Mobile and Context-aware Interactive Systems
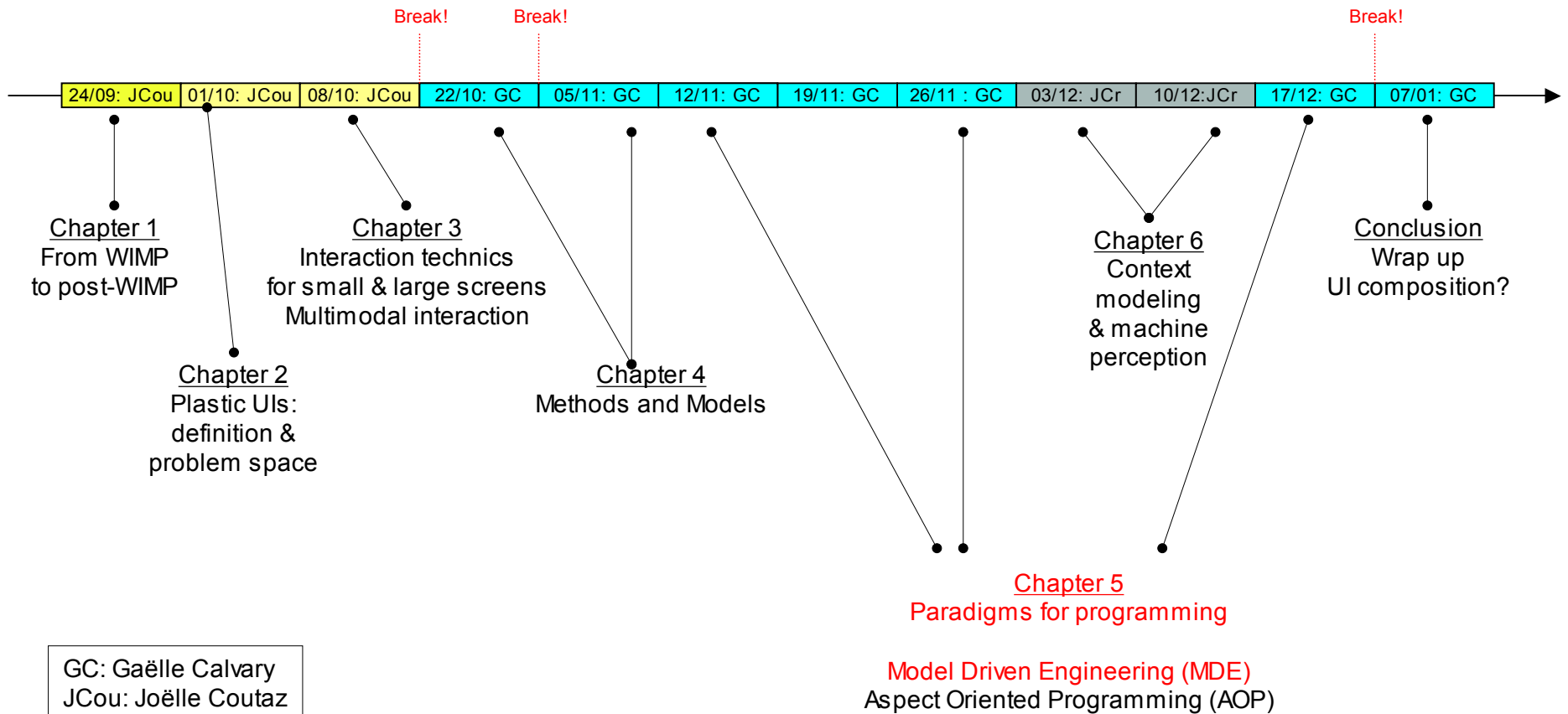
Gaëlle Calvary, Joëlle Coutaz and James L. Crowley

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

Outline and schedule

LIG

Break!   Break!   Break!

24/09: JCou	01/10: JCou	08/10: JCou	22/10: GC	05/11: GC	12/11: GC	19/11: GC	26/11 : GC	03/12: JCr	10/12:JCr	17/12: GC	07/01: GC

Chapter 1
From WIMP
to post-WIMP

Chapter 3
Interaction technics
for small & large screens
Multimodal interaction

Chapter 6
Context
modeling
& machine
perception

Conclusion
Wrap up
UI composition?

Chapter 2
Plastic UIs:
definition &
problem space

Chapter 4
Methods and Models

Chapter 5
Paradigms for programming

Model Driven Engineering (MDE)
Aspect Oriented Programming (AOP)

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

2

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 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
   - Situation Models for Interaction
   - Entities, Relations, and Situation Graphs
   - Roles and Situations
   - Context and Situation.
4) Acquiring situation models
5) Autonomic software components

Situation Models:
a formal foundation for context modeling

Context:

- The situation within which something exists or happens, and that can help explain it [Cup];
- Any information that can be used to characterize situation. [Dey01]

Situation:

- the set of things that are happening and the conditions that exist at a particular time and place. [Cup].

[Cup] Cambridge University on-line dictionary of the English language

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 Models for Interaction

In Theatre:

A script defines a linear sequence of scenes.

Actors use props to play roles

The roles define the space of action for an actor
                    (movements, expressions, etc)

The Script defines the appropriate spoken phrases for each scemne

In human activity

People play roles in shared interaction contexts

Roles define appropriate and inappropriate actions

Social interaction is not linear but includes alternatives and loops.
                    (A network rather than a sequence.)

Social interaction is modeled as a Situation Graph

Situation Graph

Situation Graph

```
Situation-1 ──→ Situation-3 ──→ Situation-5

Situation-2    Situation-4    Situation-6
```
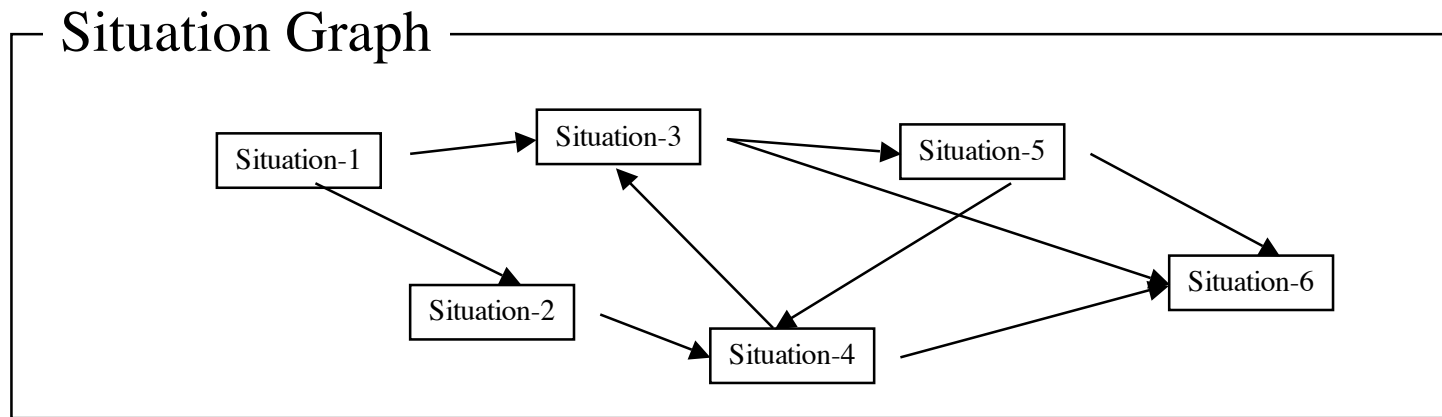
A <u>situation graph</u> describes a state space of situations

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 |

AIBO Behaviours
  Be Quite (10)
  Play (-5)
  Talk to John (-10)
  Dance (-10)
  Sing (-20)

AIBO Behaviours
  Be Quite (5)
  Play (0)
  Talk to John (-5)
  Dance (-10)
  Sing (-20)

AIBO Behaviours
  Be Quite (-10)
  Play (5)
  Talk to John (10)
  Dance (5)
  Sing (-5)
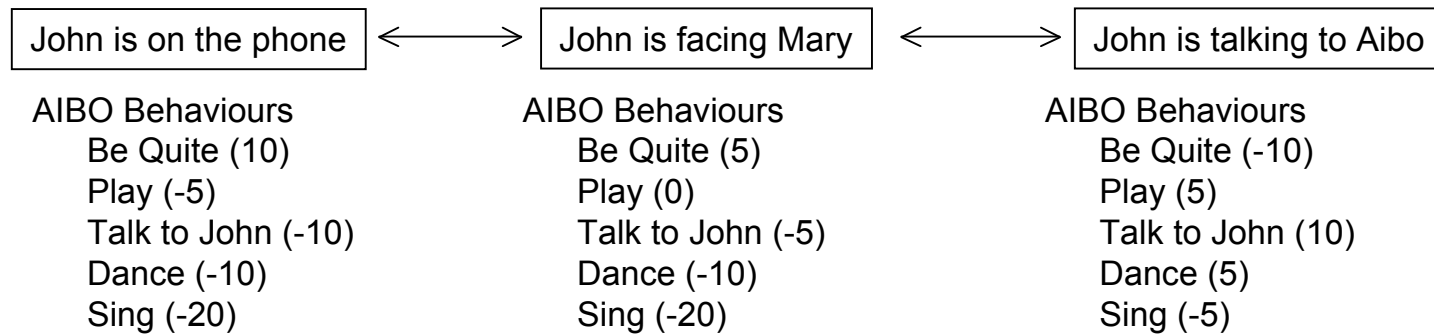
Each situation indicates:

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

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.

Roles and Situations

| <X> is on the phone | ⟷ | <X> is on the phone |

| Role <X> = John | | Role <X> = Mary |
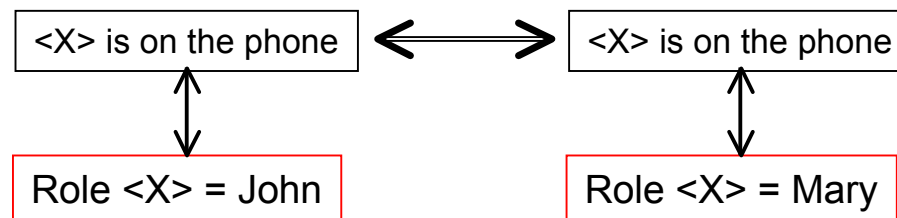
A <u>role</u> is a "variable" for entities.

Roles allow generalizations of situations.

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

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 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
   - The acquisition problem
   - Hand-Crafting situation models
   - Off-line: Learning a Generic Situation Model
   - On-line: Accommodating Individual Preferences
5) Autonomic software components

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

<u>The Knowledge Barrier</u>:

The extreme complexity of human activity and individual preferences

Proposed Solution

<u>Machine Learning</u>

<u>Off-line</u>: Learning of prototype scripts

<u>On-line</u>: Adaptation of scripts to accommodate preferences

Learning Situation Models

Four Learning Problems

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 service Behaviour |
| ⇑ |
| Learning Situation Graph |
| ⇑ |
| Situation Discovery |
| ⇑ |
| Learning Role Detection |
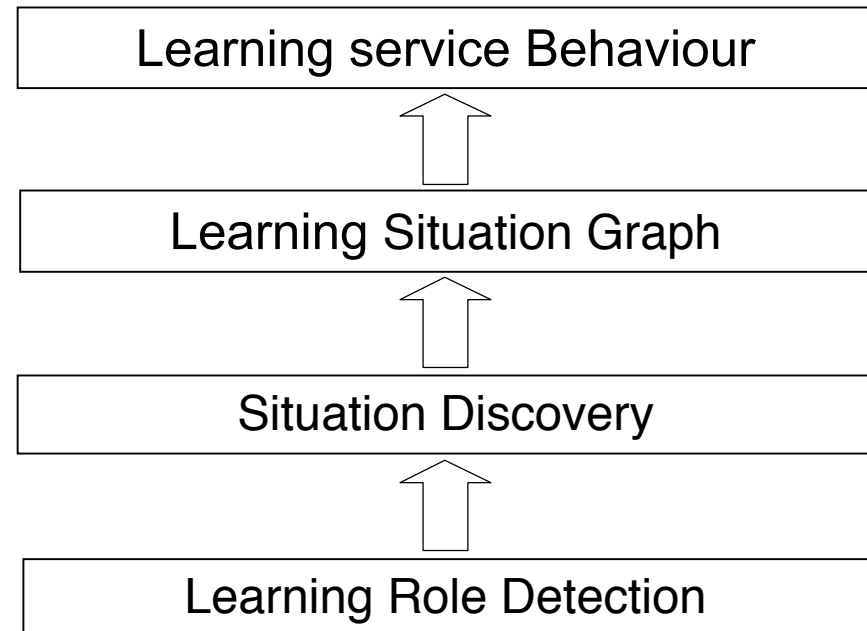
Learning Role Assignment

Four Learning Problems

Learning Service Behaviour
⇒ Supervised on-line Learning

Learning Situations Graphs
⇒ Supervised off-line Learning
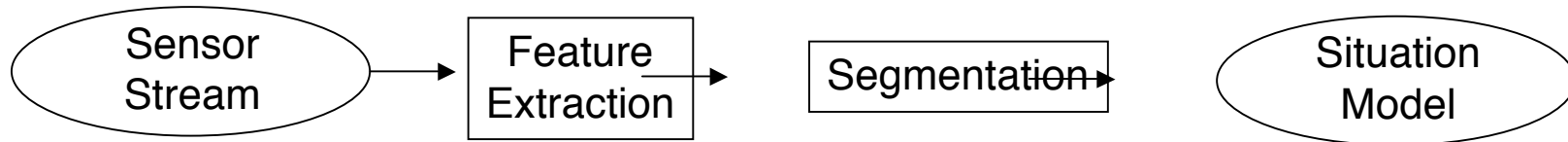
Situation Discovery
⇒ Unsupervised off-line Learning

Role Assignment
⇒ Off-line Statistical Learning

```
┌─────────────────────────────────┐
│  Learning service Behaviour     │
└─────────────────────────────────┘
              ⇧
┌─────────────────────────────────┐
│   Learning Situation Graph      │
└─────────────────────────────────┘
              ⇧
┌─────────────────────────────────┐
│      Situation Discovery        │
└─────────────────────────────────┘
              ⇧
┌─────────────────────────────────┐
│   Learning Role Assignment      │
└─────────────────────────────────┘
```
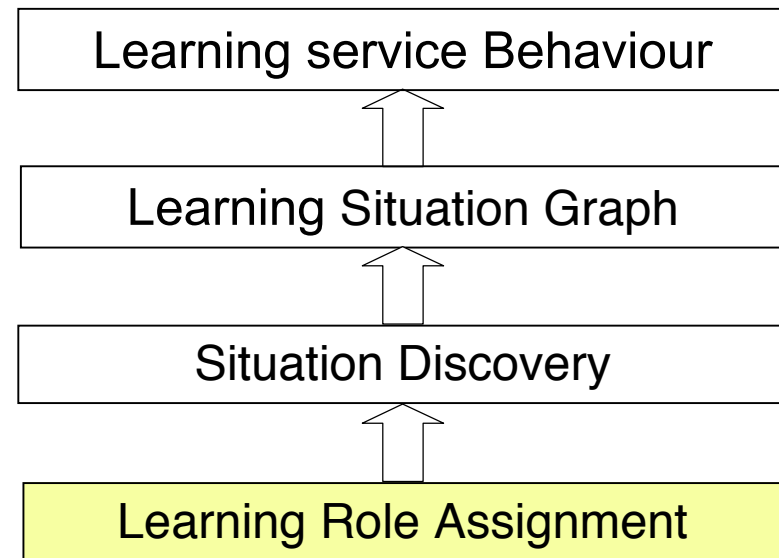
Role Detection: Supervised learning using SVMs

Learning Role Assignment

Sensor Stream → Feature Extraction → Supervised Learning → Role Assignment Tests
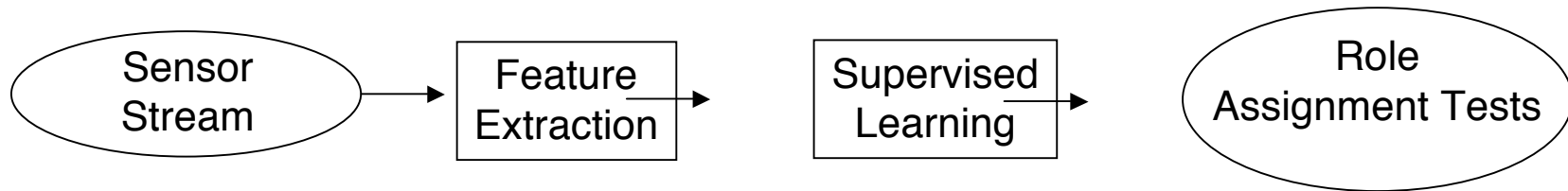
Approach:

1) Visual and acoustic tracking

2) Compute feature stream

3) Hand Label examples

4) Train Role Recognition Classifiers
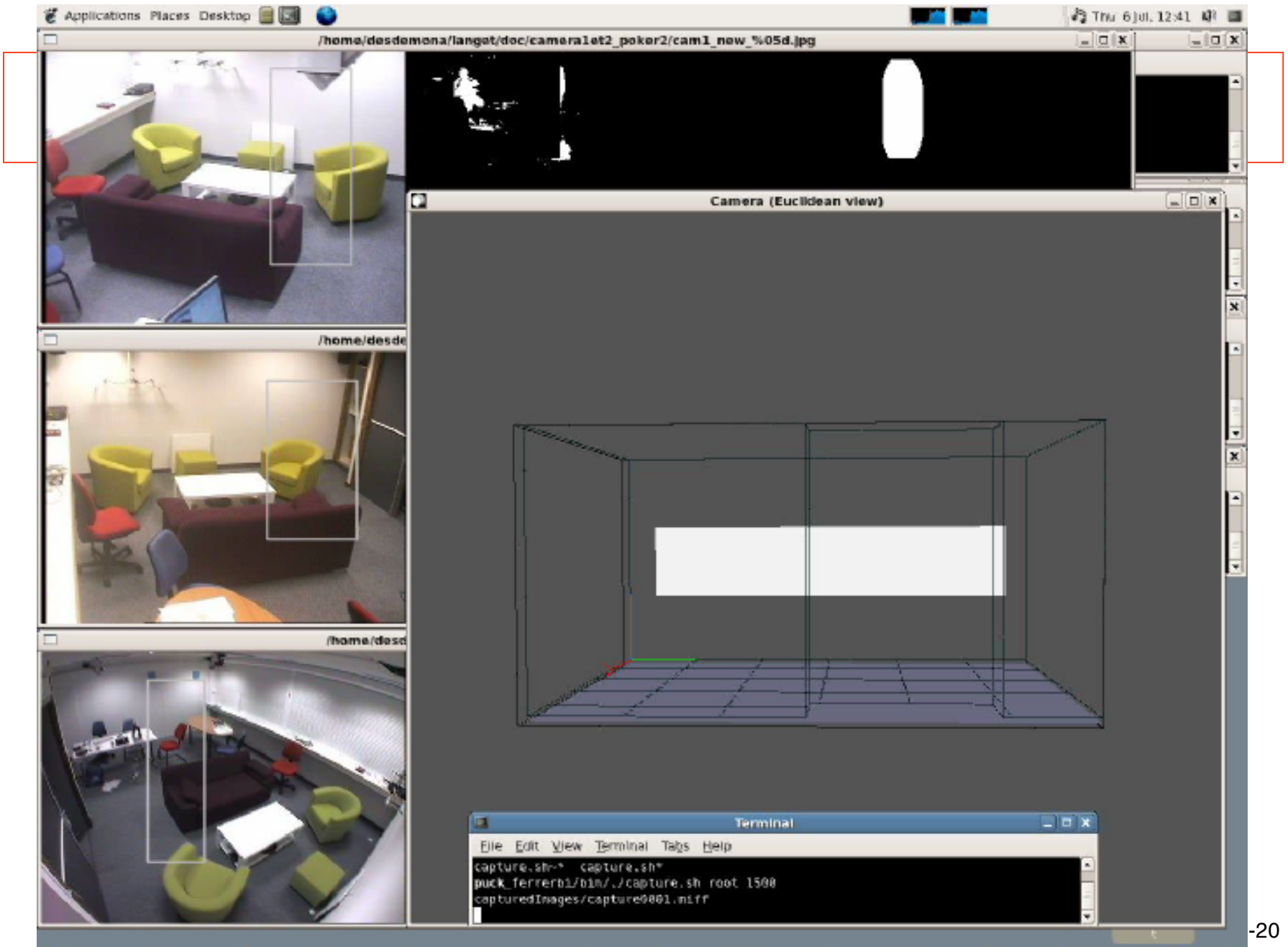
Simple Features from Tracker

1. Acoustic and Visual Blob tracking

2. <u>Features</u> 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.
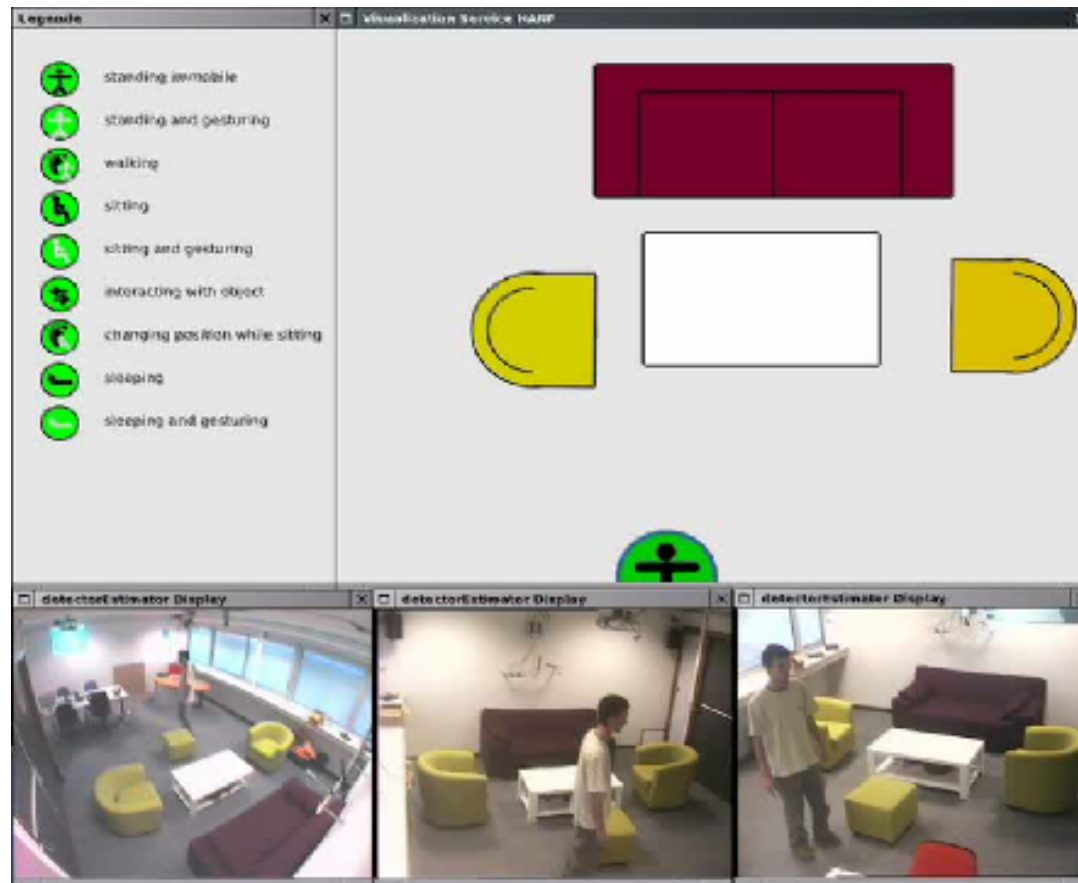
Learning Situation Models

Four Learning Problems

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

Learning service Behaviour
⇑
Learning Situation Graph
⇑
Situation Discovery
⇑
Learning Role Assignment
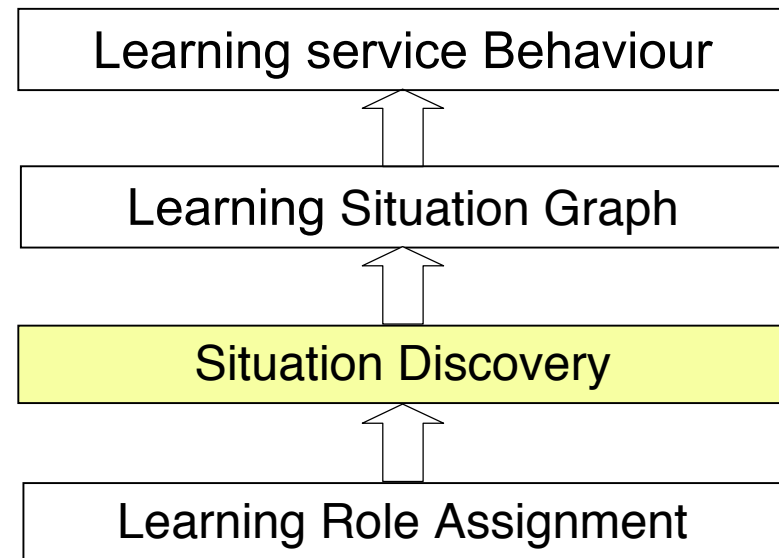
Initial Situation Model: Learned by Segmenting Feature Stream

Learning an Initial Situation Model

```
┌─────────┐        ┌──────────┐                        ┌──────────┐
│ Sensor  │───────▶│ Feature  │───▶│Segmentation│───▶  │Situation │
│ Stream  │        │Extraction│                        │  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
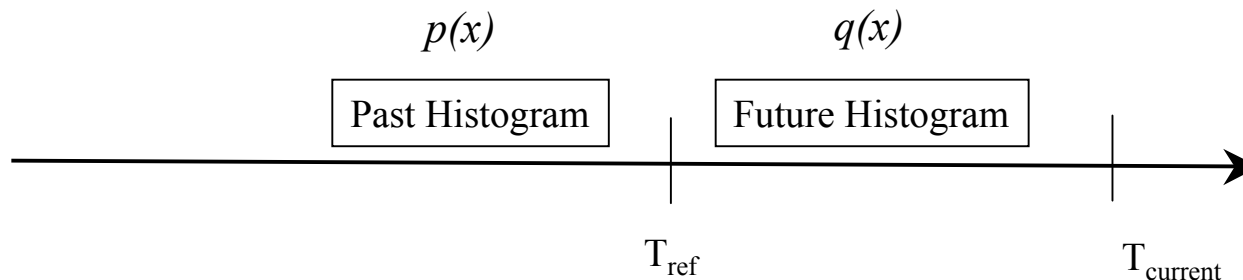
$$J_{p,q} = \sum_{x \in X} p(x) \cdot \log \frac{p(x)}{\dfrac{p(x) + q(x)}{2}} + q(x) \cdot \log \frac{q(x)}{\dfrac{p(x) + q(x)}{2}}$$

Segmenting Feature Stream

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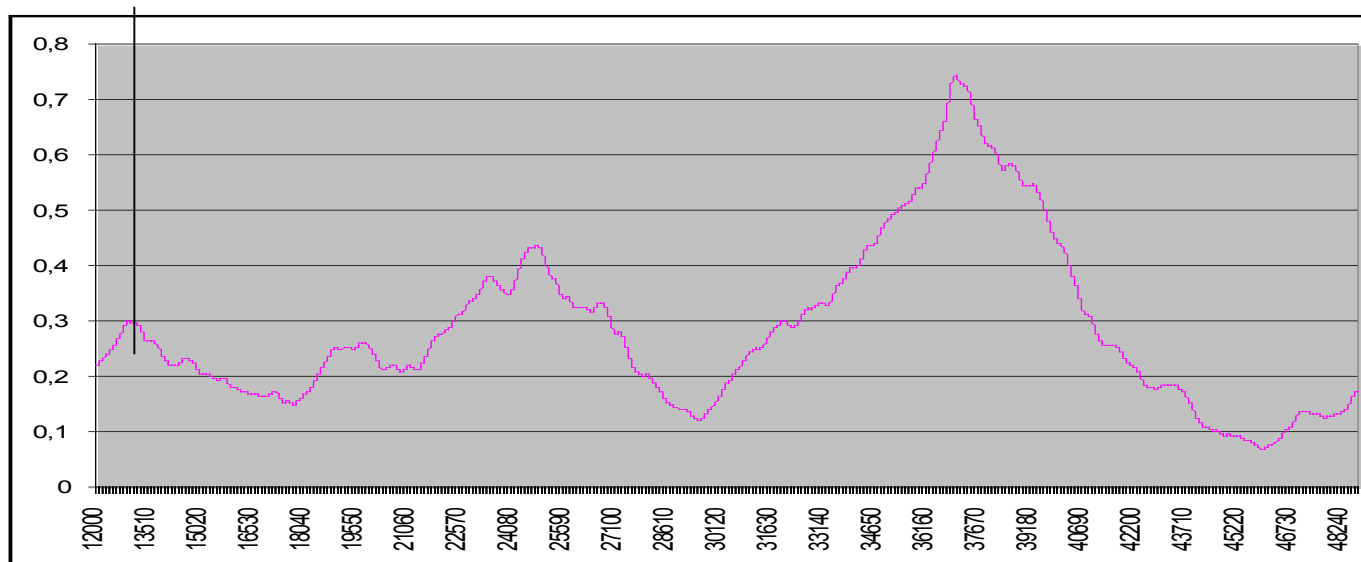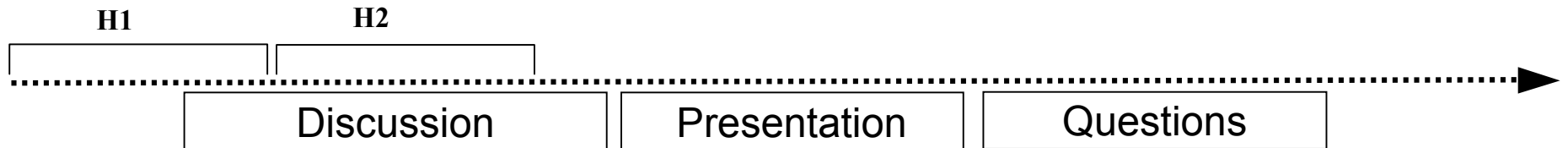
Segmenting Feature Stream

multi-scale analysis: Jeffrey divergence curves for different window sizes
4000-16000 observations (between 64sec and 4min 16sec)

Jeffrey divergence curve (histogram windows size=4000 obs.)

Jeffrey divergence curve (histogram windows size=12000 obs.)
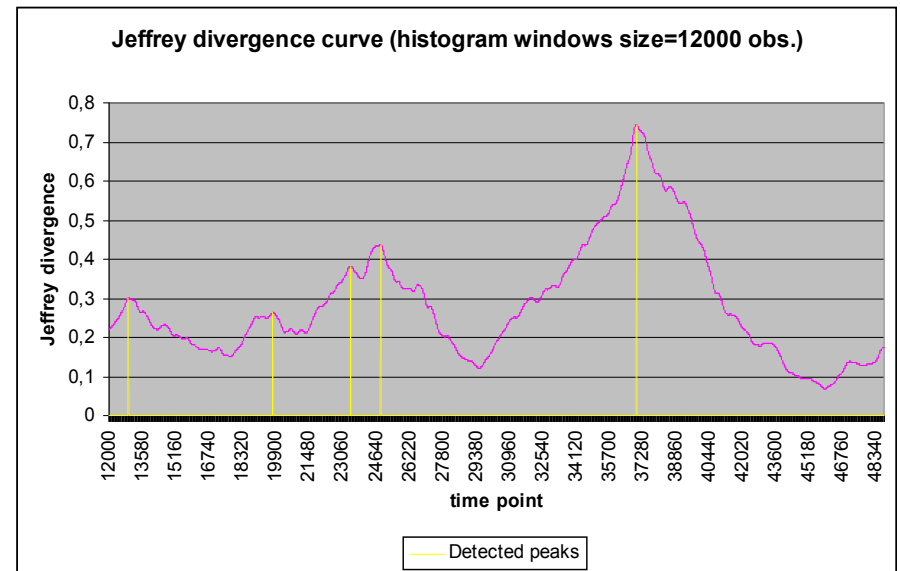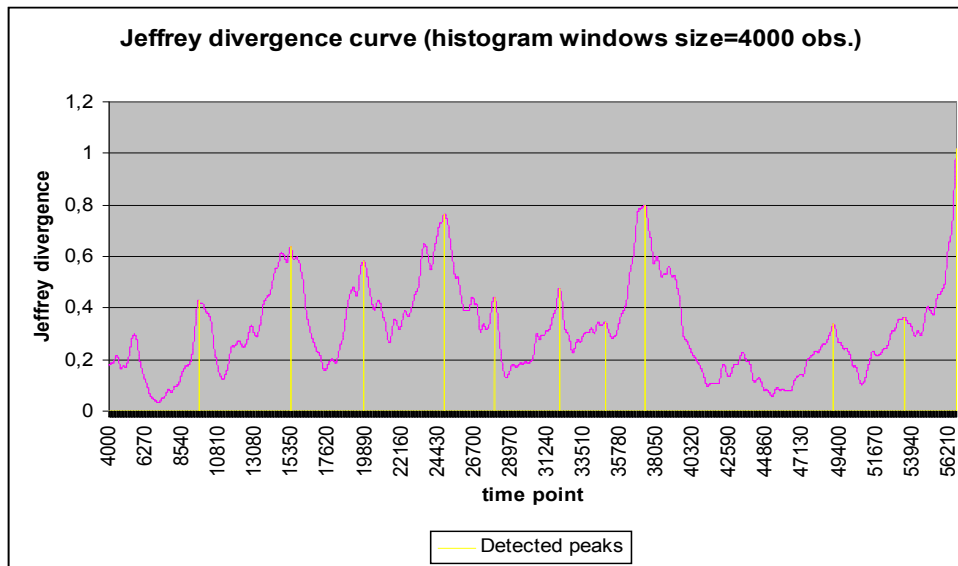
Learning Situation Models

Four Learning Problems

Role Detection
  ⇒ Statistical Learning

Situation Discovery
  ⇒ Unsupervised Learning

Learning Situations Graphs
  ⇒ Supervised Learning

Learning service Behaviour
  ⇒ Supervised Learning

Learning service Behaviour

⇑

Learning Situation Graph

⇑

Situation Discovery

⇑

Learning Role Detection
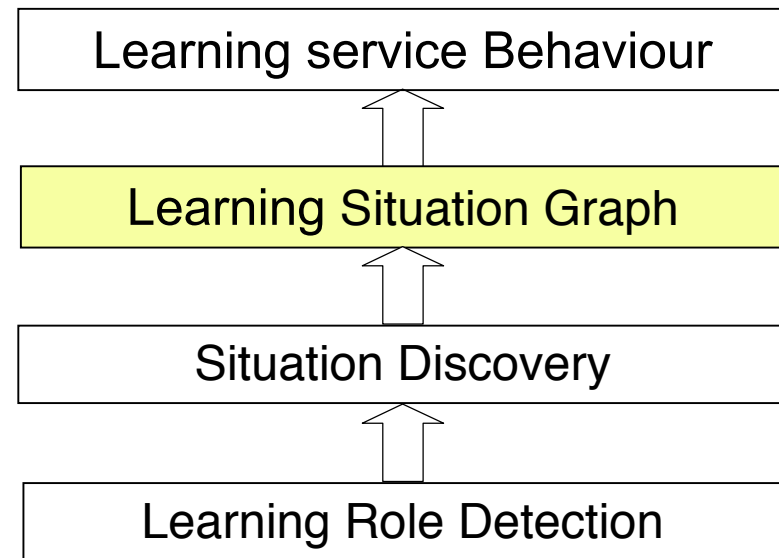
Supervised Situation Learning

n observation sequences associated to *m* situation labels (*m≤n*)
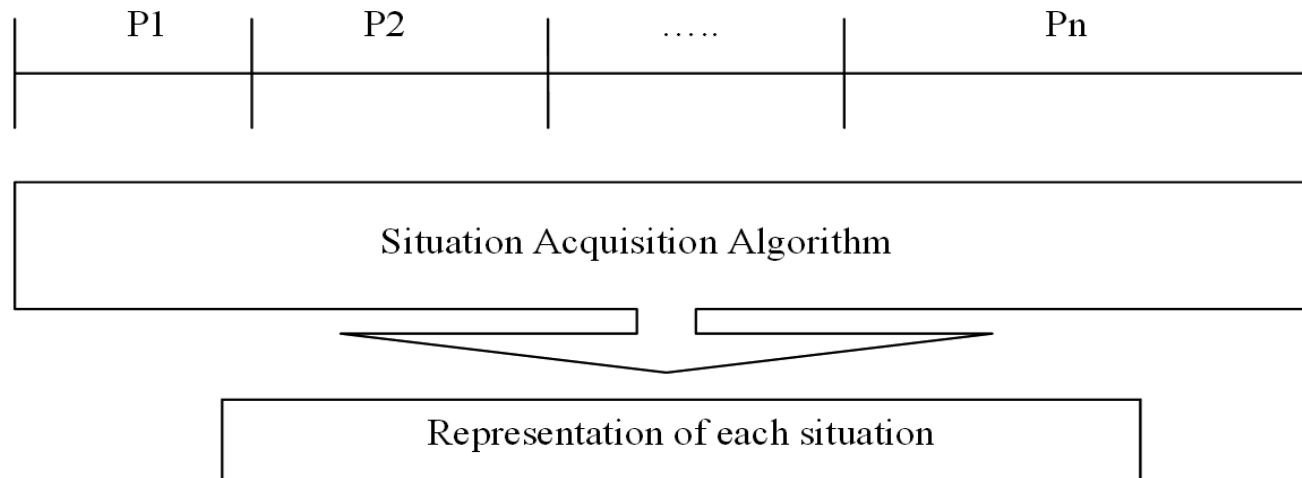
Each sequence corresponds to one situation

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

$\Rightarrow$ Supervised on-line Learning

Learning Situations Graphs

$\Rightarrow$ Supervised off-line Learning

Situation Discovery

$\Rightarrow$ Unsupervised off-line Learning

Role Assignment

$\Rightarrow$ Off-line Statistical Learning

Learning service Behaviour
⇧
Learning Situation Graph
⇧
Situation Discovery
⇧
Learning Role Detection
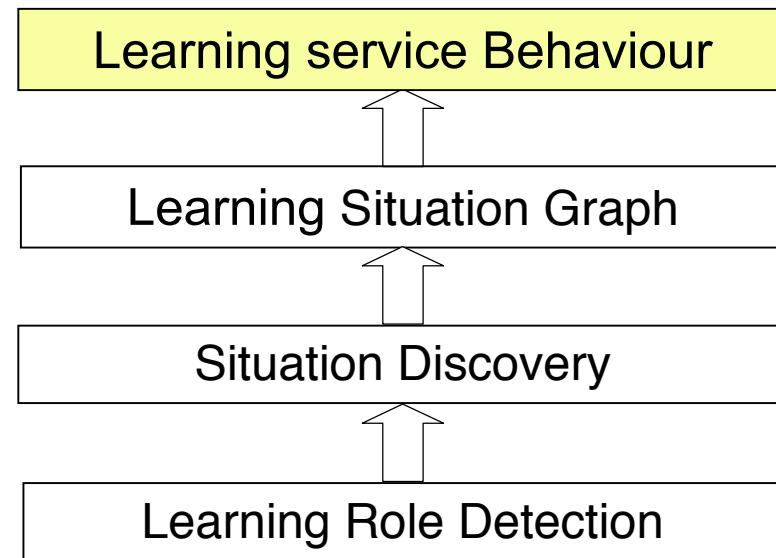
Split and Merge Situations from User Feedback

On-line: Adapting to User Preferences
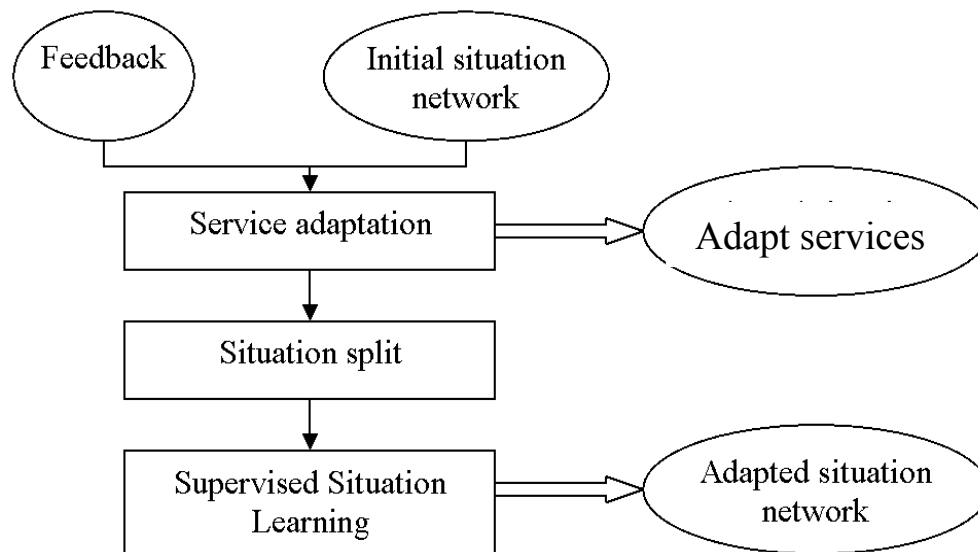
1. Modify association of service with situations

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

Sit 1 → *Split* →
- Sit 1a
- Sit 1b
- Sit 1c

S_1

S_{1a}

S_{1b}

S_{1c}
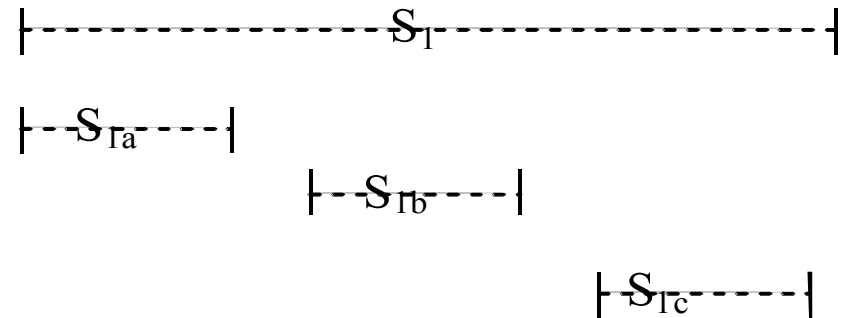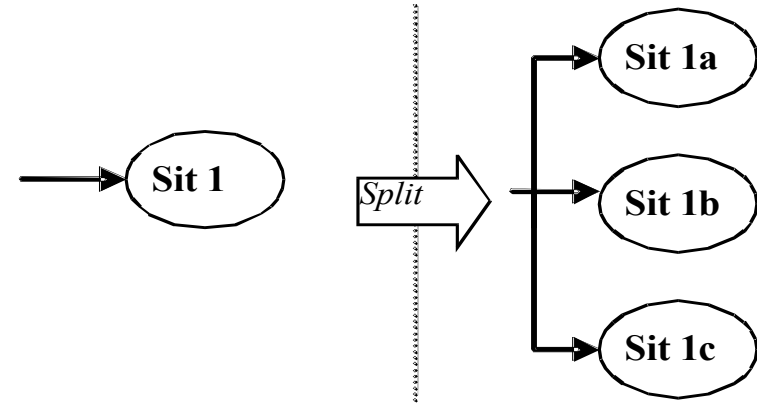
Multimodal observation of the scene

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

Experiment

```
Learning service Behaviour
        ↑
Learning Situation Graph
        ↑
   Situation Discovery
        ↑
  Learning Role Detection
```
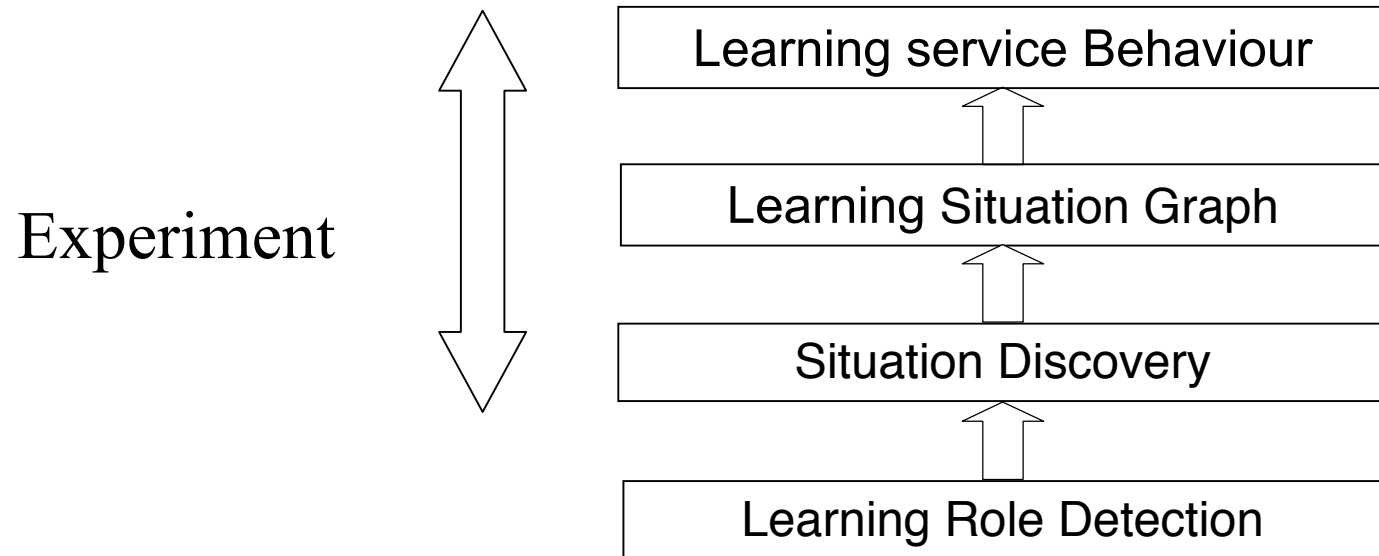
Experimental Demonstration

3 scenarios recordings

Situations: "introduction", "aperitif",
"siesta", "presentation",
"game"
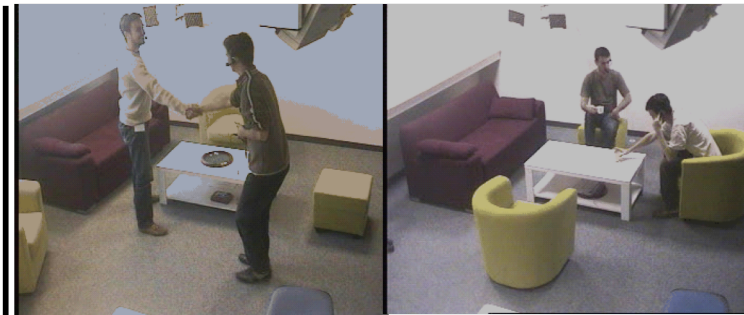
Introduction   Aperitif
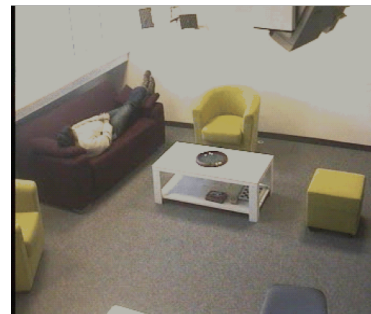
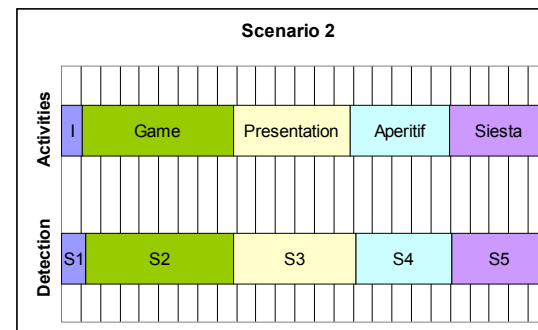Siesta   Presentation   Game

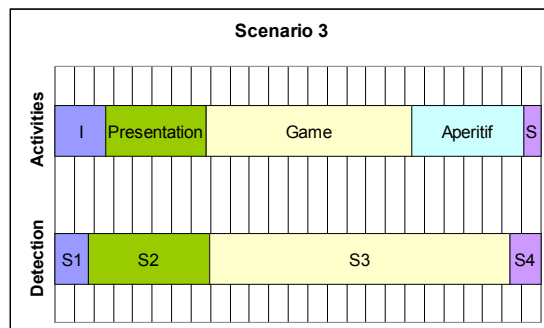"online" in-scenario situation recognition

Experimental Demonstration

Off-line: unsupervised situation discovery for protype situations

Scenario 1

| Activities | I | Aperitif | Game | Presentation | Siesta |
| Detection | S1 | S2 | S3 | S4 |

Q = 0.68

Scenario 2

| Activities | I | Game | Presentation | Aperitif | Siesta |
| Detection | S1 | S2 | S3 | S4 | S5 |

Q = 0.95

Scenario 3

| Activities | I | Presentation | Game | Aperitif | S |
| Detection | S1 | S2 | S3 | S4 |
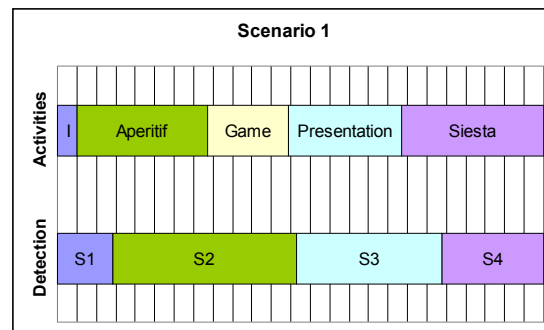
Q = 0.74

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)

Situation Recognition Rate

(x-axis: recognition window size (frames); y-axis: recognition rate (%))
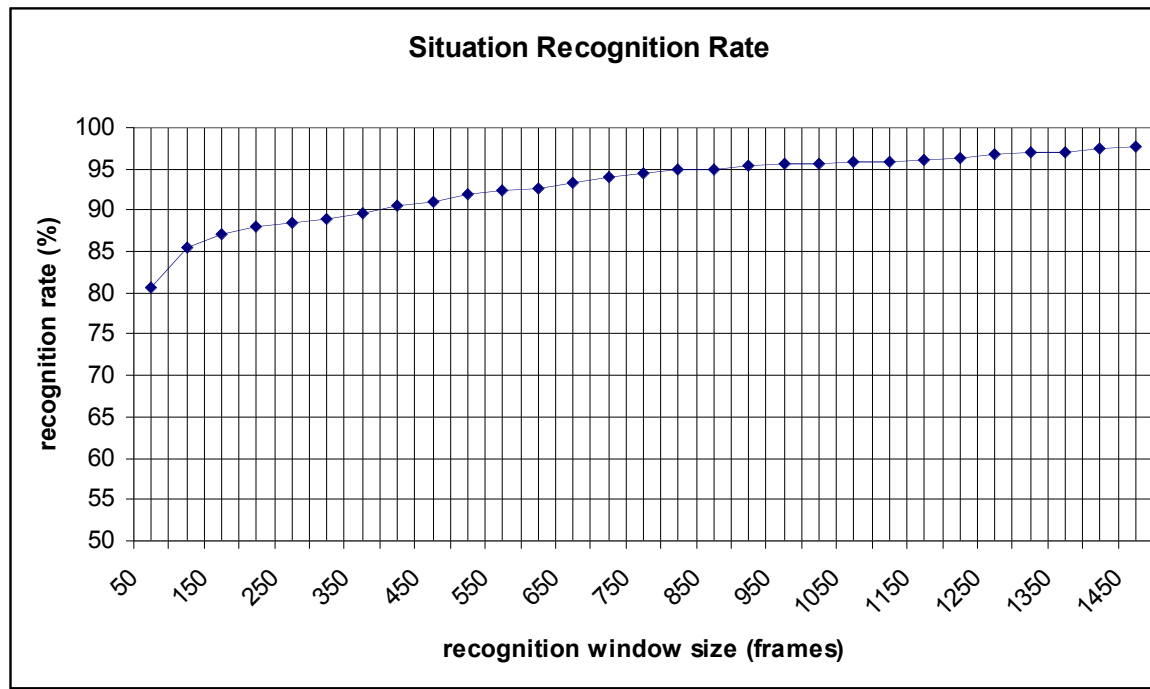
Experimental Demonstration

On-line: Integration of user preferences

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

Situation Recognition Rate
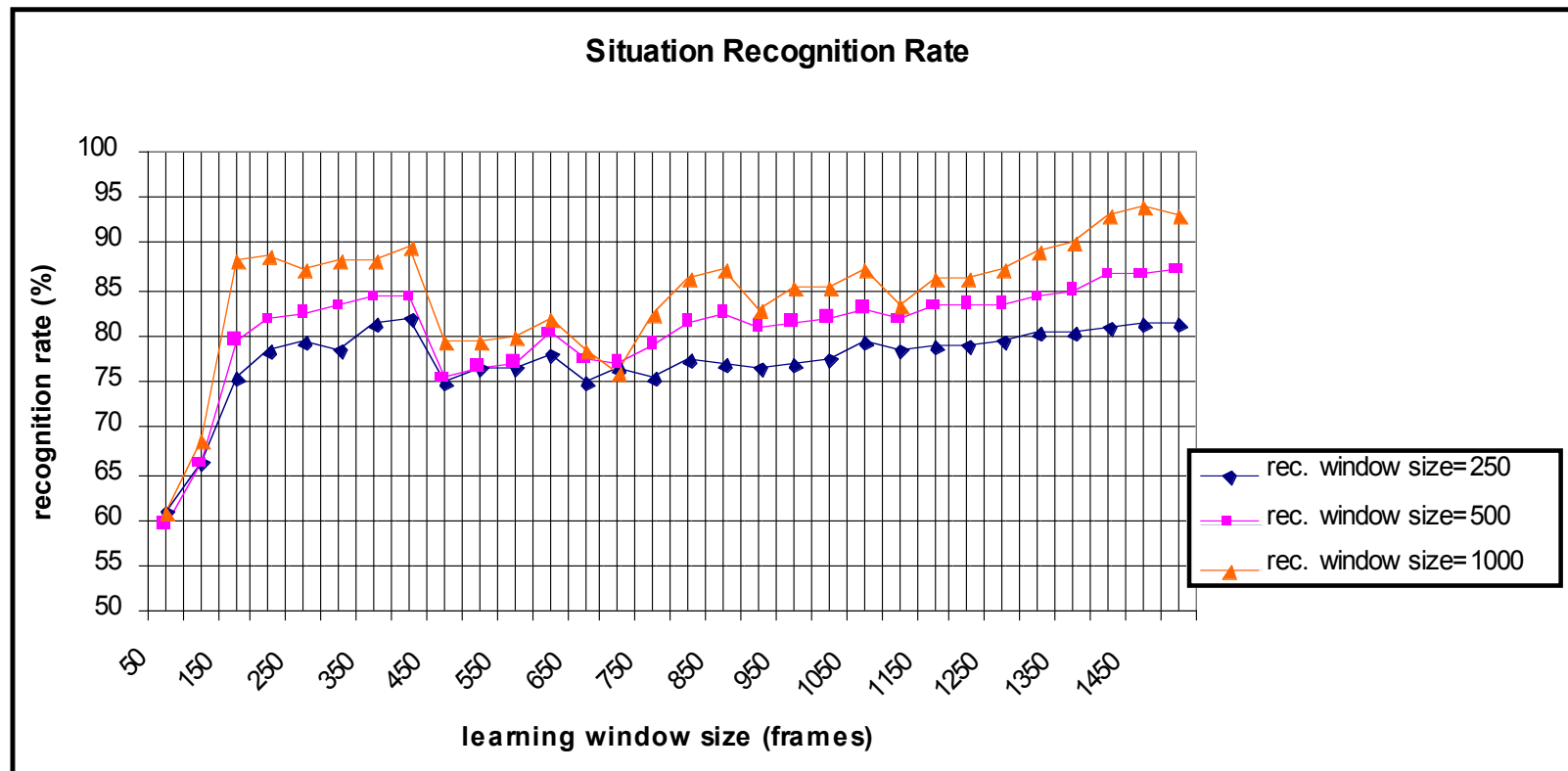
Summary and Conclusions

Situation networks provide scripts for services

Fundamental Problem: Knowledge Barrier

Proposed solution: machine learning

- Off-line: Learn prototype "generic" scripts.
    - þ Initial situation discovery by unsupervised segmentation
    - þ Supervised learning to build situation networks

- On-line: Adapt Networks and services to preferences
    - þ Use feedback from people for on-line adaptation of situation models and services
    - þ Split and merge Situation networks.
    - þ 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 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 software components

  • Need for Autonomic Components

  • Origins of Autonomic Computing

  • Autonomic Properties

  • Methods for building Autonomic perception/action components

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 Activity Sensor
(PETS 2002 Data)
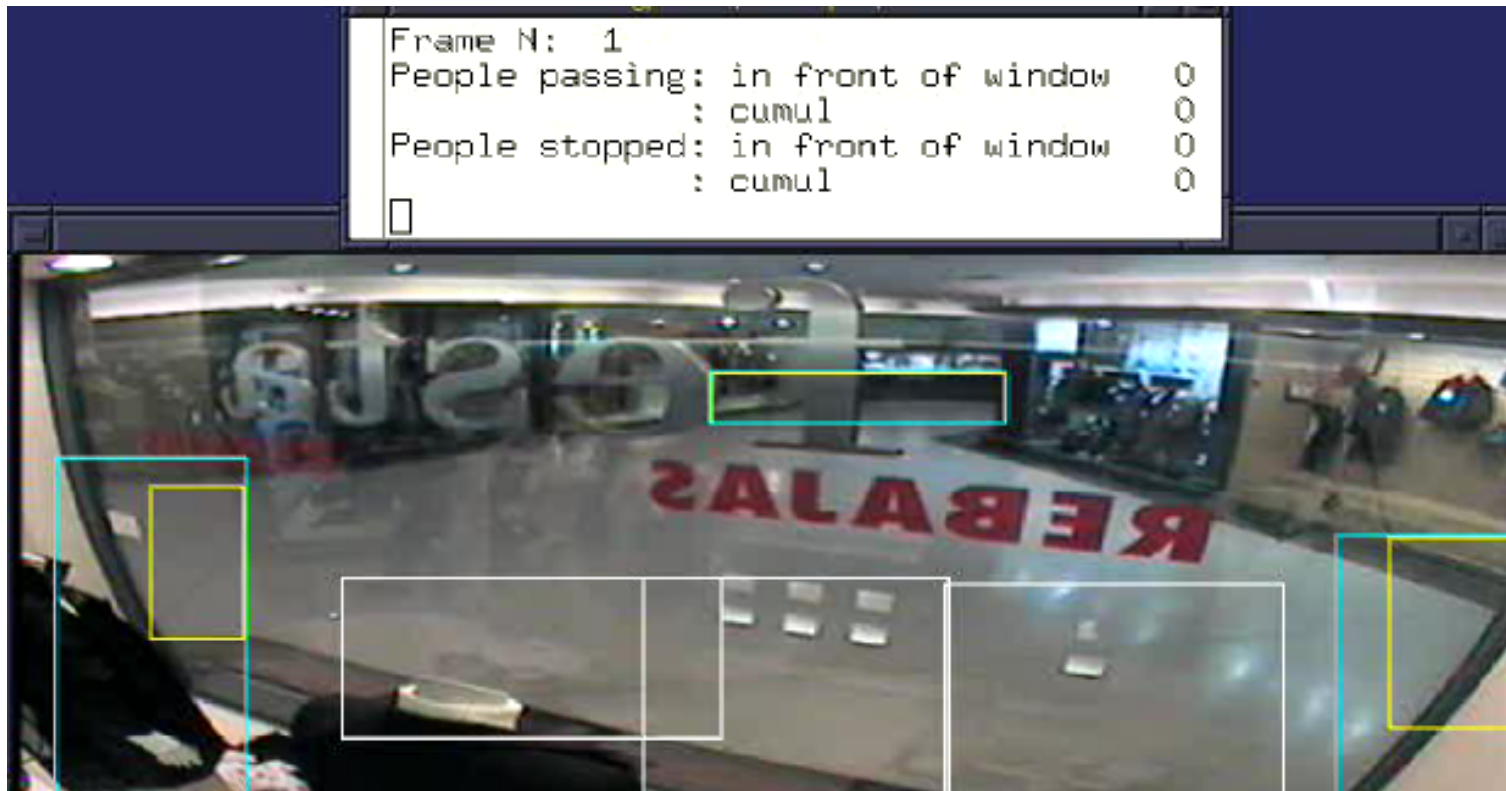
CAVIAR
Outdoor Test Bed
INRIA
Back Parking Lot

2 Outdoor Surveilance Platforms,
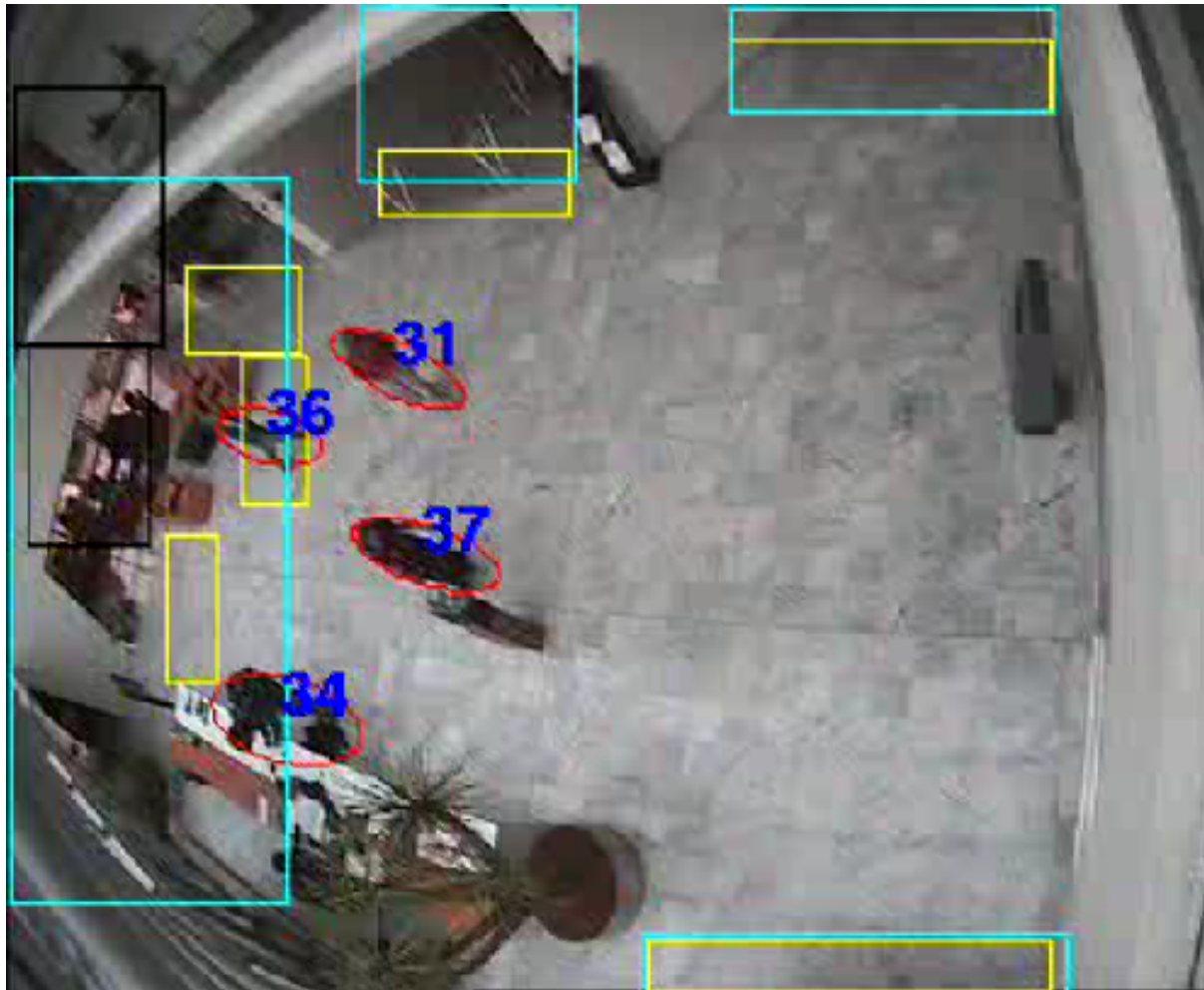3 m separation, 3 meter height
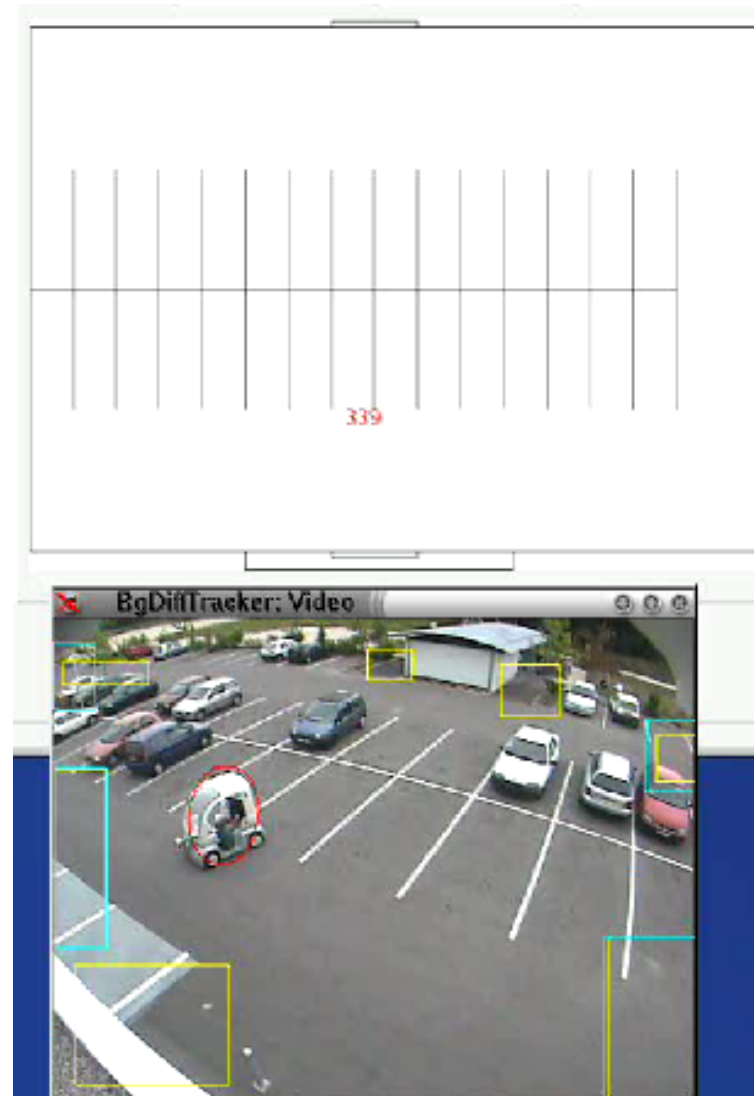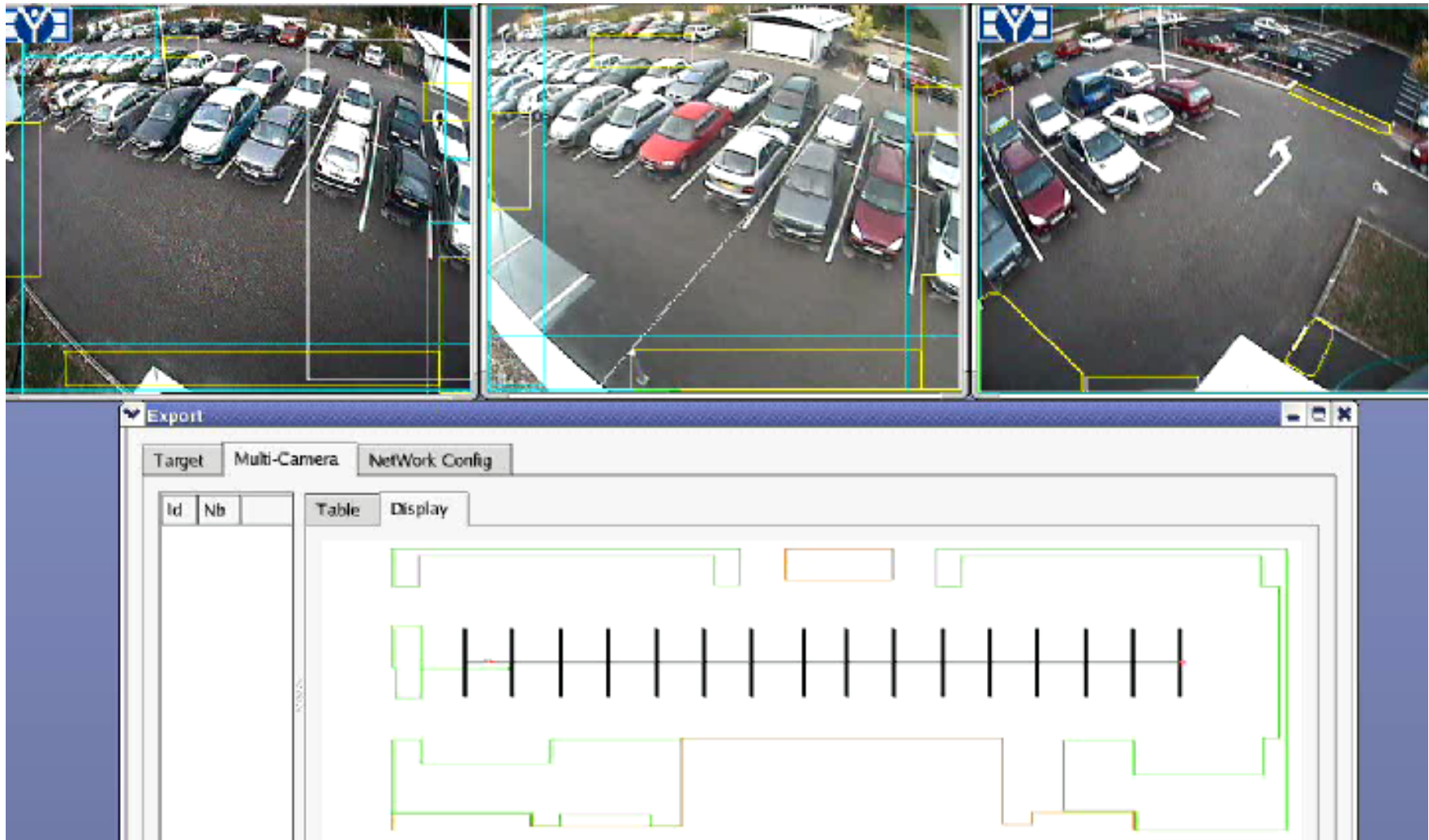
CAVIAR Indoor Test-bed: INRIA Entrance Hall

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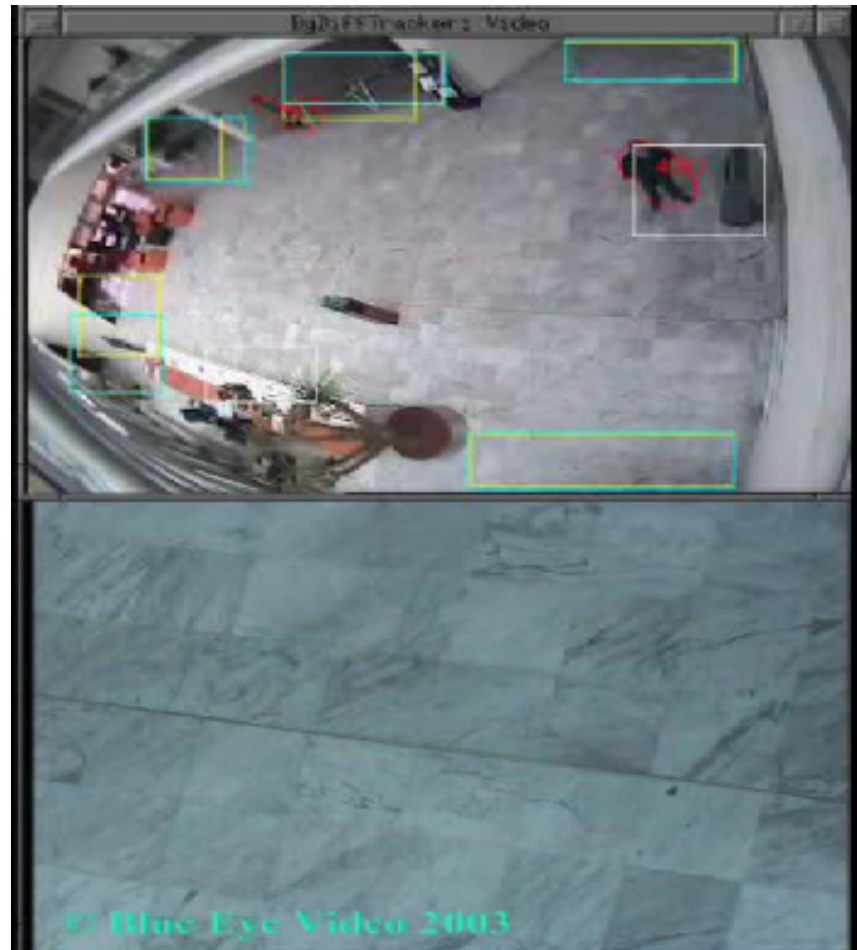
Tracking Multiple Targets

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

<u>Autonomic computing</u> :

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)

<u>Autonomic computing systems</u> 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.

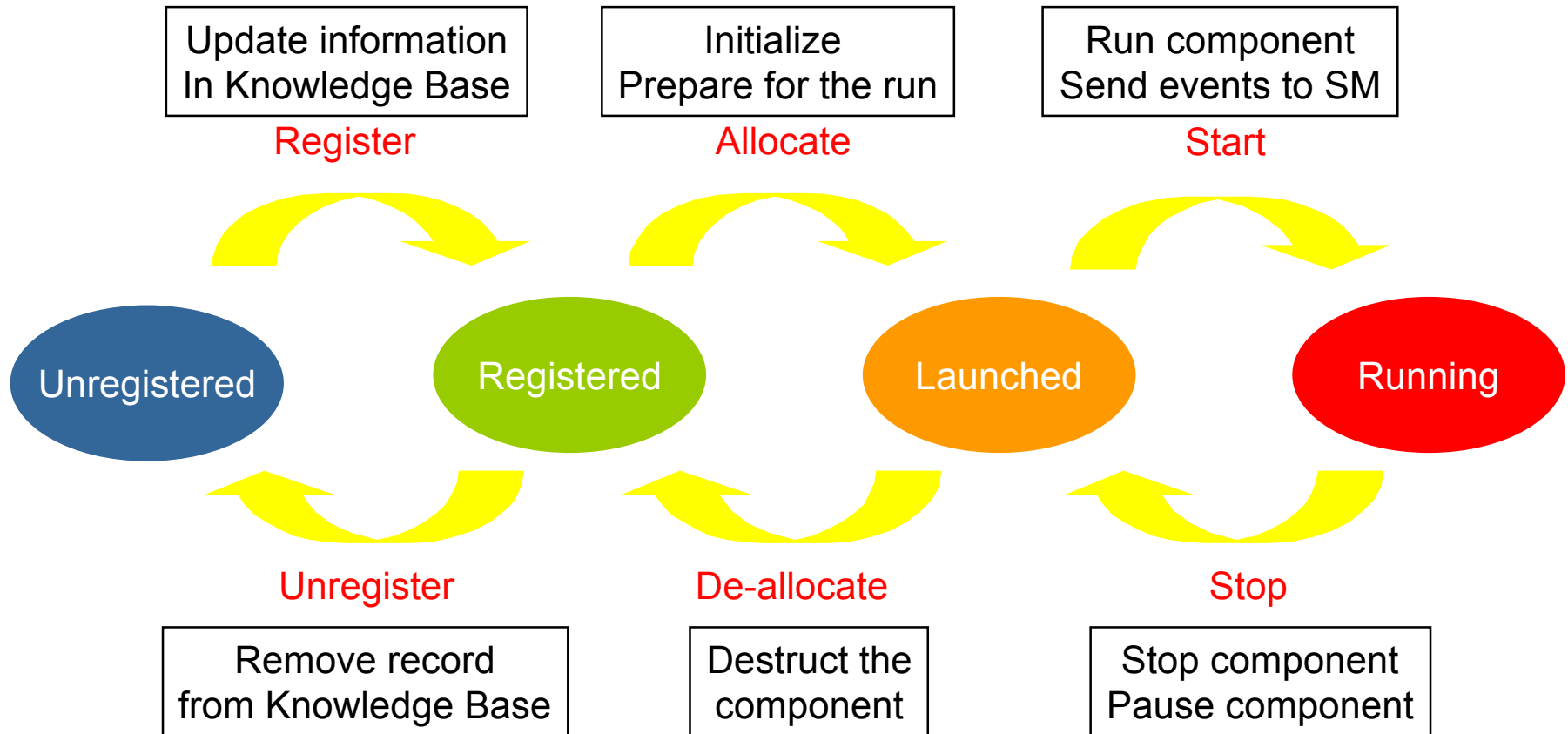# Autonomic Properties for Perceptual Components

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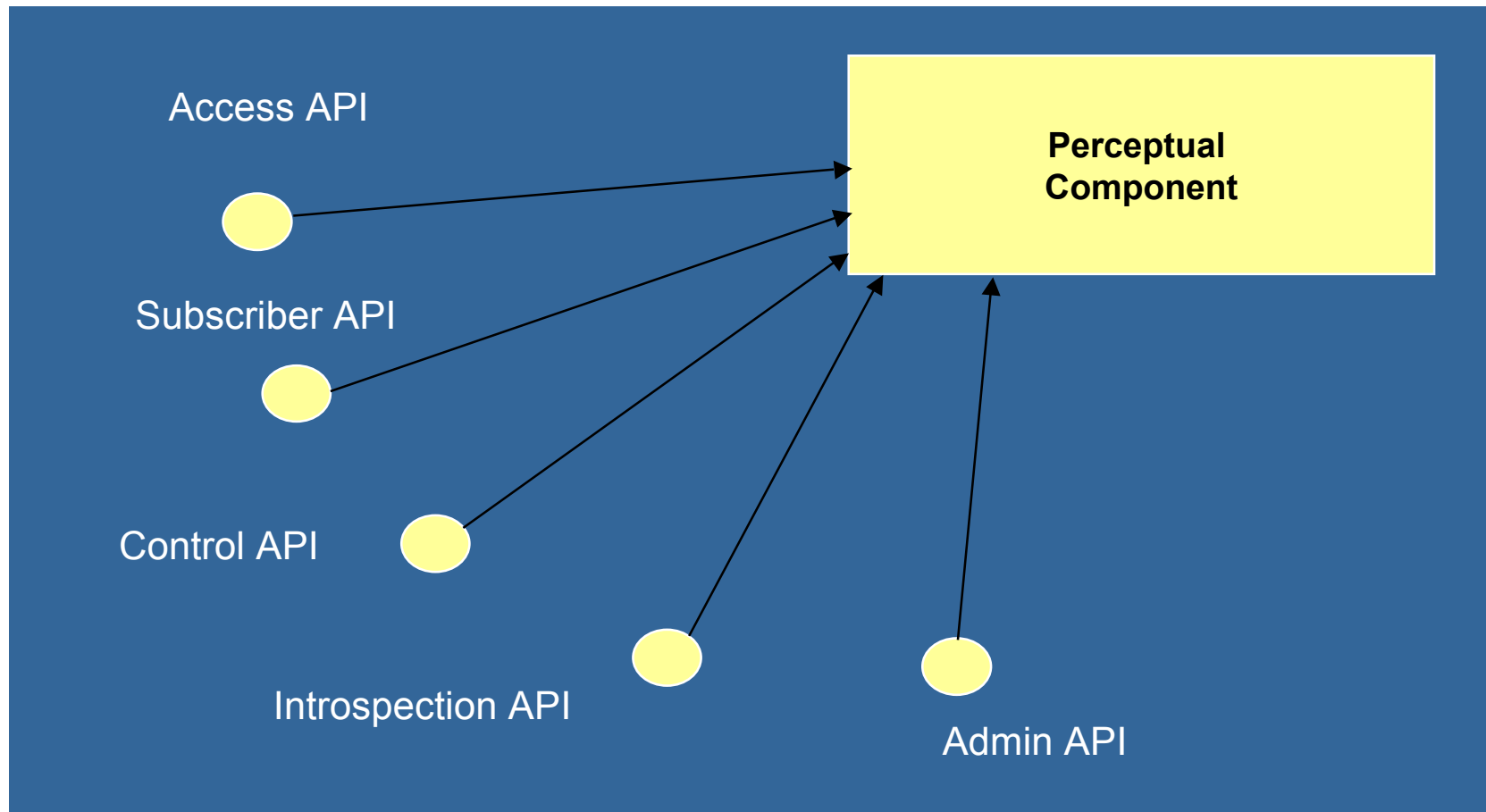
Self-repair: The process controller can detect and correct conditions by reconfiguring modules.

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

Perceptual Component LifeCycle

Update information In Knowledge Base	Initialize Prepare for the run	Run component Send events to SM

Register   Allocate   Start

Unregistered   **Registered**   **Launched**   **Running**

Unregister   De-allocate   Stop

Remove record from Knowledge Base	Destruct the component	Stop component Pause component

Perceptual Components API

Access API

Subscriber API

Control API

Perceptual Component

Introspection API

Admin 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
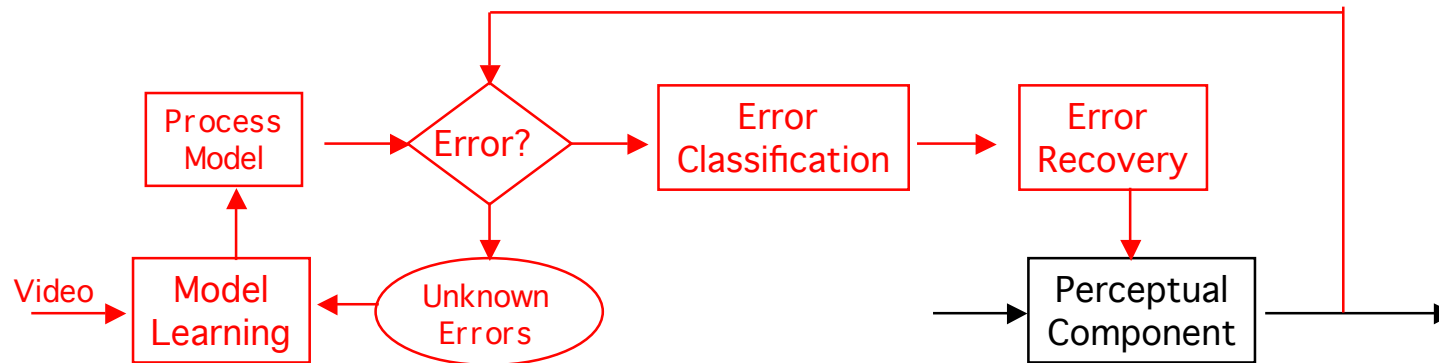
Autonomic Properties for Perceptual Components

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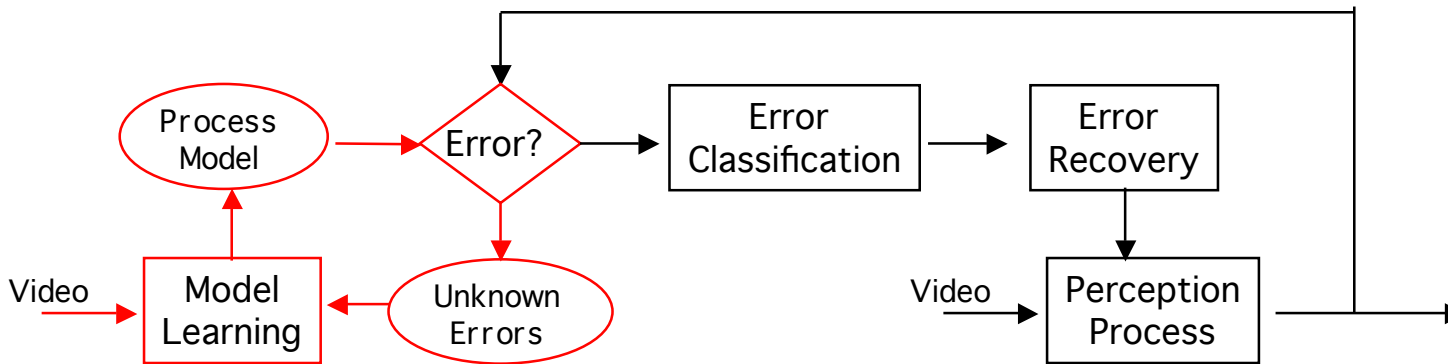
Self-regulation: The component supervisor adapts parameters to maintain a desired process state.

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).

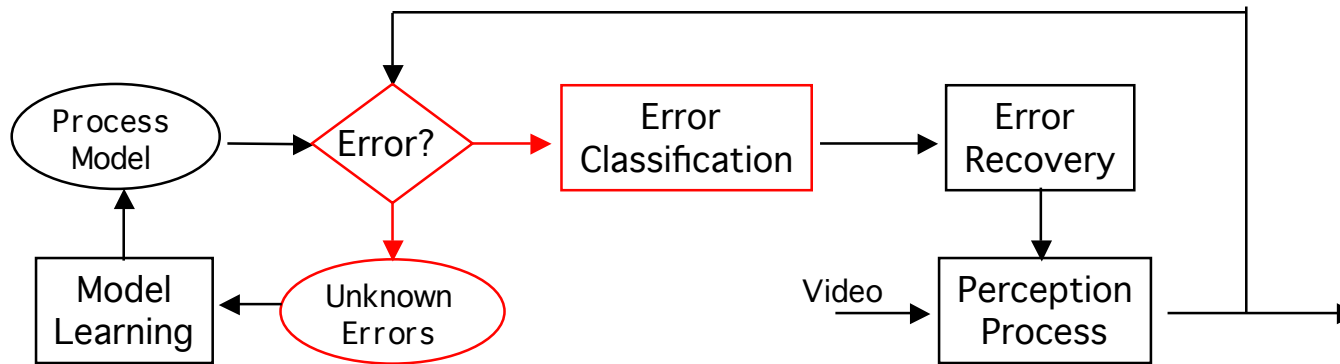# Autonomic Properties for Perceptual Components

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
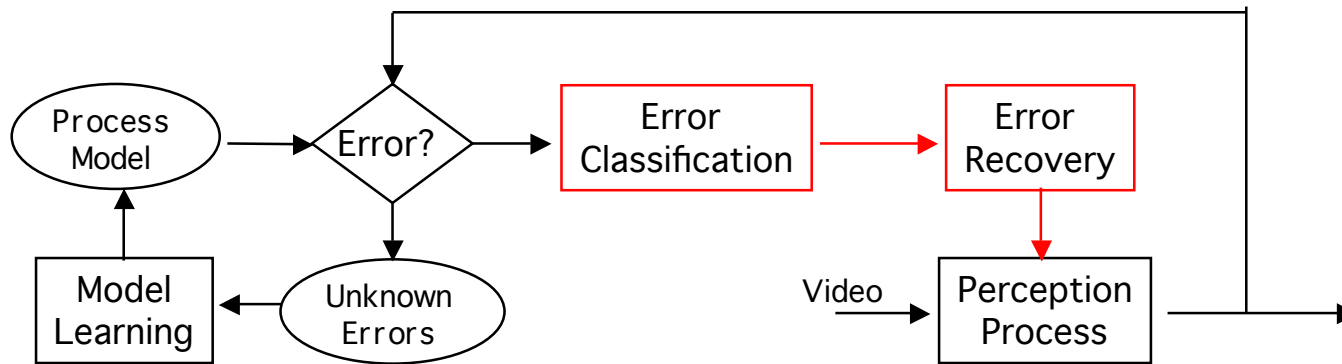

Two Cases:

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
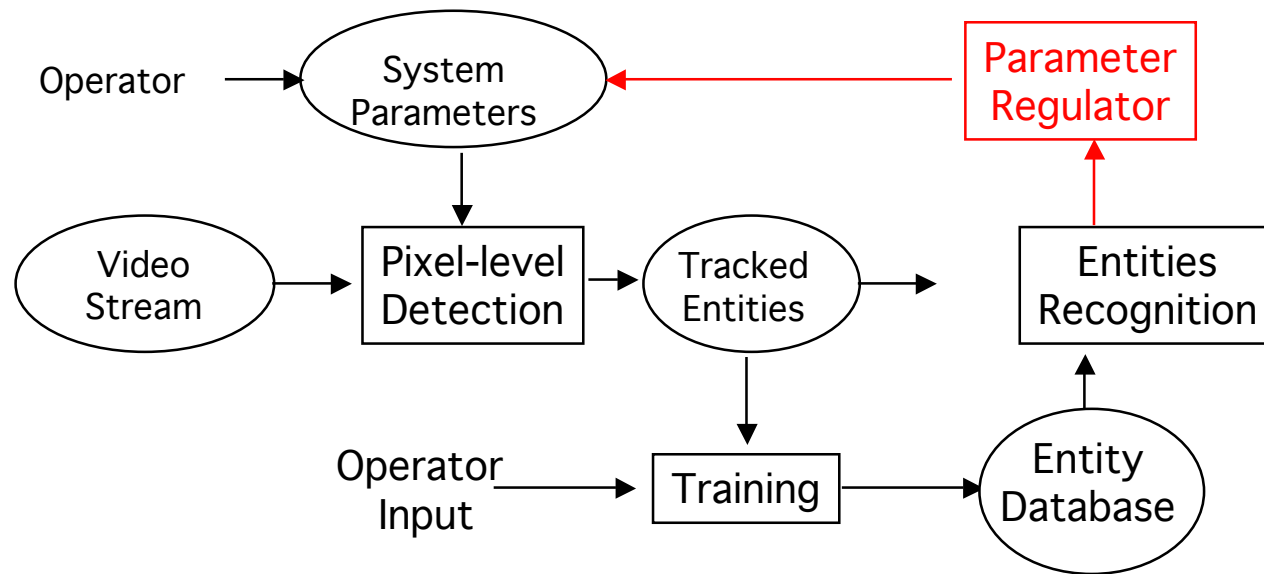
Parameter regulation provides robust adaptation to
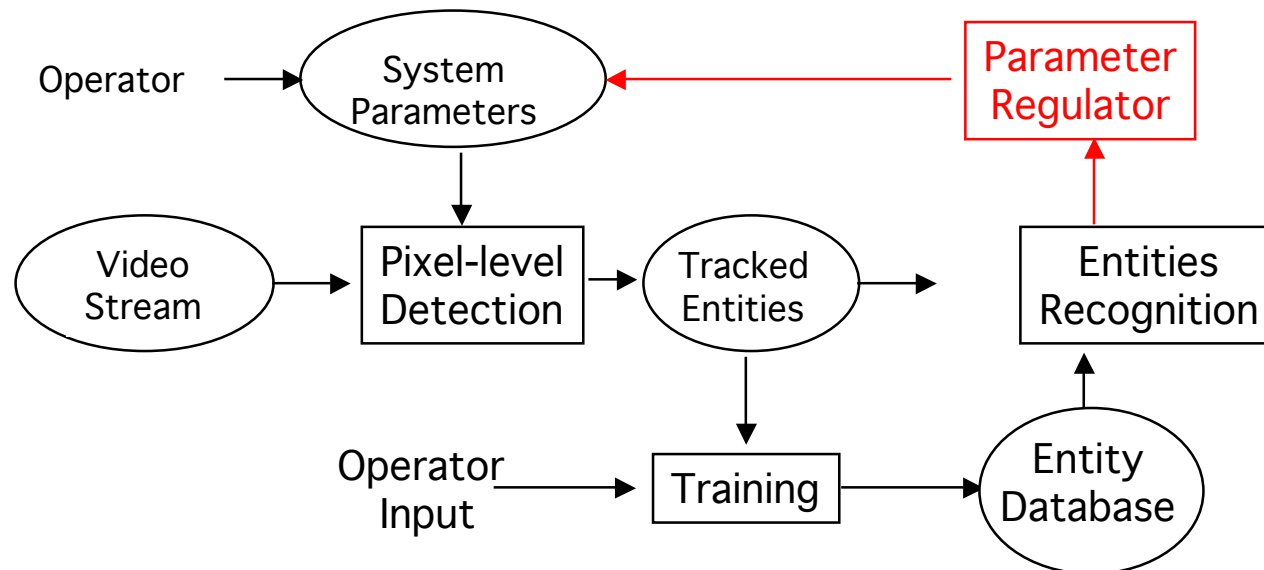Changes in operating conditions.

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)

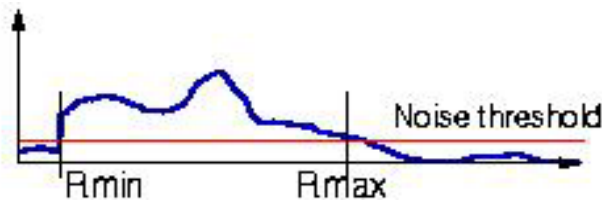# Autonomic Processes Regulation

For parameter regulation, we need

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

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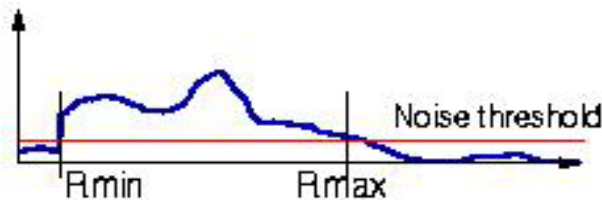
Target Detection

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
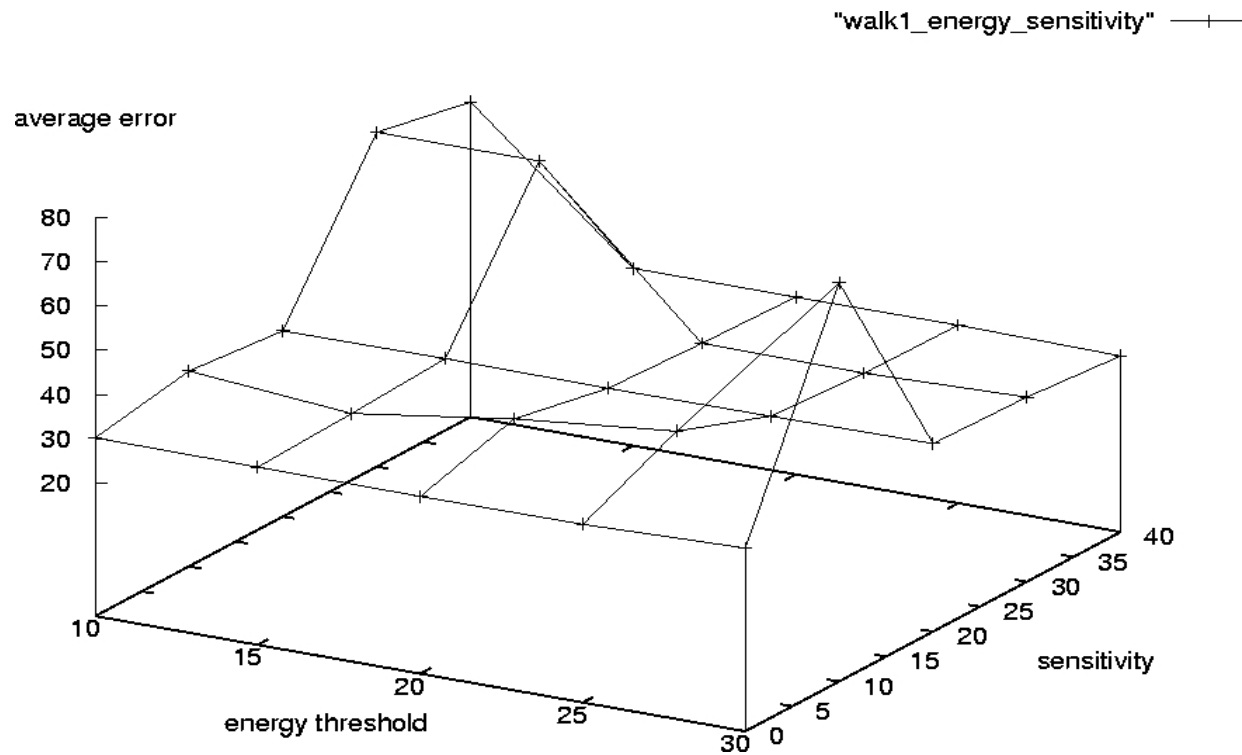
Solution: Use ground truth data as reference
(Ground truth can be obtained by Semi-supervised learning)
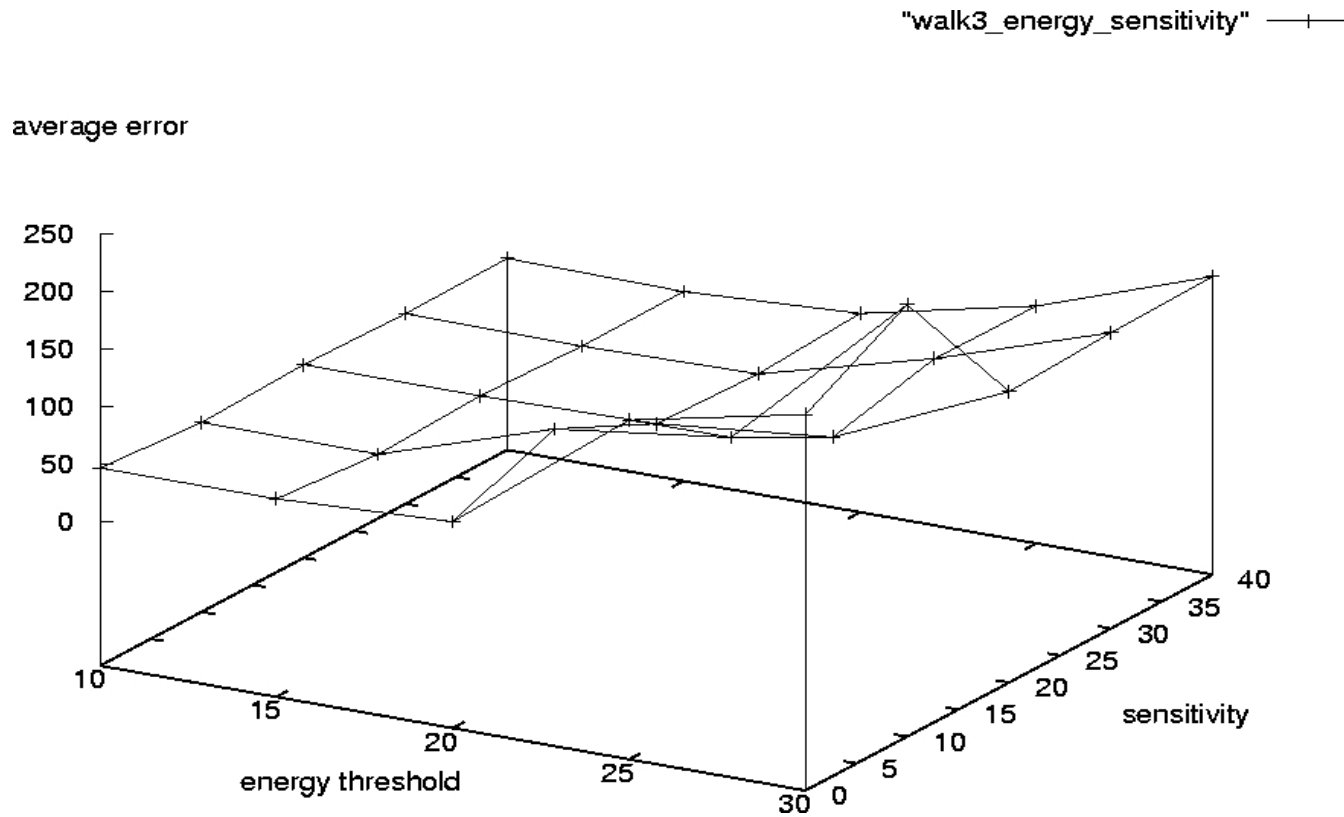
Target Detection: PETS 04 Data

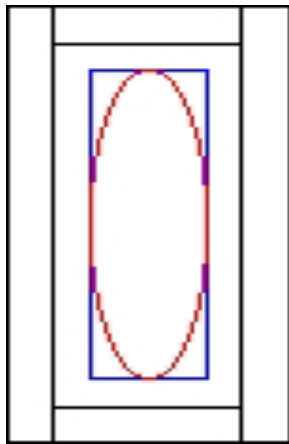# Detection error as a function of threshold and sensitivity

Sequence: Walk1

Detection error as a function of threshold and sensitivity

Sequence: Walk3

"walk3_energy_sensitivity"

average error

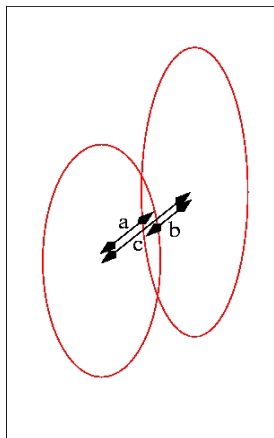energy threshold

sensitivity

Split and Merge

Split:

Targets surrounded by a detection "halo"
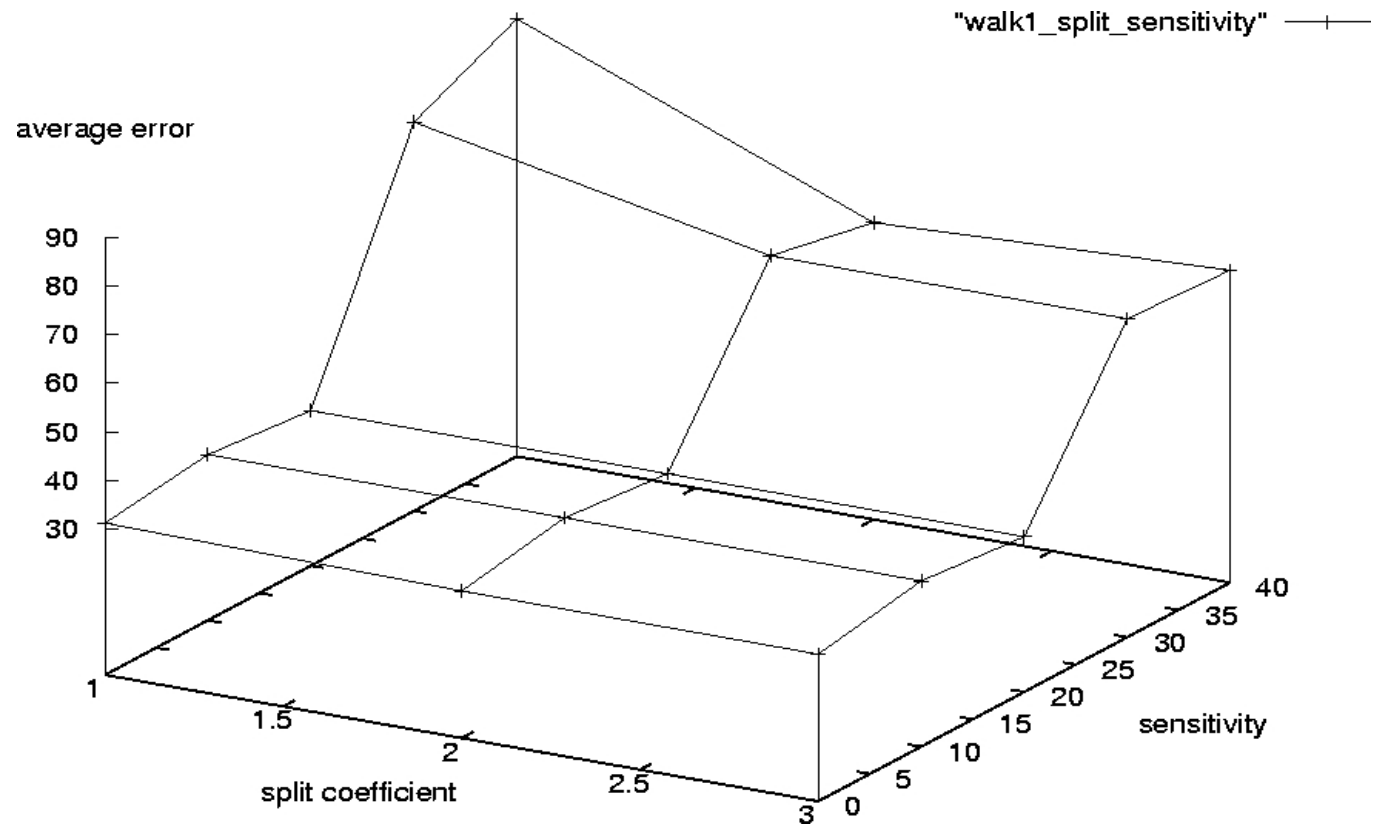
Parameters: Size of Halo, Size of "dead-zone"

Merge:

Target overlap

Parameter: Mahalanobis distance

Split and Merge

Lessons from Autonomic Vision

Statistical Learning is a powerful tool for Autonomic computing.

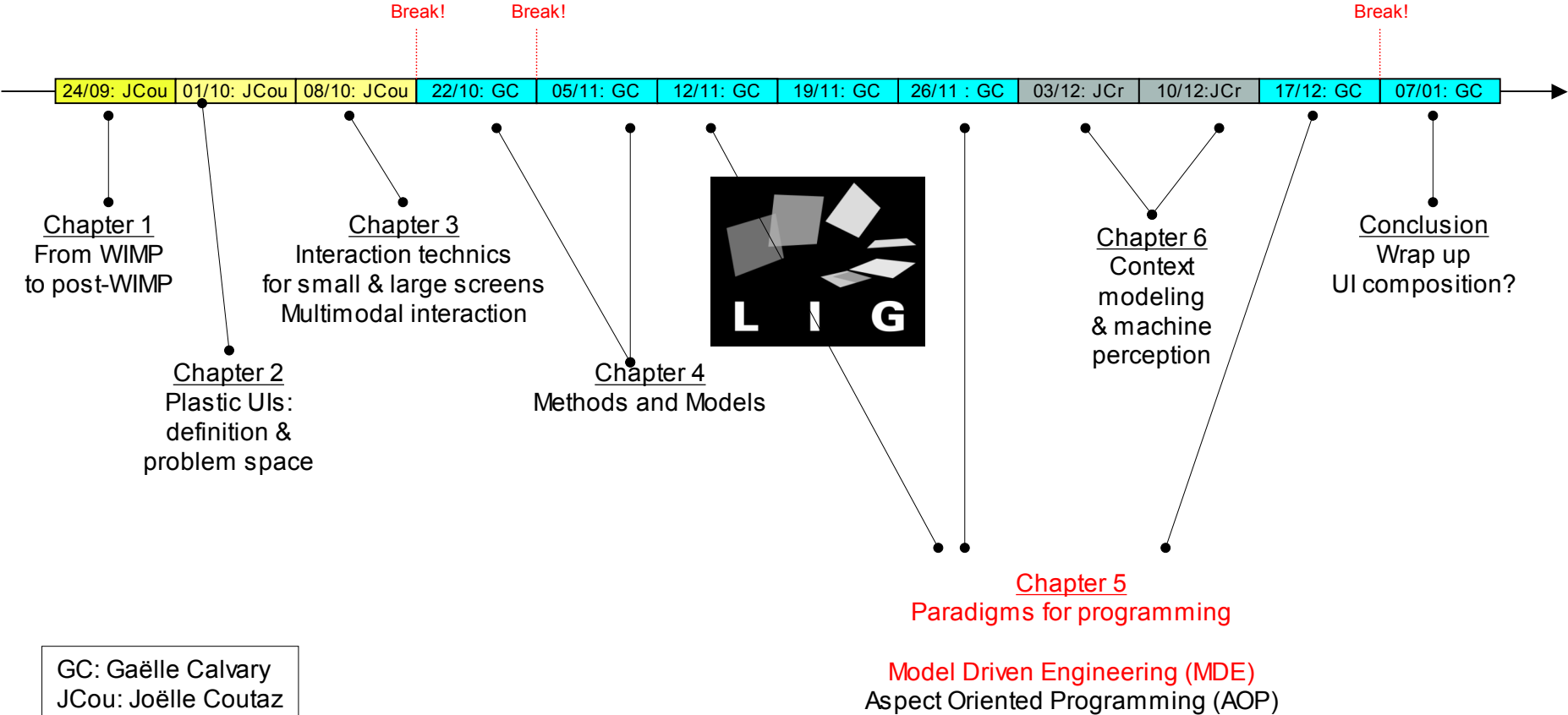Machine perception is an ideal domain for experimenting with Autonomic Computing.

- 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

Break! Break! Break!

| 24/09: JCou | 01/10: JCou | 08/10: JCou | 22/10: GC | 05/11: GC | 12/11: GC | 19/11: GC | 26/11 : GC | 03/12: JCr | 10/12:JCr | 17/12: GC | 07/01: GC |

Chapter 1
From WIMP
to post-WIMP

Chapter 3
Interaction technics
for small & large screens
Multimodal interaction

Chapter 6
Context
modeling
& machine
perception

Conclusion
Wrap up
UI composition?

Chapter 2
Plastic UIs:
definition &
problem space

Chapter 4
Methods and Models

L I G

Chapter 5
Paradigms for programming

Model Driven Engineering (MDE)
Aspect Oriented Programming (AOP)

GC: Gaëlle Calvary
JCou: Joëlle Coutaz
JCr: 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