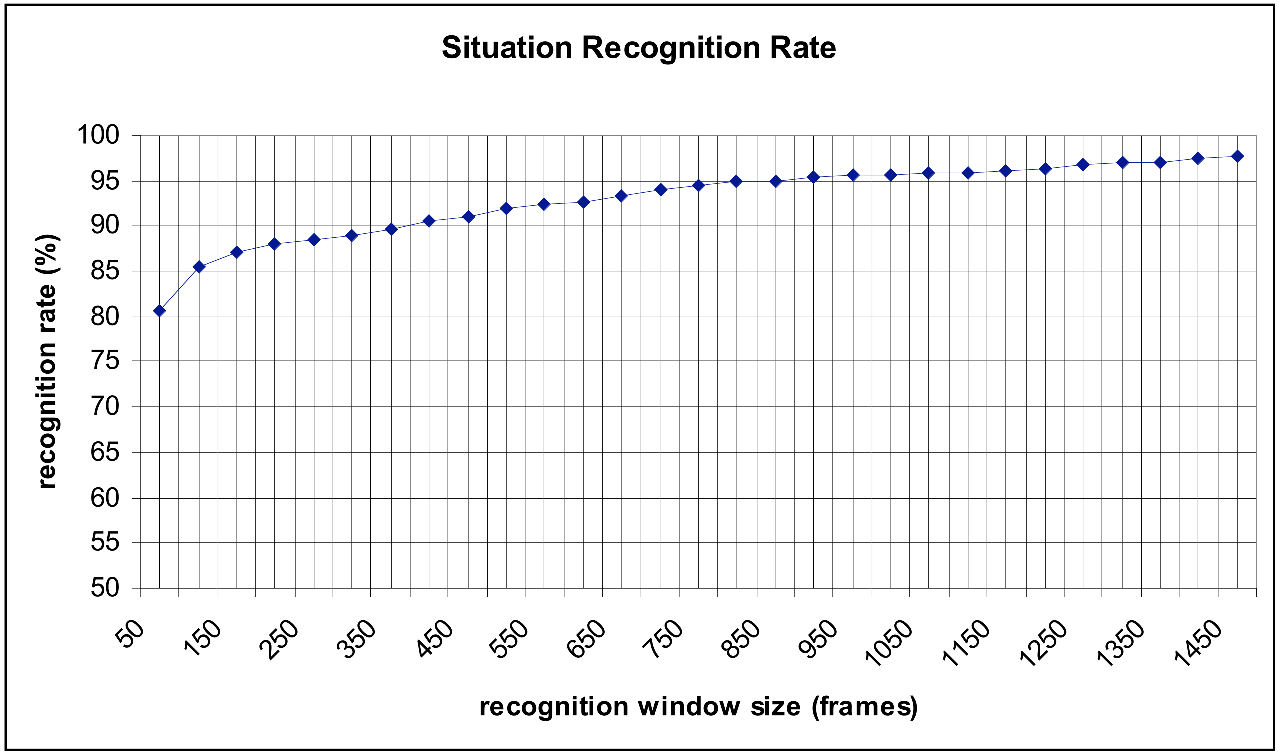


Q = 0.74

Experimental Demonstration

Supervised situation learning

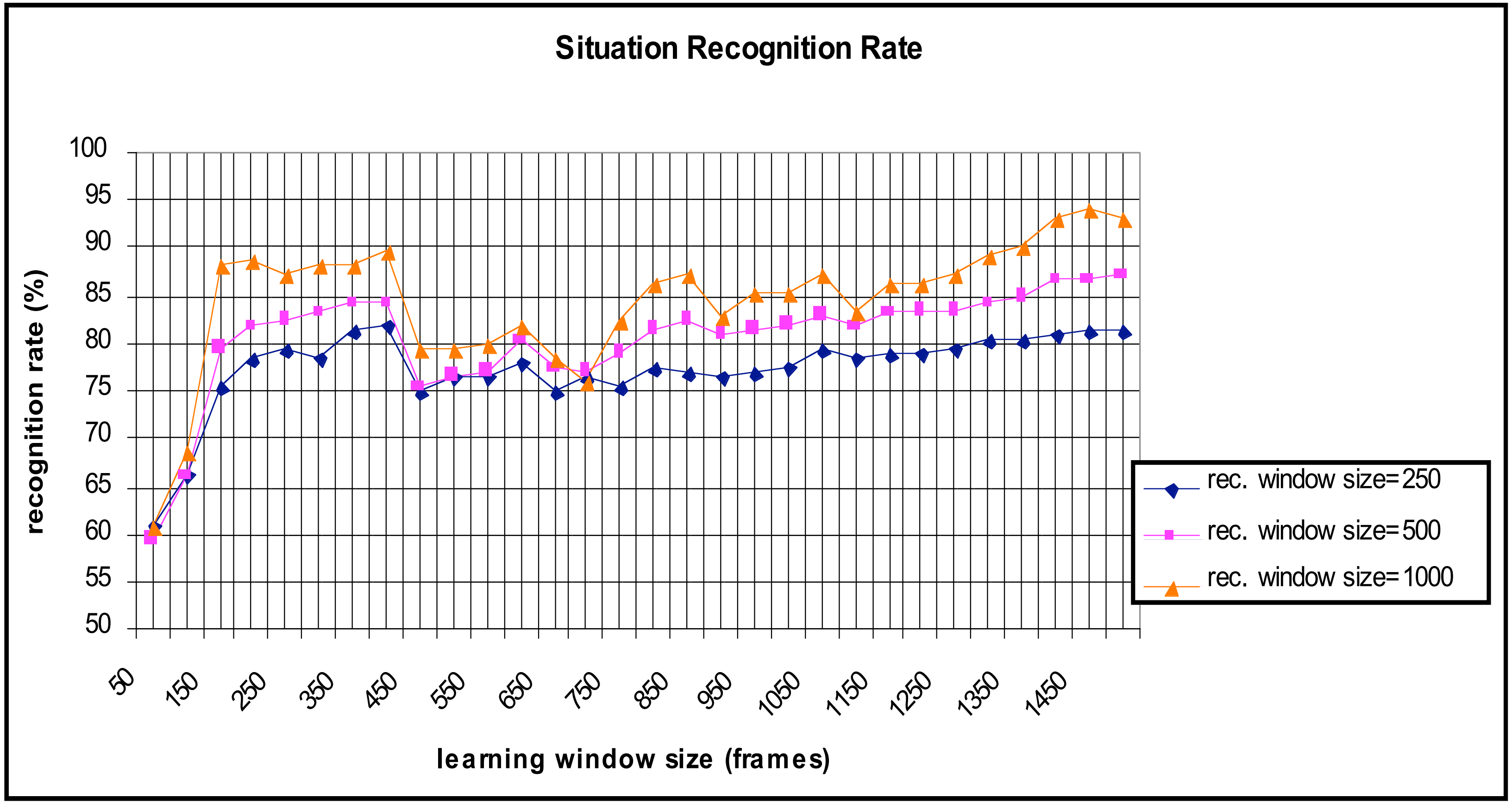
- 4 situations: “introduction”, “presentation”, “siesta” and *“group activity”*
- 3-fold cross-validation: 2 scenarios for learning, 1 scenario for testing
- EM as learner class
- HMMs (8-16 states)



Experimental Demonstration

On-line: Integration of user preferences

- Split *"group activity"* and learn new sub-situations *"aperitif"* and *"game"*



Summary and Conclusions

Situation networks provide scripts for services

Fundamental Problem: Knowledge Barrier

Proposed solution: machine learning

- Off-line: Learn prototype “generic” scripts.
  - ⇒ Initial situation discovery by unsupervised segmentation
  - ⇒ Supervised learning to build situation networks
- On-line: Adapt Networks and services to preferences
  - ⇒ Use feedback from people for on-line adaptation of situation models and services
  - ⇒ Split and merge Situation networks.
  - ⇒ Associate services to situations.

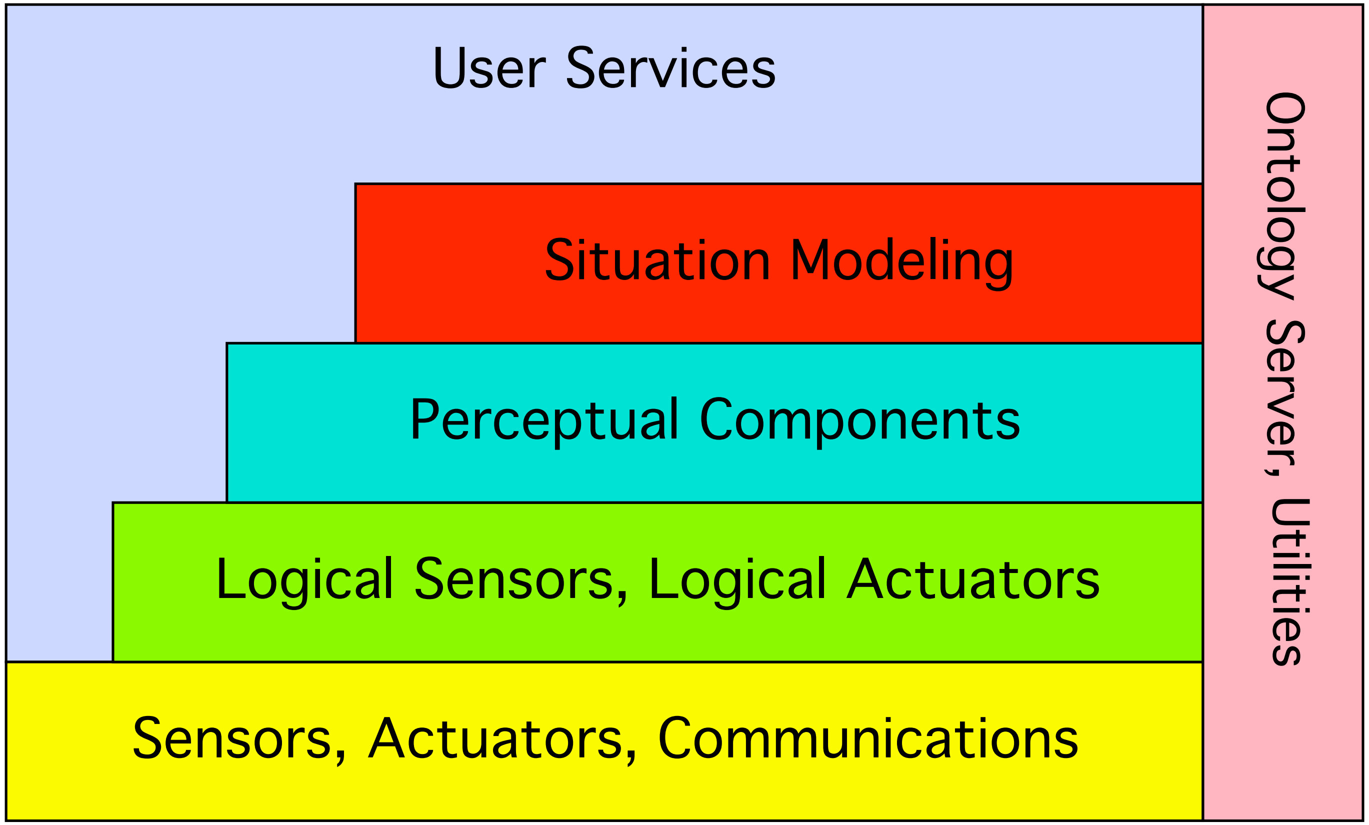
Lesson Plan

1) Introduction: Context Aware Systems and Services
2) Software components for perception, action and interactive
3) Situation Models: a formal foundation for context modeling
4) Acquiring situation models
5) Autonomic methods for software components

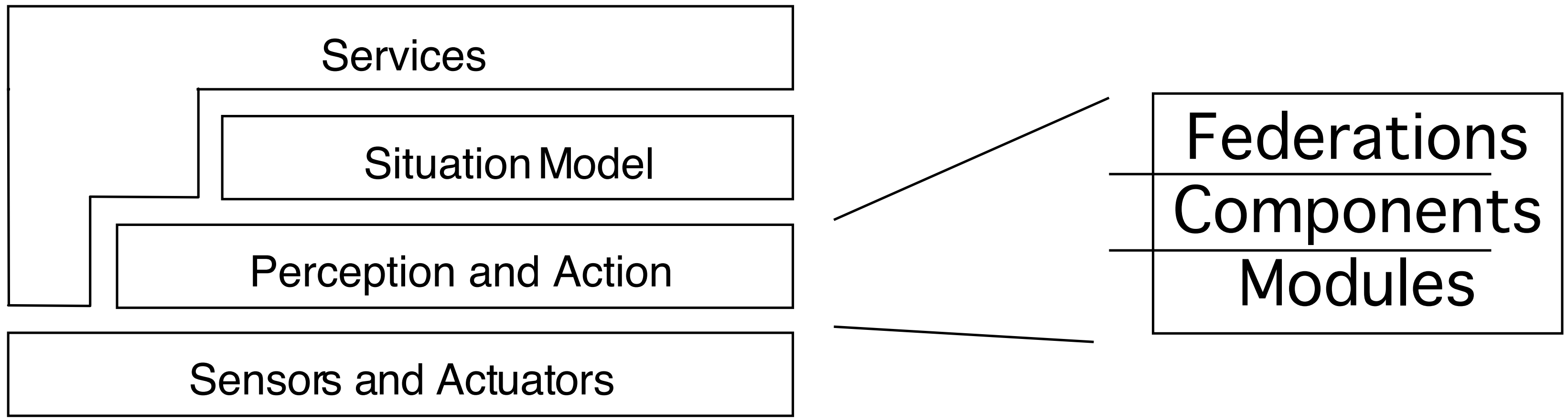
Lesson Plan

1) Introduction: Context Aware Systems and Services
2) Software components for perception, action and interactive
3) Situation Models: a formal foundation for context modeling
4) Acquiring situation models
5) Autonomic methods for software components
   - Reminder: Components for Perception and Action
   - Need for Autonomic Components
   - Origins of Autonomic Computing
   - Autonomic Properties
   - Methods for building Autonomic perception/action components

Software Architectural Reference Model



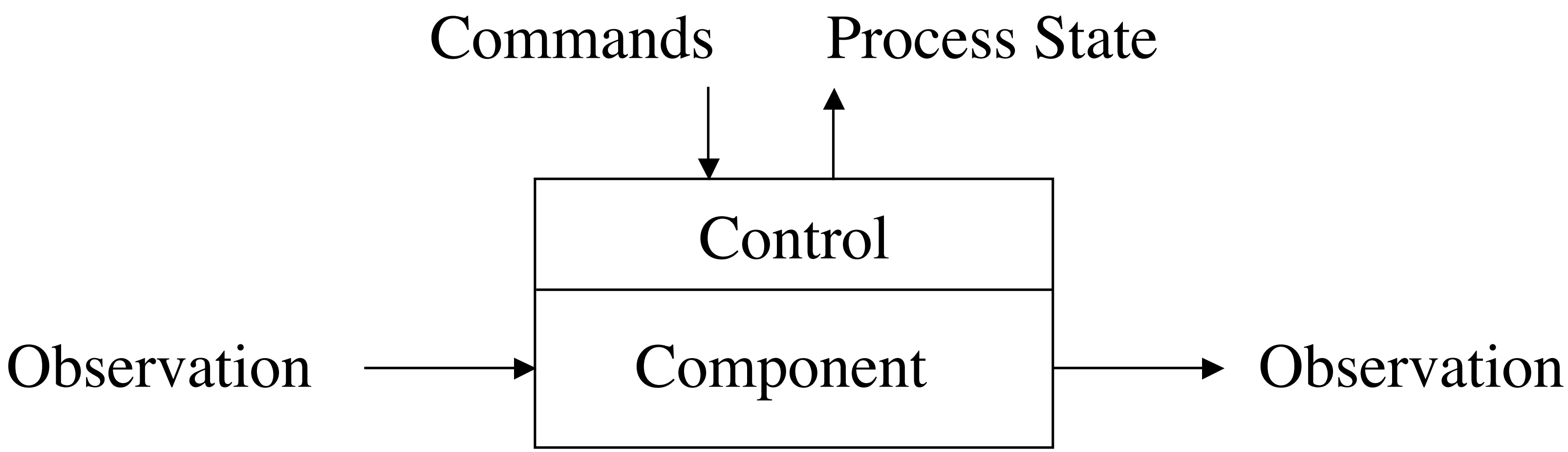
Components for Perception and Action



Perception - Action Layer:

Ad-hoc assembly of components to provide software services.

Sensory Motor Components



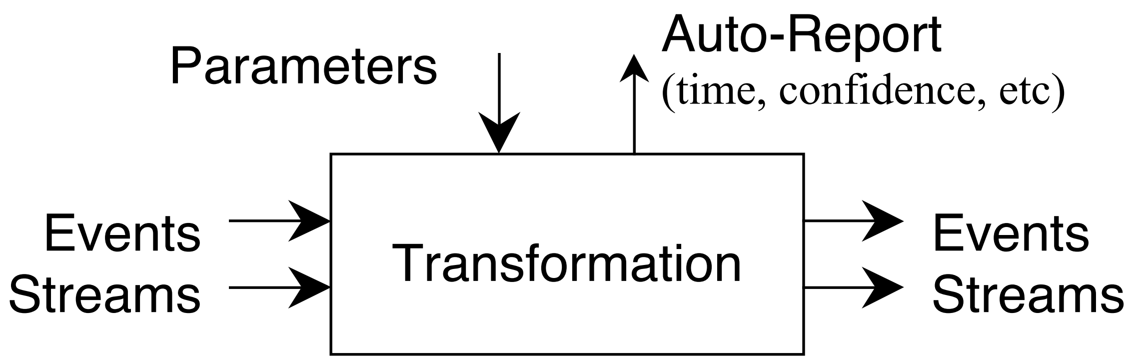
Process model (Finkelstein et al 94).

Data flow Software Architecture (Shaw-Garlan 96)

Process Federations (Estublier and Cunin 97)

Auto-Critical Software Modules

Perceptual Components are composed of modules.



Module: Synchronous Data Transformation

Modules transform data and returns a report on results

Report describes resources used (time, memory) and quality of result

Start-up: Blue Eye Video

PDG: Pierre de la Salle, Jean Viscomte, Stephane Richetto, Pierre-Jean Riviere, Fabien Pelisson, Sebastien Pesnel and James L. Crowley

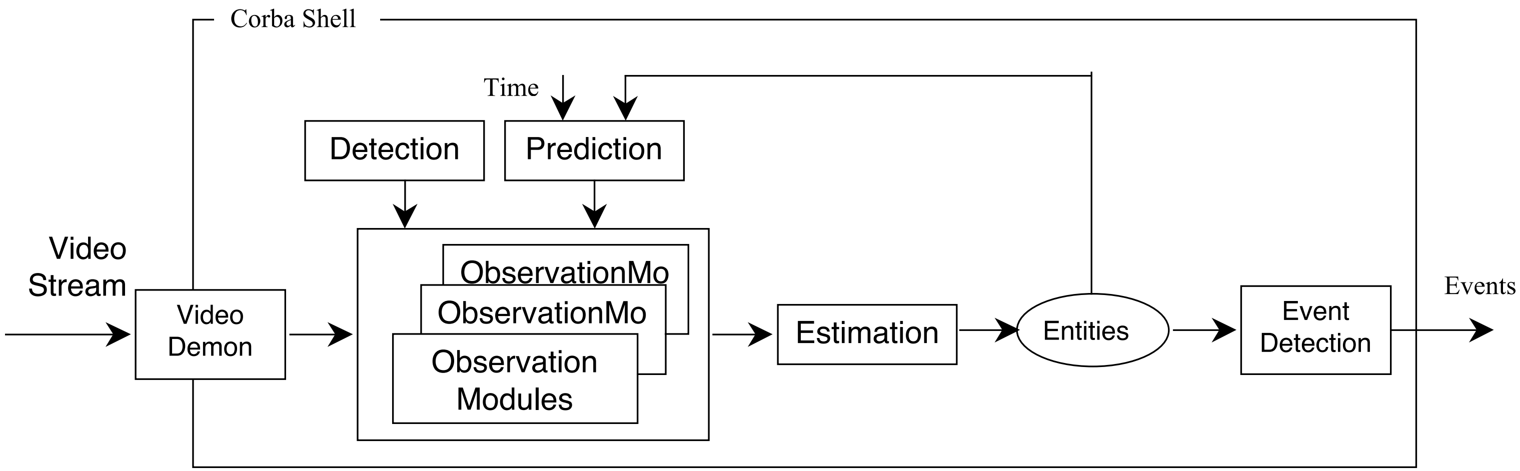
Creation: 1 June 2003

Product: Autonomous IP-Camera with embedded detection and tracking.
Market: Observation of human activity
Sectors: Commercial services, security, and traffic monitoring

Status: > 400 K Euros in sales in 2006, > 200 Systems installed
Sales doubled every 12 months until 2006

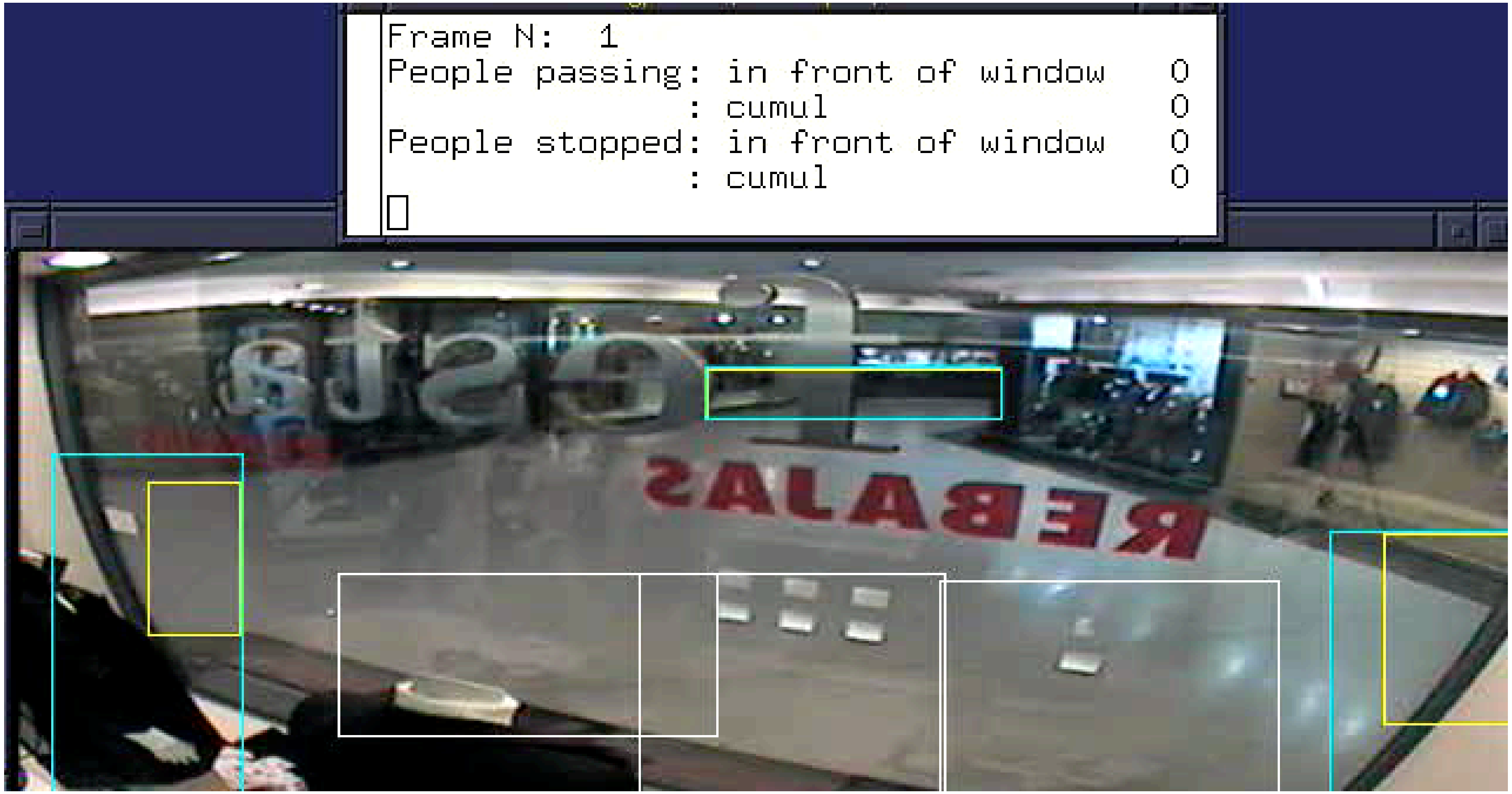
Barrier: Installation and maintenance

Blue Eye Video
Entity Detection and Tracking Process



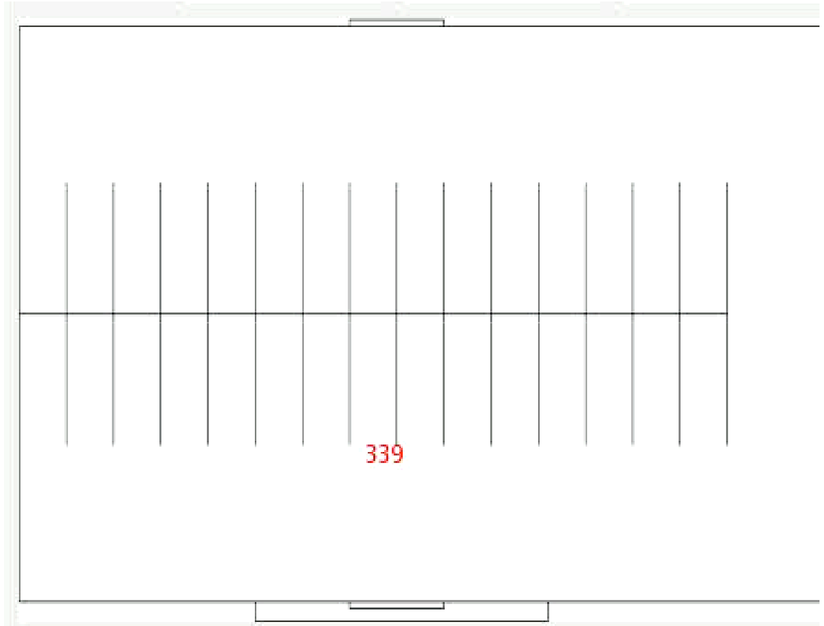
- Hardwired Control in C++
- Observation Modules:
  - Color Histogram Ratio, Background Difference, Motion History Image,
  - Local Appearance, Receptive Field Histograms
- Industrial Grade System

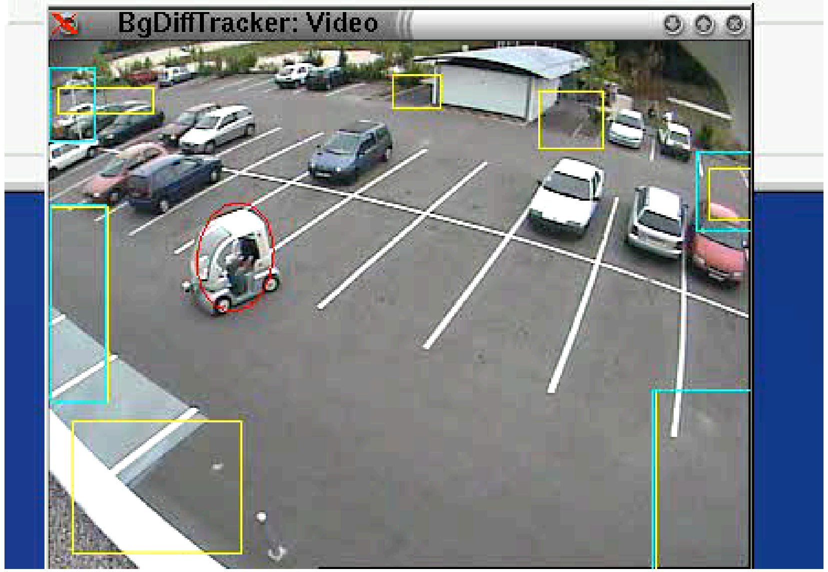
Blue Eye Video Activity Sensor
(PETS 2002 Data)

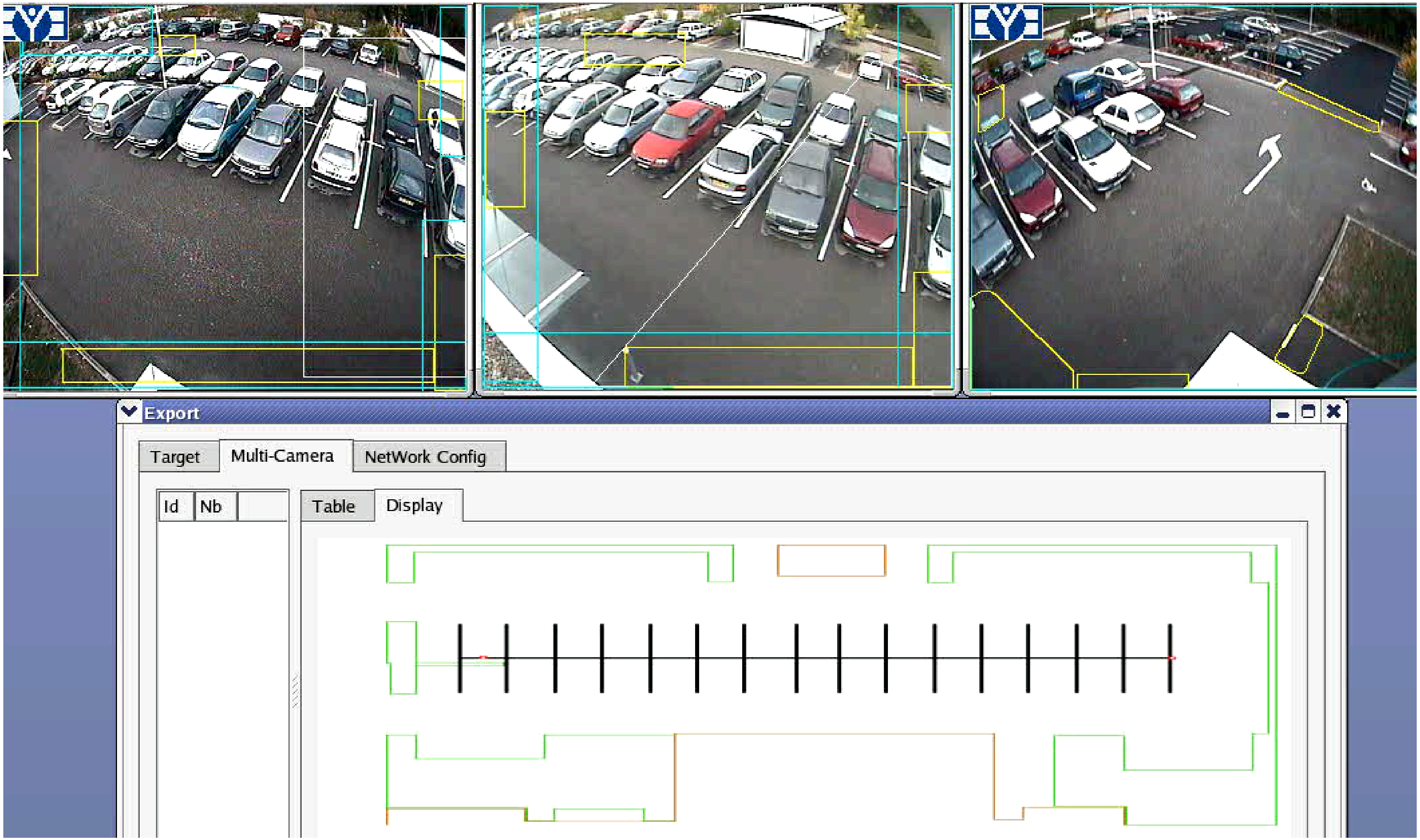


CAVIAR
Outdoor Test Bed
INRIA
Back Parking Lot

2 Outdoor Surveilance Platforms,
3 m separation, 3 meter height

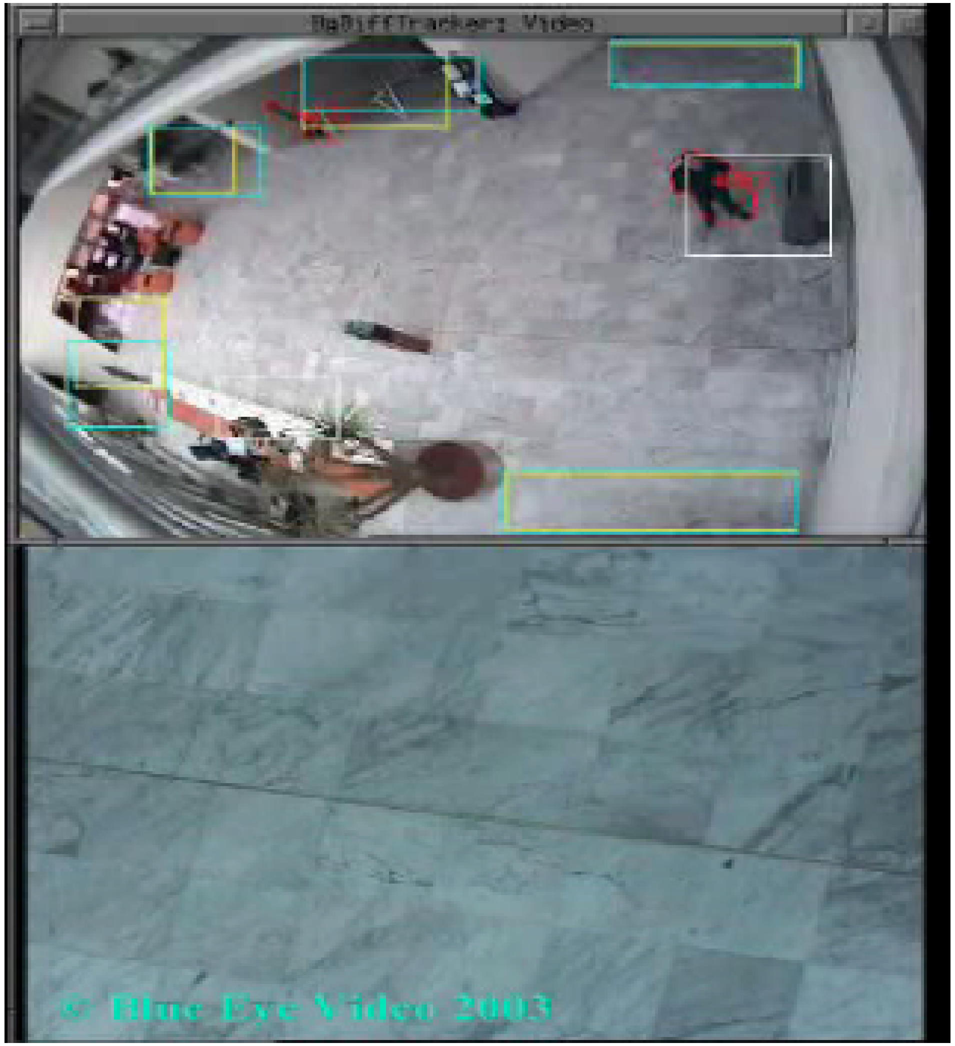




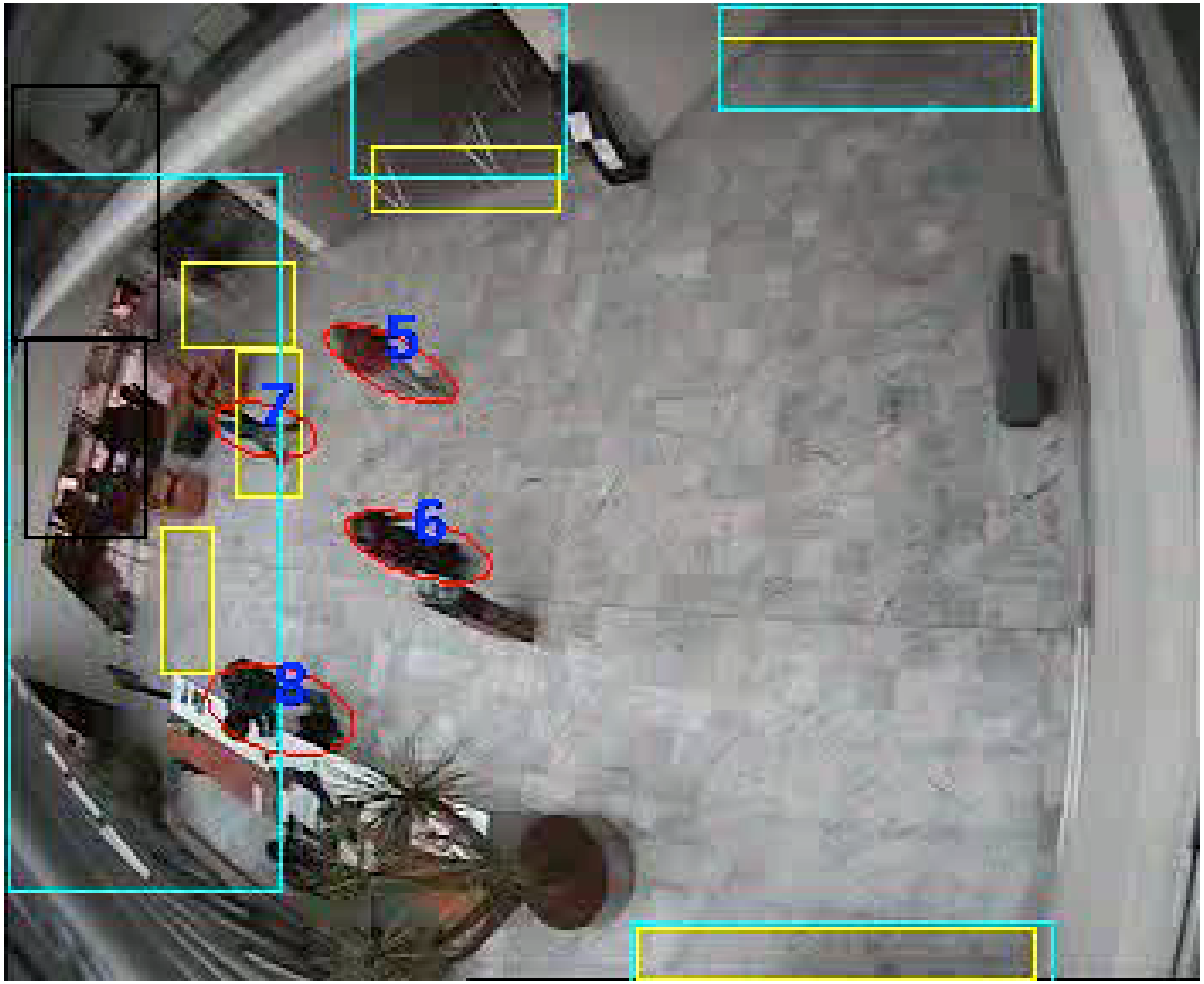

Export
Target
Multi-Camera
NetWork Config
Id
Nb
Table
Display

CAVIAR Indoor Test-bed: INRIA Entrance Hall

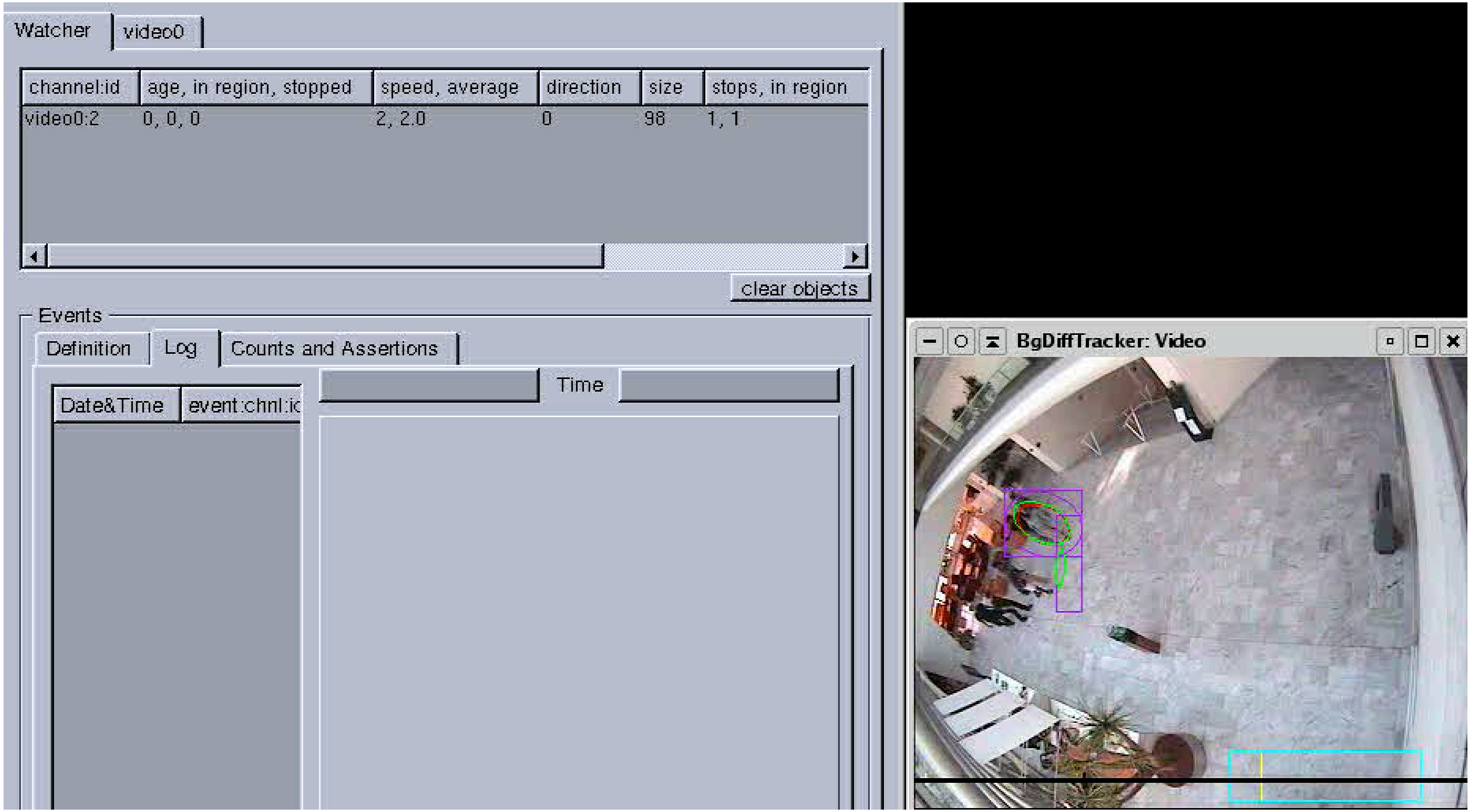
2 Cameras:
one w/wide angle lens,
one steerable pan-tilt-zoom



Tracking Multiple Targets



Abandoned Bag Detection



Reportage FR 2

Lesson from Blue Eye Video

Market Size and potential growth rate were limited by:

1) Robustness,

   and

2) Installation Cost. Currently installation requires configuration by a trained engineers. Maintaining a 100 systems was a full time job for 4 engineers!

Lesson:

Systems must Self-Configure, Self-repair and Self-regulate

⇒ Autonomic Systems Methods are fundamental to Computer Vision and to autonomous robotic systems.

Autonomous Systems

Autonomous: Self-governing, Self-protecting.
Able to self-maintain functional <u>integrity</u>.

Autonomy:

Self-maintenance of functional integrity

Enabling Technology for Autonomy:

Autonomic Computing

Origins of Autonomic Computing

March 2001 Keynote address to the National Academy of Engineers by Paul Horn (IBM vice president)

Autonomic computing systems as systems that manage themselves given high-level objectives from administrators.

Autonomic computing was adapted as a metaphor inspired by natural self-governing systems, and the autonomic nervous system found in mammals.

Autonomic Nervous System (ANS)

The ANS regulates the homeostasis of physiological functions
The ANS is not consciously controlled.

Commonly divided into three subsystems:

- Sympathetic nervous systems (SNS)) (fight or flight)
- Parasympathetic nervous system (PNS) (rest and digest)
- Enteric nervous systems (ENS) (the second brain)

Origins of Autonomic Computing

Autonomic computing :

a metaphor inspired by natural self-governing systems, and the autonomic nervous system found in mammals.

March 2001 Keynote address to the National Academy of Engineers by Paul Horn (IBM vice president)

Autonomic computing systems as systems that manage themselves given high-level objectives from administrators.

Autonomic Nervous System (ANS)

The ANS regulates the homeostasis of physiological functions

The ANS is not consciously controlled.

Commonly divided into three subsystems:

- Sympathetic nervous systems (SNS)) (fight or flight)
- Parasympathetic nervous system (PNS) (rest and digest)
- Enteric nervous systems (ENS) (the second brain)

Autonomic Properties for Perceptual Components

Self-descriptive: The component supervisor provides descriptions of the capabilities (to component registry) and the current state of the process (on request).

Self-monitoring: The component supervisor estimates state and quality of service for each processing cycle.

Self-regulation: The component supervisor adapts parameters to maintain a desired process state.

Self-repair: The process controller can detect and correct conditions by reconfiguring modules.

Autonomic Properties for Perceptual Components

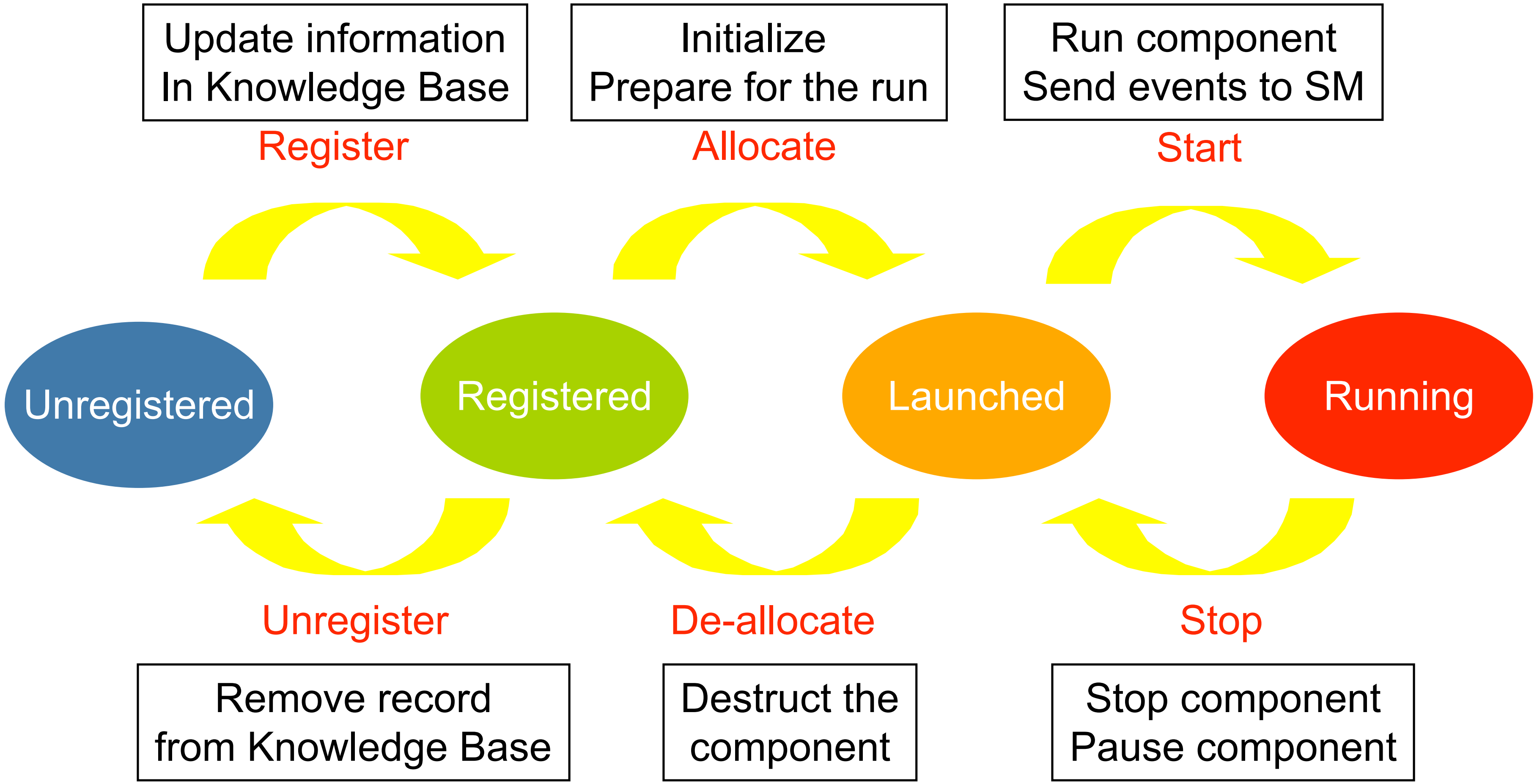
Self-description: The component supervisor provides descriptions of the capabilities (to component registry) and the current state of the process (on request).

Self-Monitoring: The component supervisor estimates state and quality of service for each processing cycle.

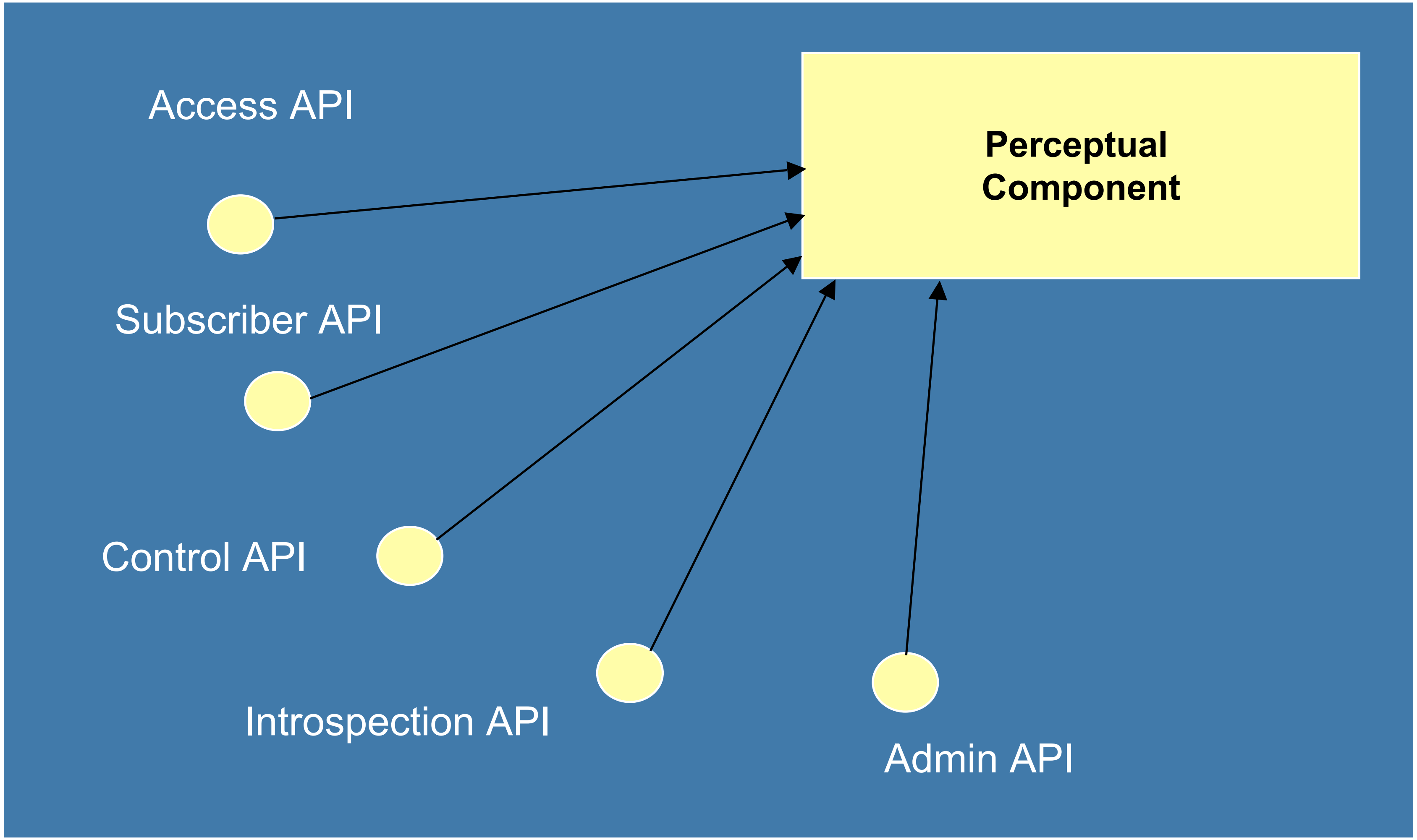
Self-repair: The process controller can detect and correct conditions by reconfiguring modules.

Self-regulation: The component supervisor adapts parameters to maintain a desired process state.

Perceptual Component LifeCycle



Perceptual Components API



Component Registry and Interconnection

O3MiSCID :
Object-Oriented Open-source Middleware for Service Connection, Inspection and Discovery

System Level

- Dynamic discovery of available hardware and software components
- Standardized communication protocol running on multiple platforms
- Support component auto-description and distributed assembly.

Ontological Level

- An XML based ontology for multi-modal perceptual spaces
- Distributed knowledge base for a perceptual environment
- Path discovery with XPath
- Supports dynamic reasoning for automatic component interconnection

Autonomic Properties for Perceptual Components

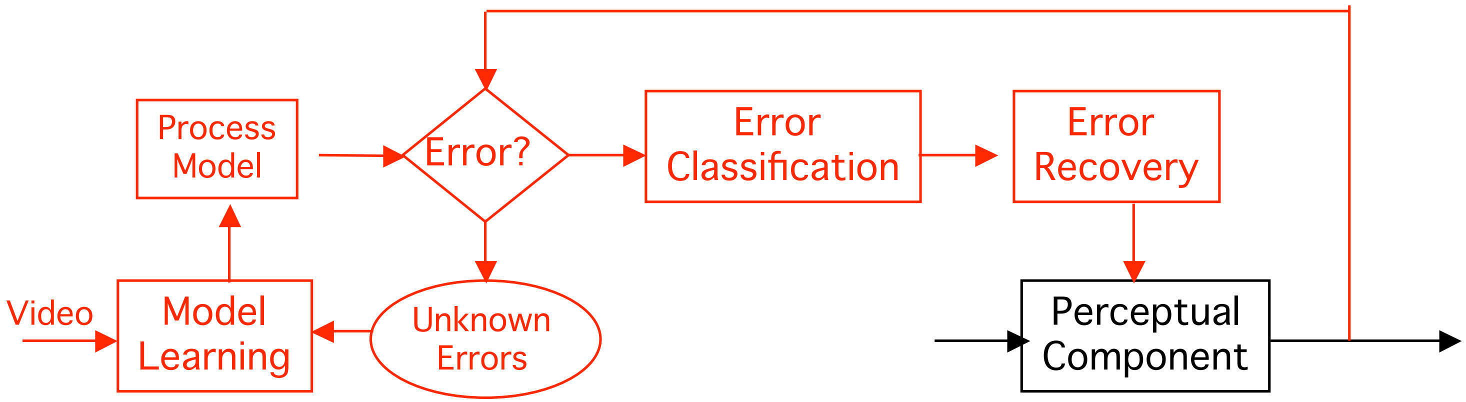
Self-description: The component supervisor provides descriptions of the capabilities (to component registry) and the current state of the process (on request).

Self-Monitoring: The component supervisor estimates state and quality of service for each processing cycle.

Self-repair: The process controller can detect and correct conditions by reconfiguring modules.

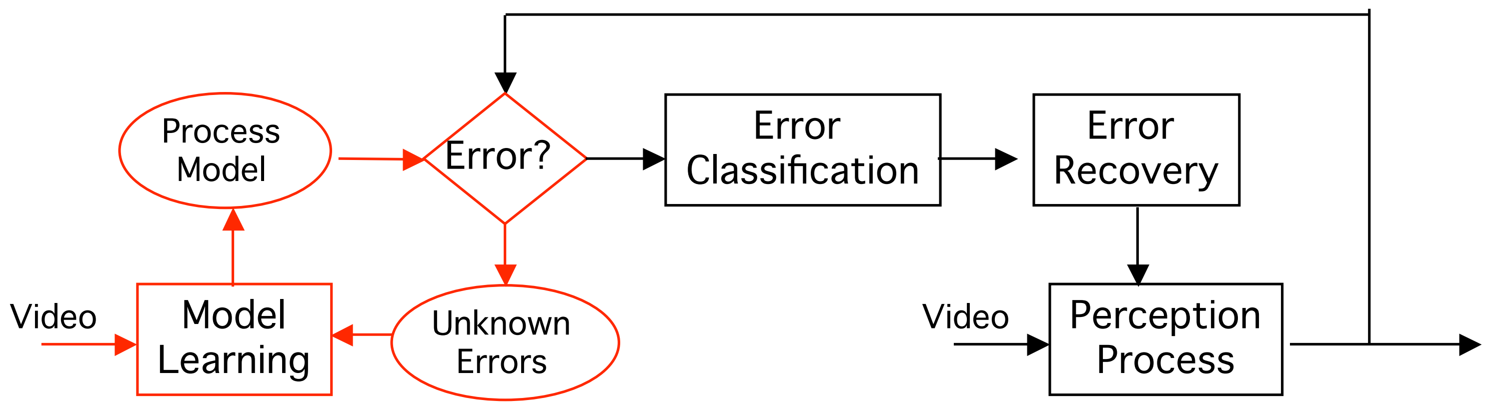
Self-regulation: The component supervisor adapts parameters to maintain a desired process state.

Self-monitoring Perceptual Components



- Component monitors likelihood of output
- When an performance degrades, process adapts processing (modules, parameters, and data)

Training the Process Model



Process Model: histogram for process outputs

Semi-Supervised learning:

- User configures and launches a process.
- System classifies each frame as valid, known error, unknown.
- User validates classification, sequence stored.
- Model updated after each validation.
- Process converges after a few minutes (ex. using CAVIAR indoor testbed).

Autonomic Properties for Perceptual Components

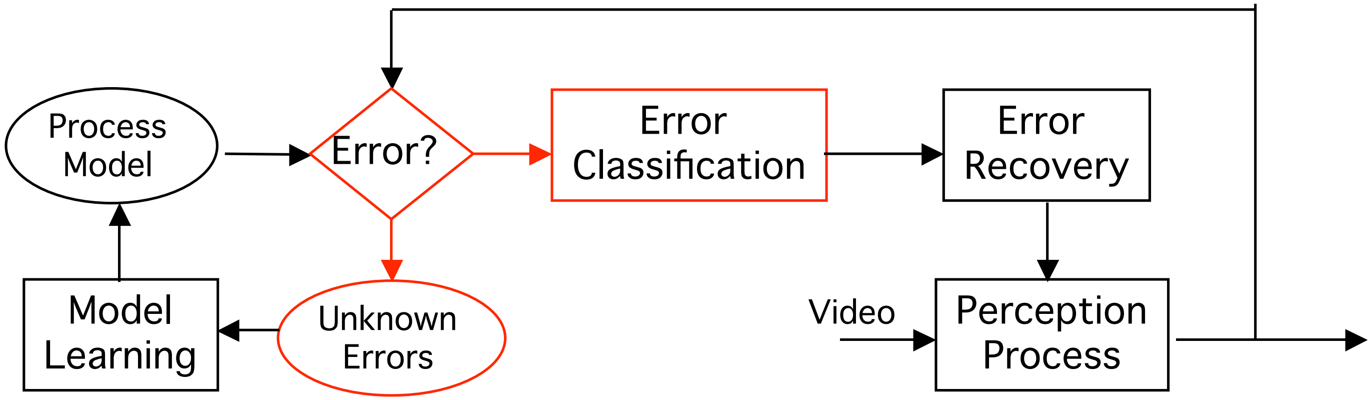
Self-description: The component supervisor provides descriptions of the capabilities (to component registry) and the current state of the process (on request).

Self-Monitoring: The component supervisor estimates state and quality of service for each processing cycle.

Self-repair: The process controller can detect and correct conditions by reconfiguring modules or suppressing targets.

Self-regulation: The component supervisor adapts parameters to maintain a desired process state.

Error Recovery and Self Repair



<u>Two Cases</u>:

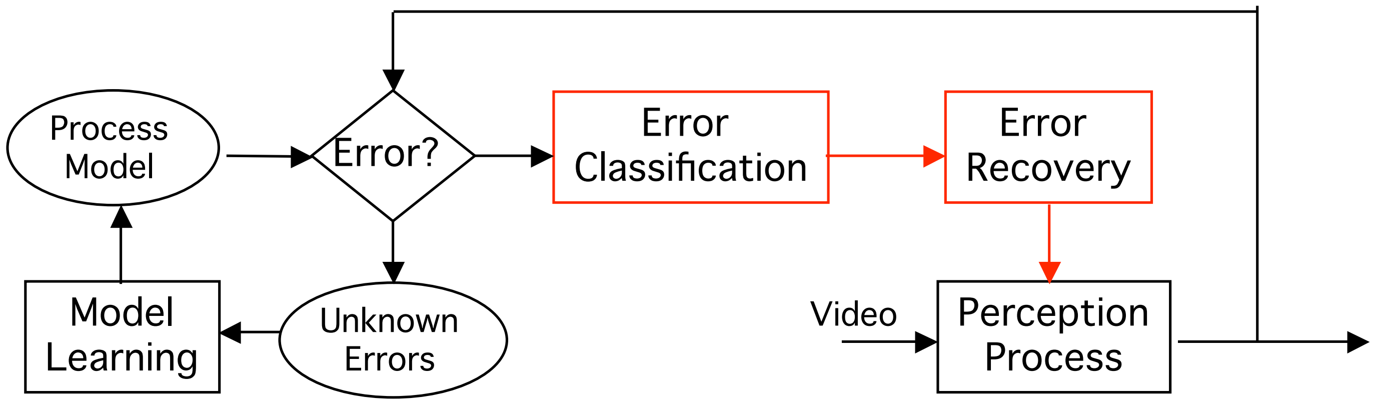
If Error labeled as a known class

- Use repair code for class to reconfigure process.

if Error labeled as *Unknown*

- Store data sequence in data base for off line learning.

Error Recovery and Self Repair



If Error class is recognized, execute error recovery script.

Example Error Recovery Script:

- Change detection method
- Suppress false interpretation
- Merge false split of entities
- Raise/lower detection thresholds

Autonomic Properties for Perceptual Components

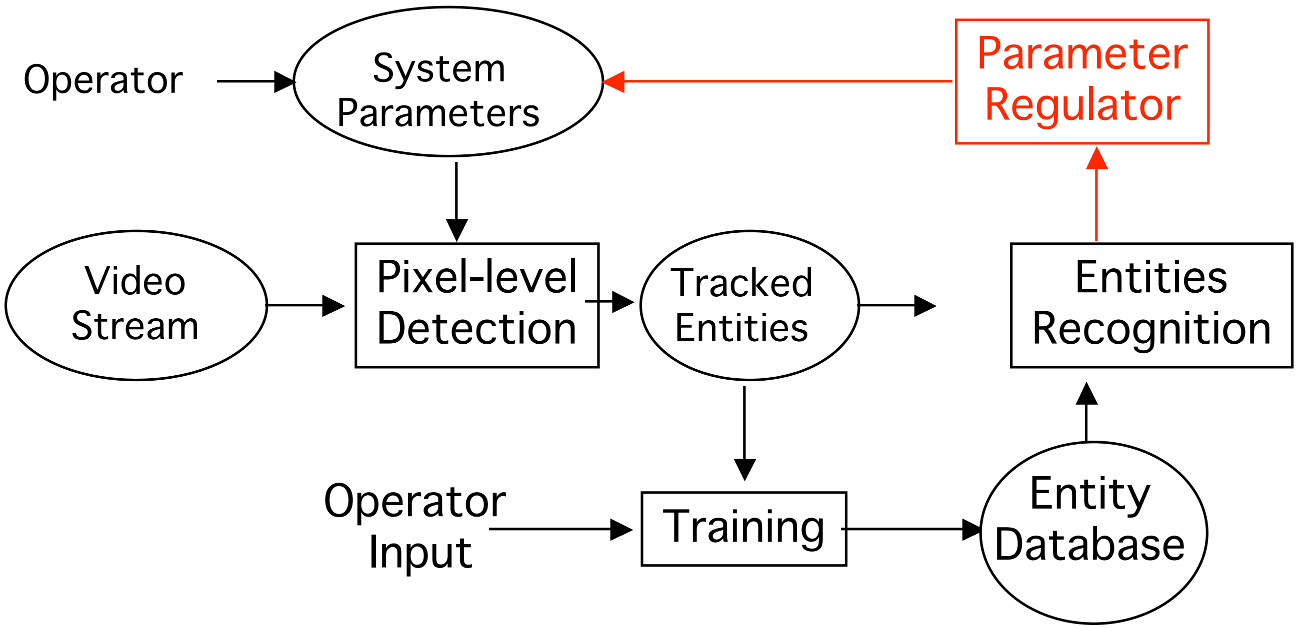
Self-descriptive: The component supervisor provides descriptions of the capabilities (to component registry) and the current state of the process (on request).

Self-Monitoring: The component supervisor estimates state and quality of service for each processing cycle.

Self-repair: The process controller can detect and correct conditions by reconfiguring modules.

Self-regulation: The component supervisor adapts parameters to maintain a desired process state.

Autonomic Parameter Regulation



Parameter regulation provides robust adaptation to Changes in operating conditions.

Parameter Regulation

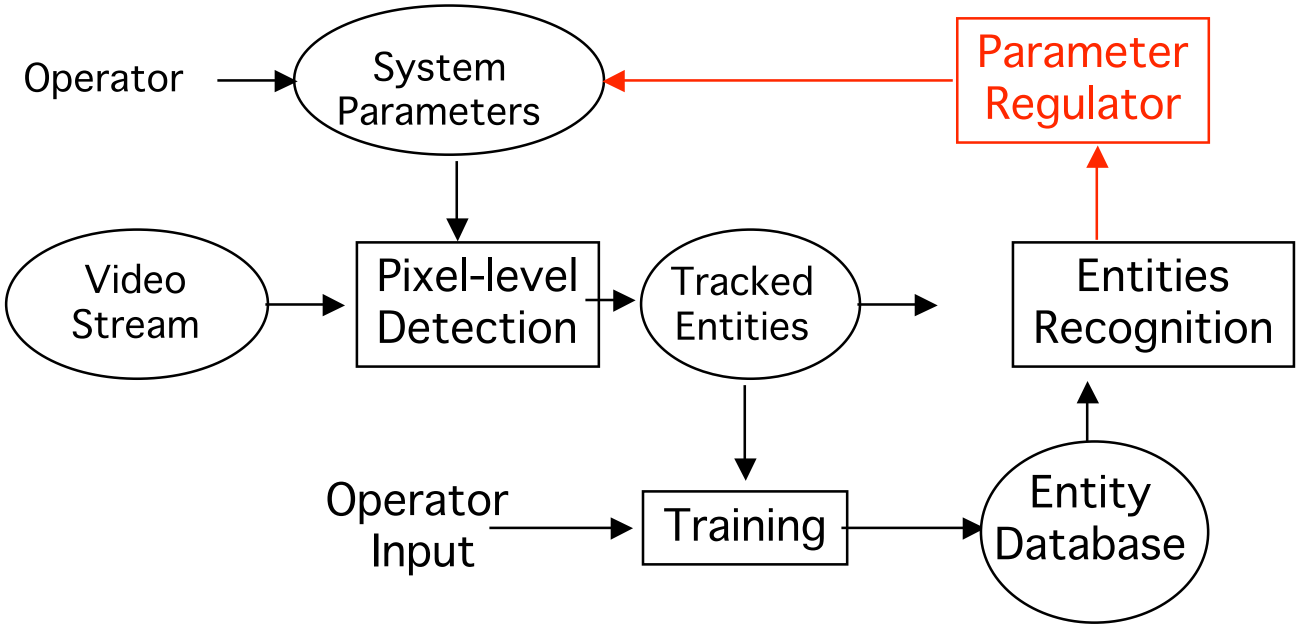
Process parameters depend on environment.

Environmental conditions change.

On-line Regulation:

- Measure quality of service
- Compare to reference
- Tweak parameters (local hill climbing)

Autonomic Processes Regulation



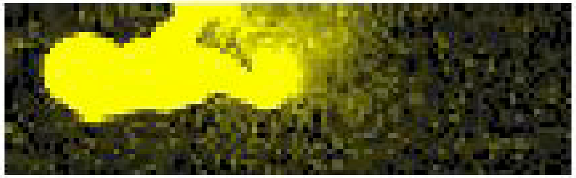
For parameter regulation, we need

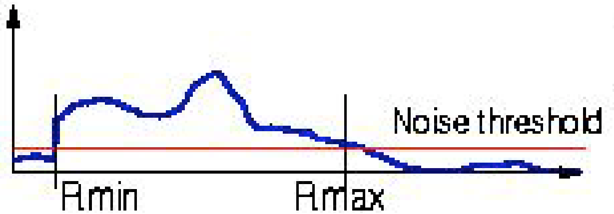
r(t) - Reference Model: Model from Semi-supervised learning

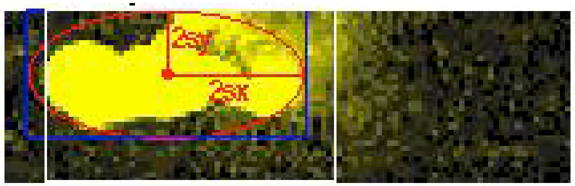
f(y(t)) - Measure: A function of component output

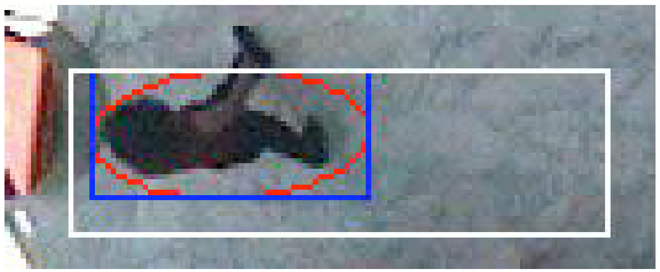
Target Detection

Target Detection









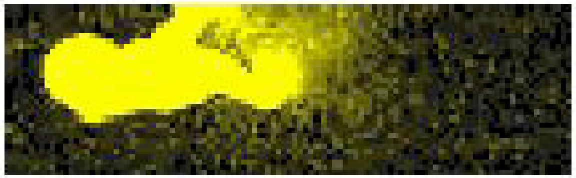
Pixel Level Detection:

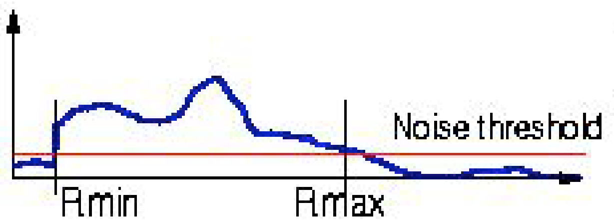
Subtraction from adaptive Background

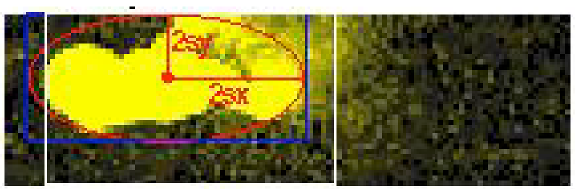
For each Detection ROI

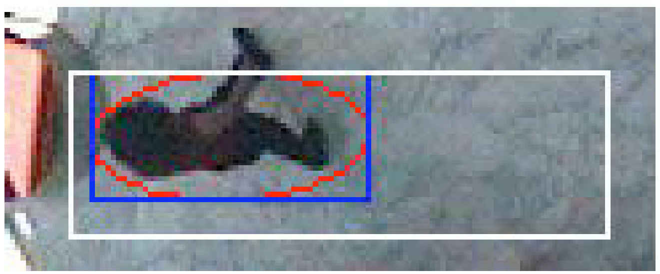
1) Sum detection pixels along rows and along columns. (Two 1-D tables).
2) Determine region where sums are above threshold
3) Sum detection pixels within region
4) If above threshold then
   - Compute moments
   - Create target

Target Detection Parameters









Detection Process:

Two Thresholds:

- Noise Threshold (row and column sums)
- Sensitivy (sum within region)

Problem: How to determine thresholds

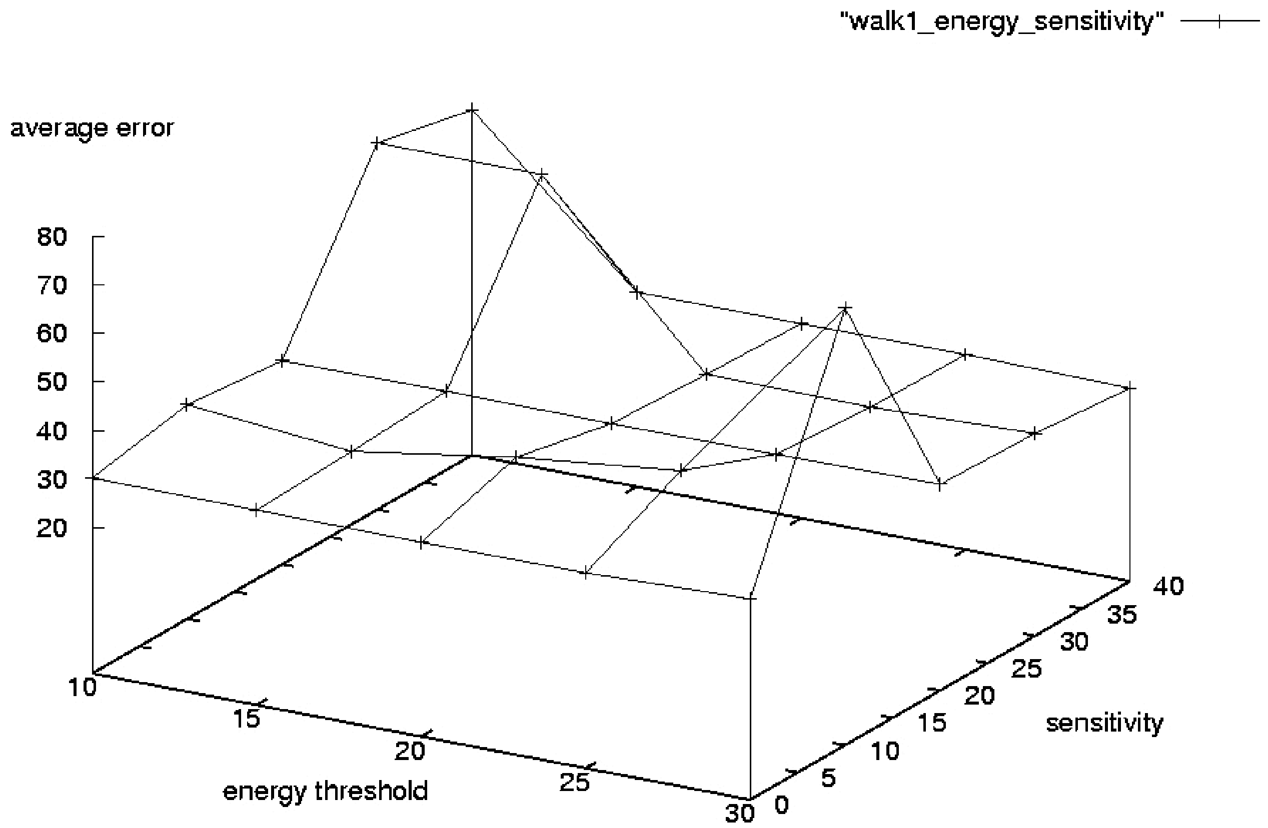
Solution: Use ground truth data as reference

(Ground truth can be obtained by Semi-supervised learning)

Target Detection: PETS 04 Data

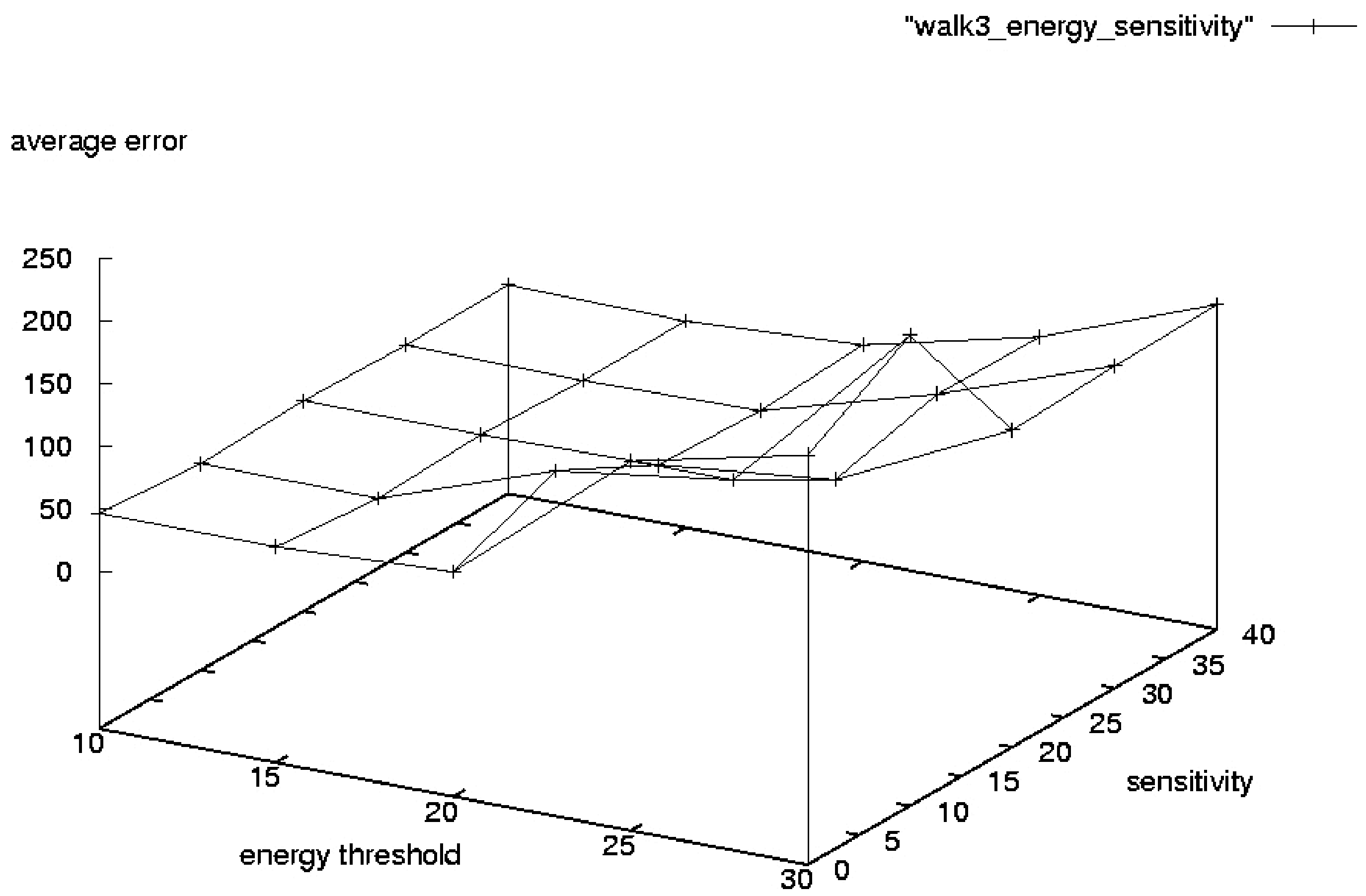
Detection error as a function of threshold and sensitivity

Sequence: Walk1

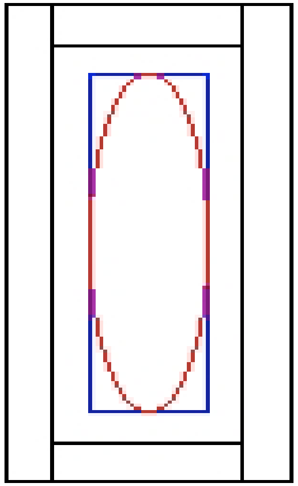


Detection error as a function of threshold and sensitivity

Sequence: Walk3



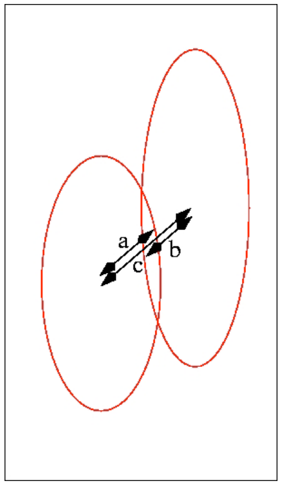
Split and Merge



Split:

Targets surrounded by a detection "halo"

Parameters: Size of Halo, Size of “dead-zone”

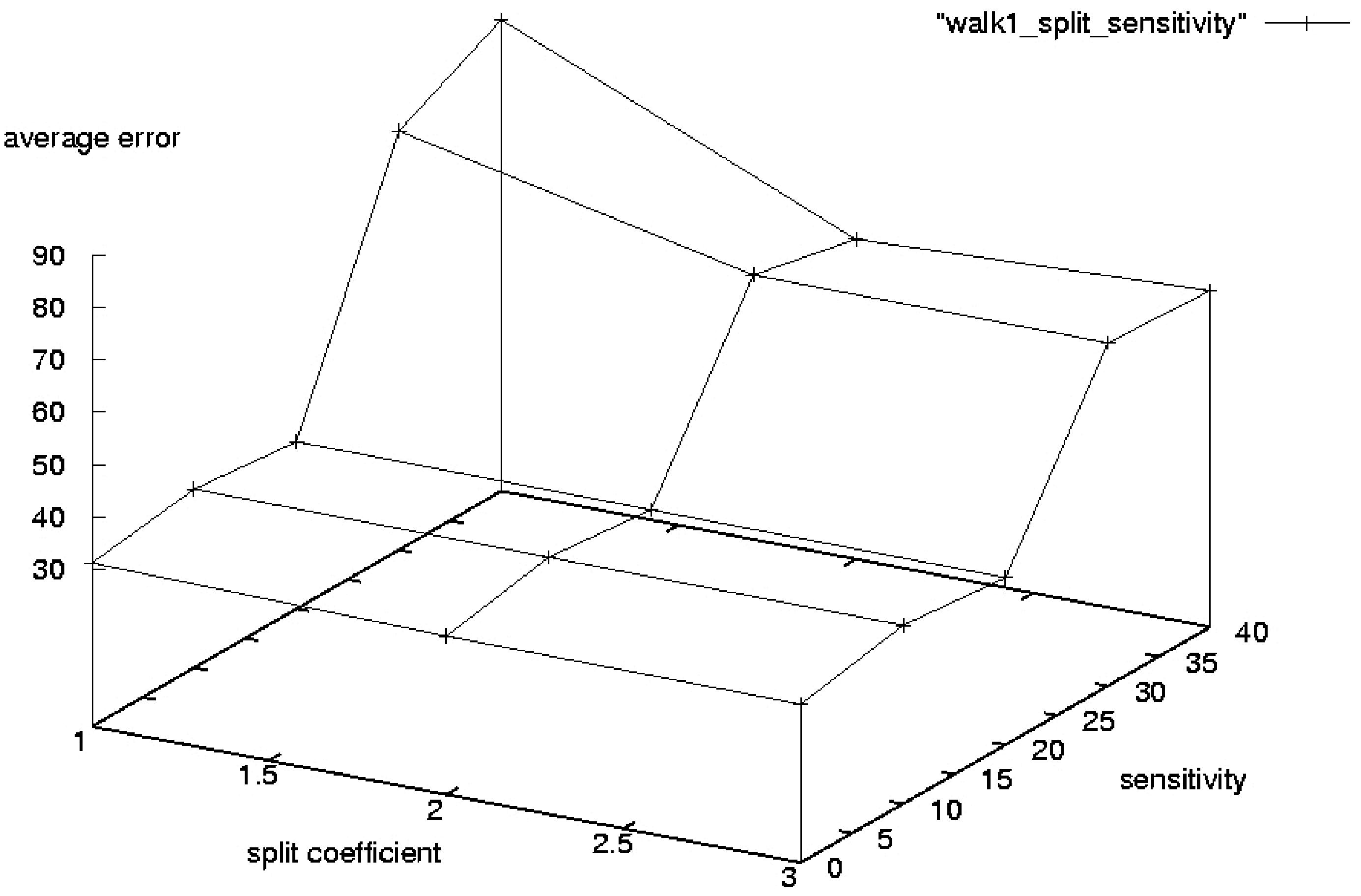


Merge:

Target overlap

Parameter: Mahalanobis distance

Split and Merge



Lessons from Autonomic Vision

Statistical Learning is a powerful tool for Autonomic computing.

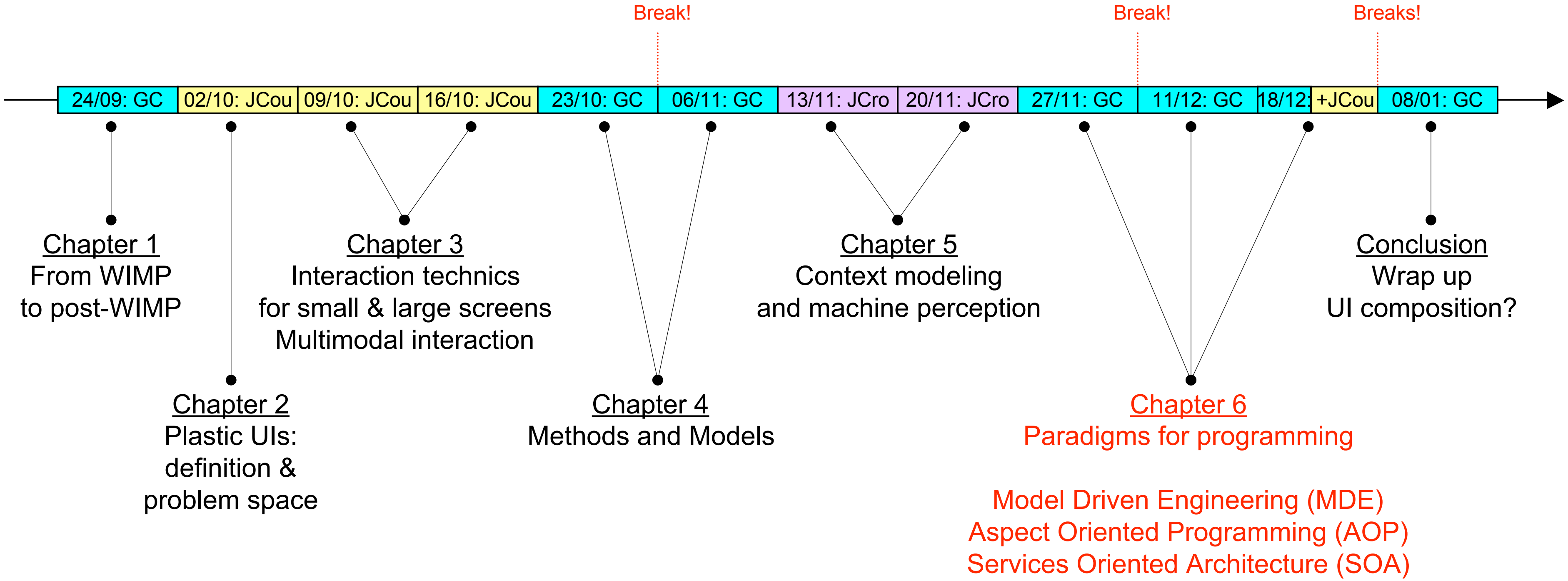
Machine perception is an ideal domain for experimenting with Autonomic Computing.

- Practical Machine Perception requires
  1) Robust Operation
  2) Dynamic reconfiguration.
  3) Adaptation to changes in operating conditions
- Robust operation requires self-monitoring, self-regulation and self-repair
- Dynamic service composition requires self-description and self-assembly

Lesson Plan

1) Introduction: Context Aware Systems and Services
2) Software components for perception, action and interaction
3) Situation Models: a formal foundation for context modeling
4) Acquiring situation models
5) Situated interactive systems and services
6) Autonomic methods for software components

Outline and schedule - What is next?



GC: Gaëlle Calvary
JCou: Joëlle Coutaz
JCro: James Crowley

Mobile and Context-aware Interactive Systems

Gaëlle Calvary, Joëlle Coutaz and James L. Crowley

Master of Science in Informatics at Grenoble
Université Joseph Fourier (Grenoble I)
ENSIMAG / INP Grenoble