Winter School ROBOTICS PRINCIPIA
Nice, January 2019

Decision Making in (multi) Robot Systems

Olivier Simonin

INSA Lyon – CITI Lab. - Inria Chroma team

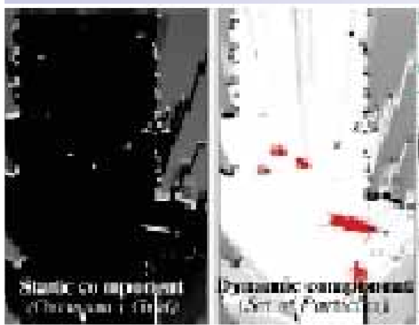
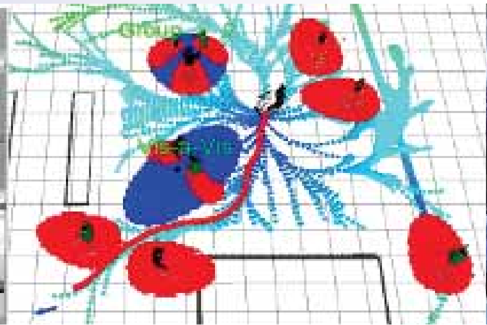
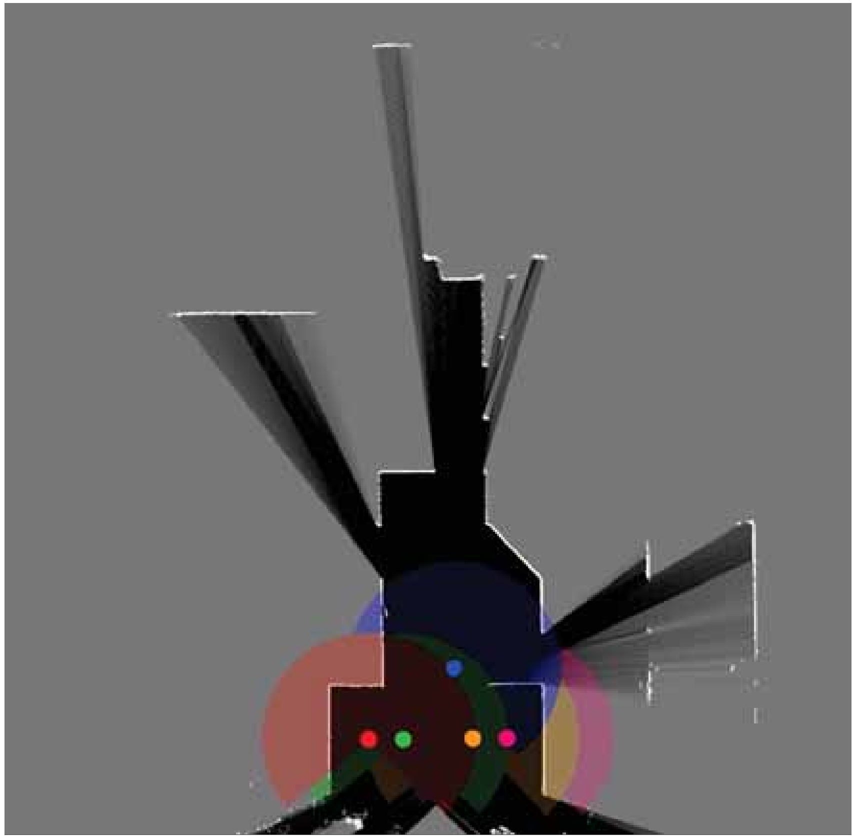




Illustration ..



ANR Carotte Challenge - Final 2012
Cartomatic team

Inria INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Outline

I. Decision making ?
II. Classical architectures
III. Learning based architectures

IV. Decision in multi-robot systems
V. Strategies for multi-robot exploration

Outline

I. Decision making ?

II. Classical architectures

III. Learning based architectures

IV. Decision in multi-robot systems

V. Strategies for multi-robot exploration

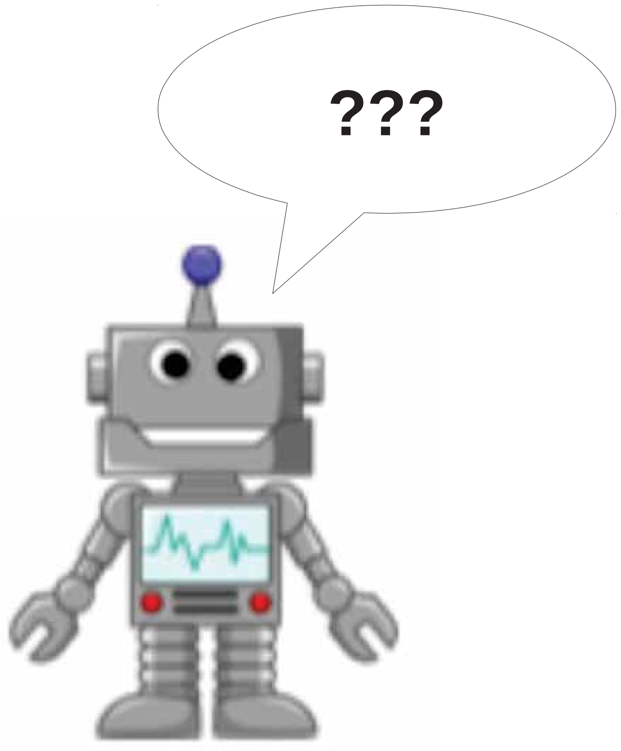
Decide what ?

Actions to fulfill my task

To avoid collisions / risks / breakdown

To cooperate with other robots

...



Decide what ?

Actions to fulfill my task

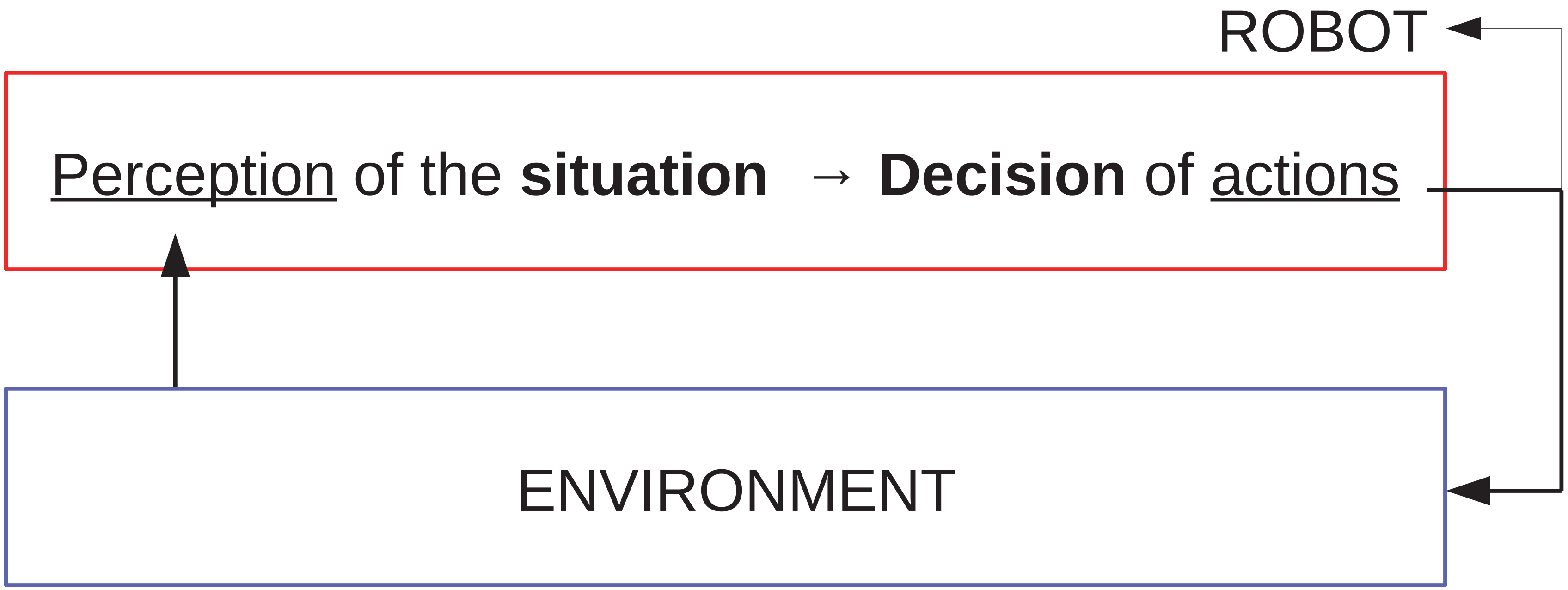
To avoid collisions / risks / breakdown

To cooperate with other robots

...

Perception of the **situation** → **Decision** of actions

Loop Perception-Decision-Action



Loop Perception-Decision-Action

ROBOT

Perception of the **situation** → **Decision** of actions

A.I.

Control

Loop Perception-Decision-Action

ROBOT

Perception of the **situation** → **Decision** of actions

Modeling the environment

Situation awareness

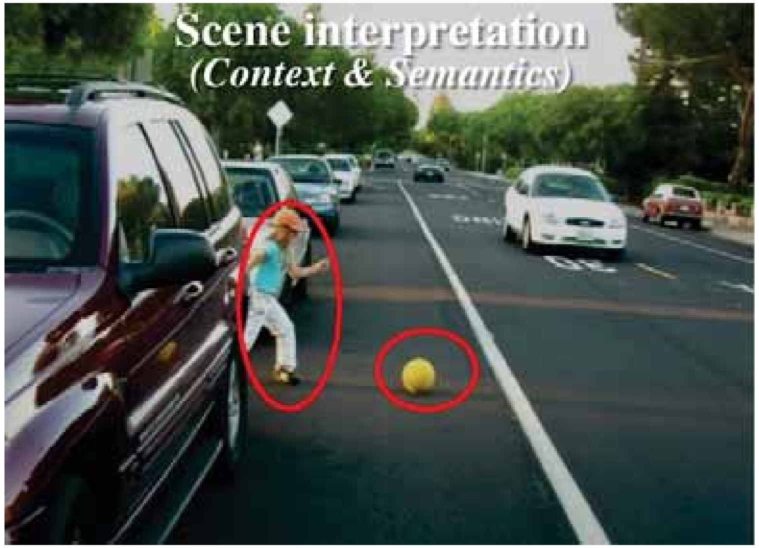
Predict

Ex.1 Situation awareness with probabilistic grid

Occupancy grid + veloc. + Bayesian F. [Laugier et al. 2012-2018]

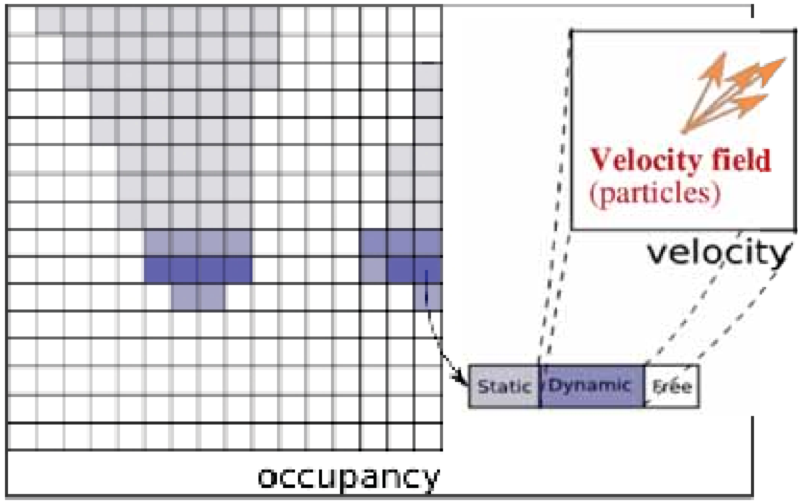
Ex.1 Situation awareness with probabilistic grid

Bayesian Occupancy Filtering (BOF[1], CMCDOT[2])
- **Occupancy grid**
- **Velocity** distribution / cell
- **Object** identification (cell clustering)
- **Prediction** of motion (bayes. filtering)



Occupancy grid :
- Probability of occupancy in each cell, $P_{occ}(x,y)$
- 1 : occupied cell (black)
- 0.5 : unknown (gray)
- 0 : free cell (white)

e.g. Frequency approach (counting) :
$P_{occ}(x,y) = occ(x,y) / (empty(x,y) + occ(x,y))$



[1] C. Laugier et al., IJRR 2005, [2] Rummelhardt et al. ITS 2015

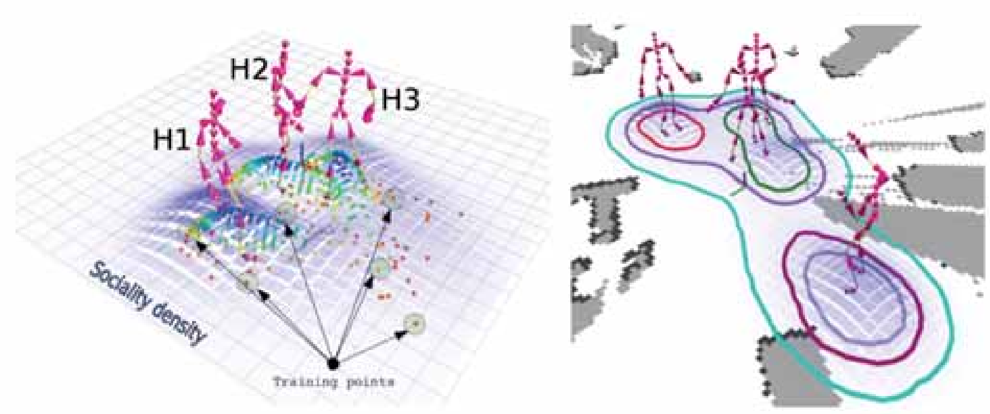
Ex.2 Social navigation : Proxemics approach

Identify humans and **predict** their motion

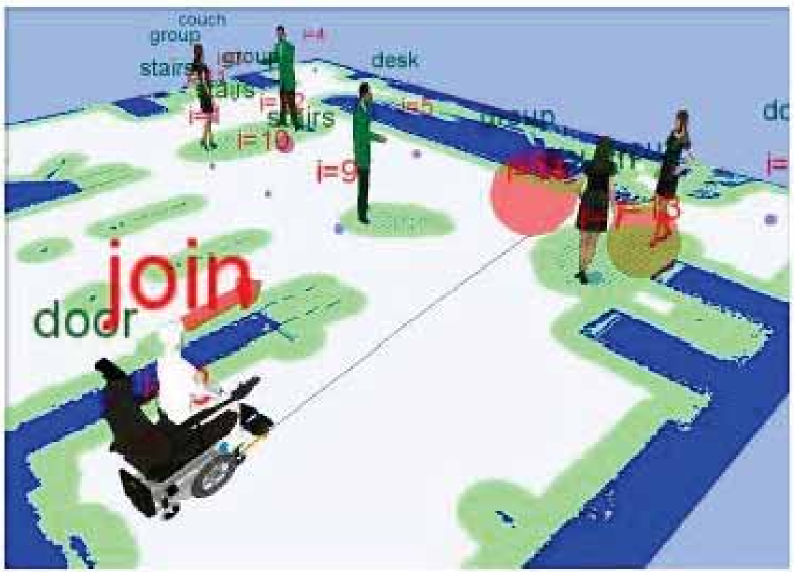
Comply with **social rules**

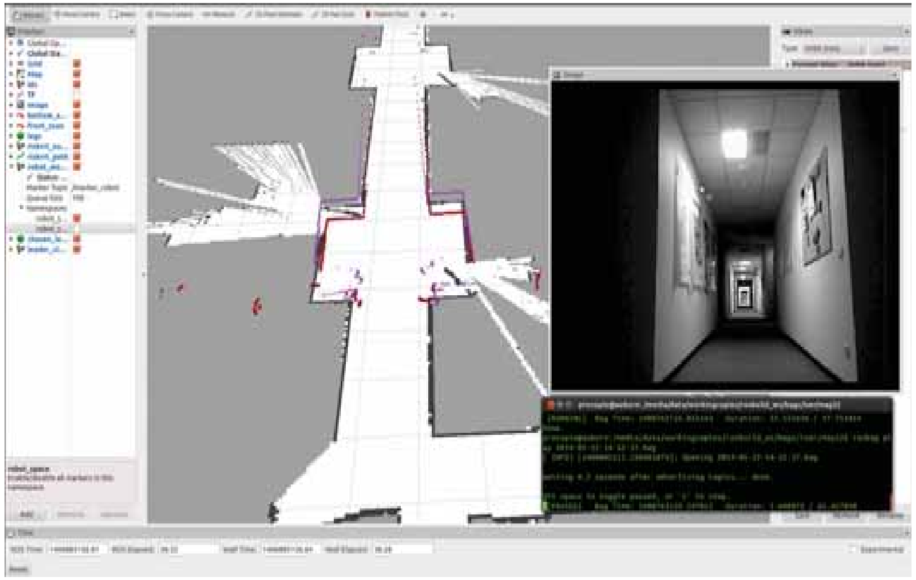
Map human activities

Compute **secure** paths and decisions



Gaussian model of personal area + geometric formation





[Spalanzani et al. 2015-18]

Loop Perception-Decision-Action

ROBOT

Perception of the **situation** → **Decision** of actions

Reacting

Planning / Learning

Reacting vs. Planning vs. Learning

Reacting : compute **one** action (real time)

Planning : compute a **sequence** of actions (eg. motion planning)

Learning : compute a **policy** (state→action)

+

Cooperate/compete : add other agents in the decision

Reacting vs. Planning vs. Learning

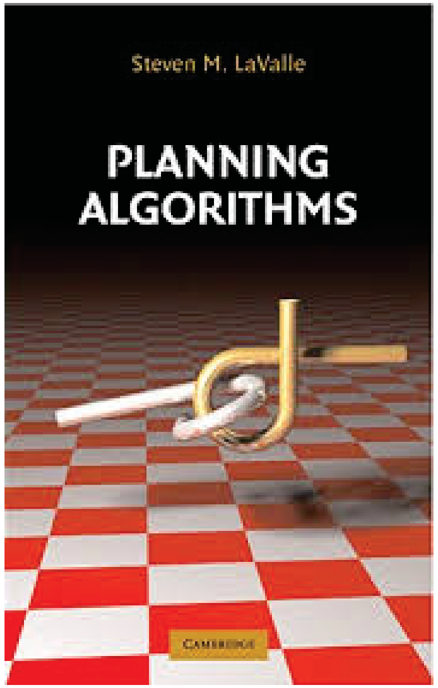
Reacting : Bio-inspired behaviors, Connexionism

Planning : STRIPS, PDDL, SAT, CSP, PRM ..

Learning : Reinforcement L., (PO)MDP, RNN, DeepLearning

+

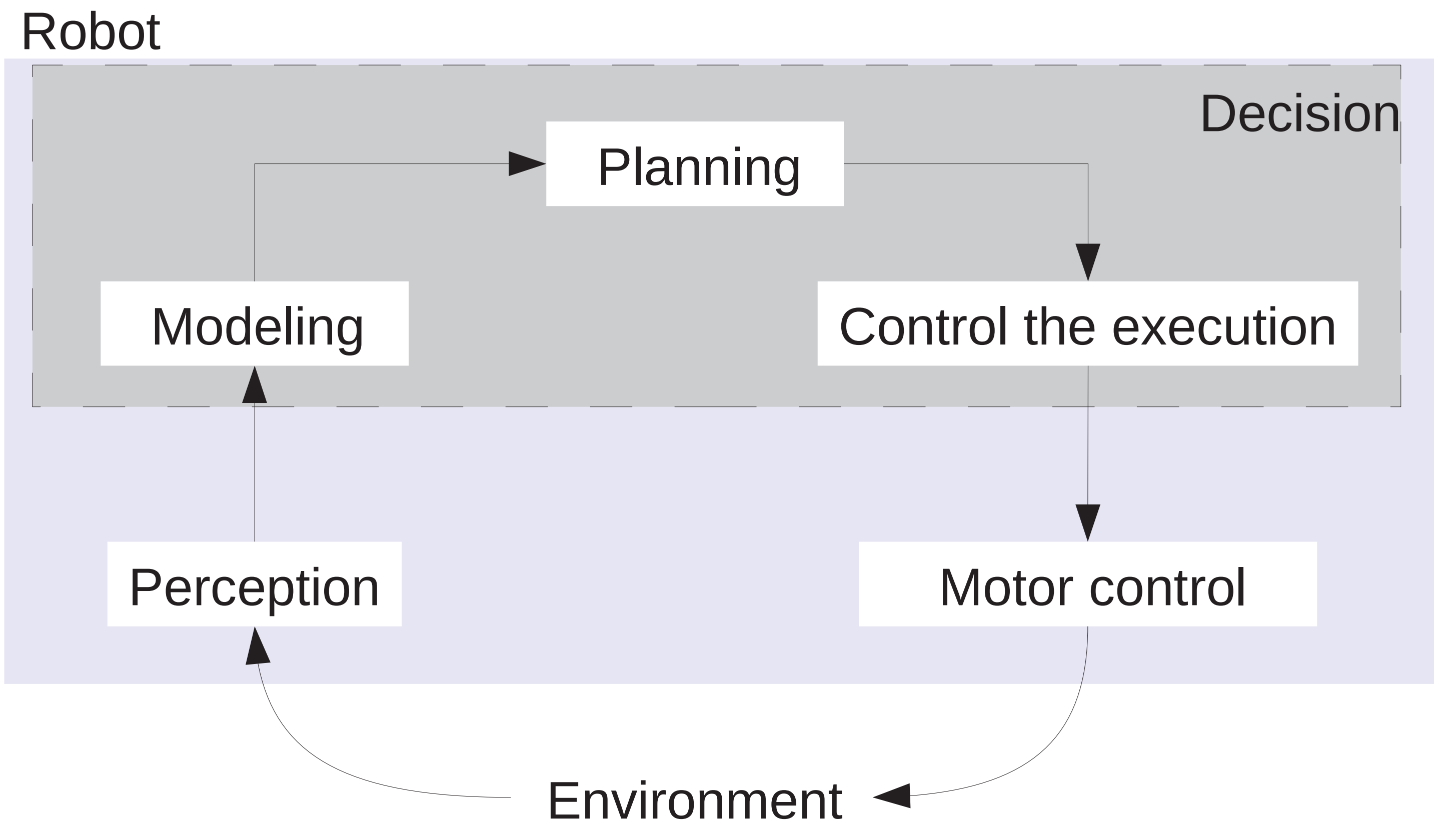
Cooperate/compete : add other agents in the decision



Outline

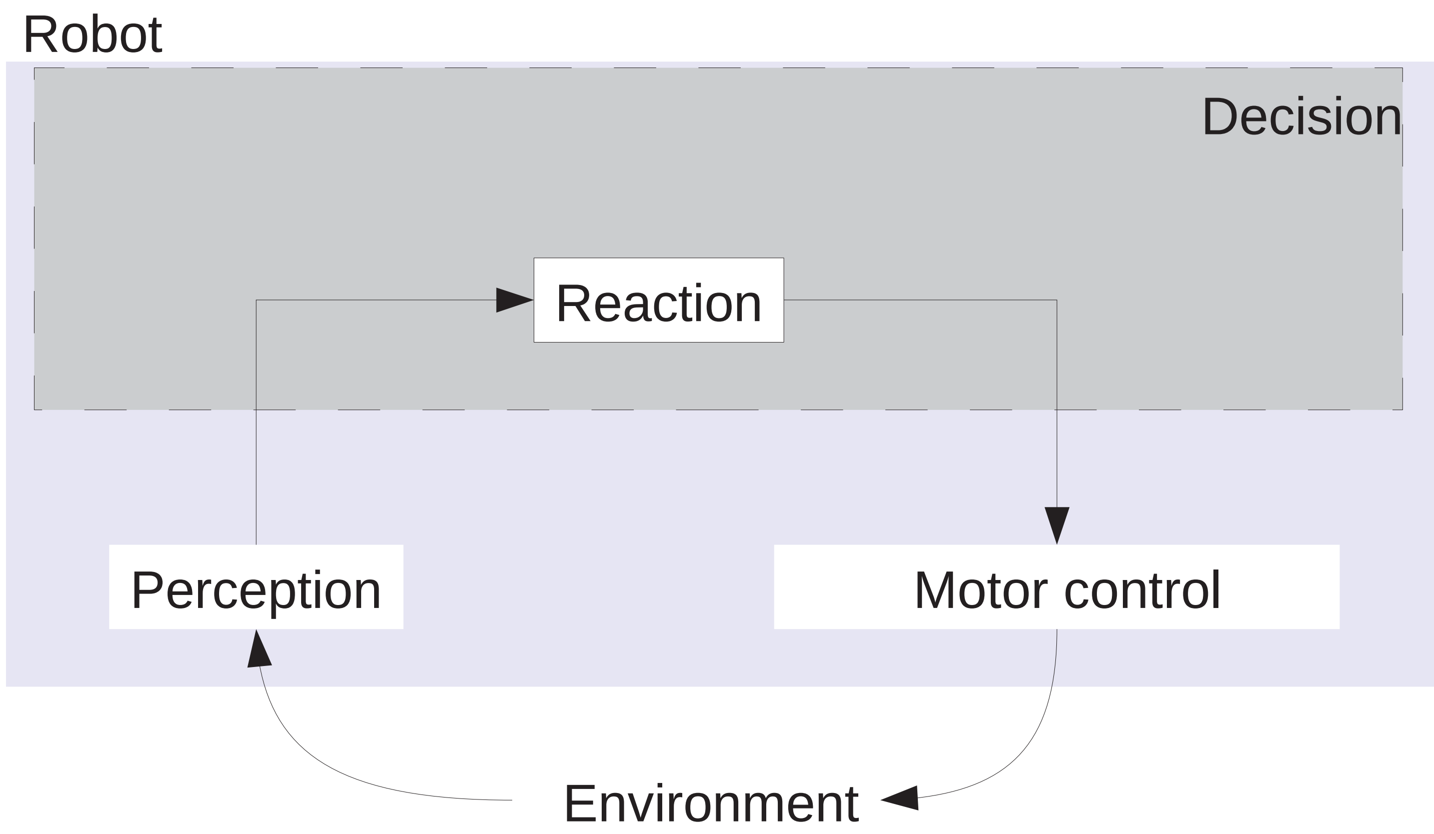
Sense-Plan-Act

[Nilsson80]



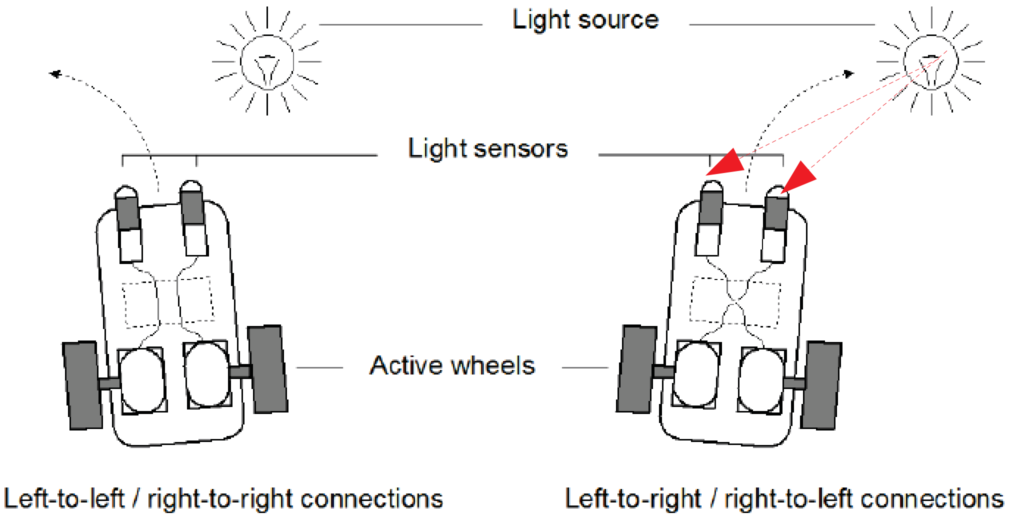
Slow planning → bottleneck !

Reactive architecture

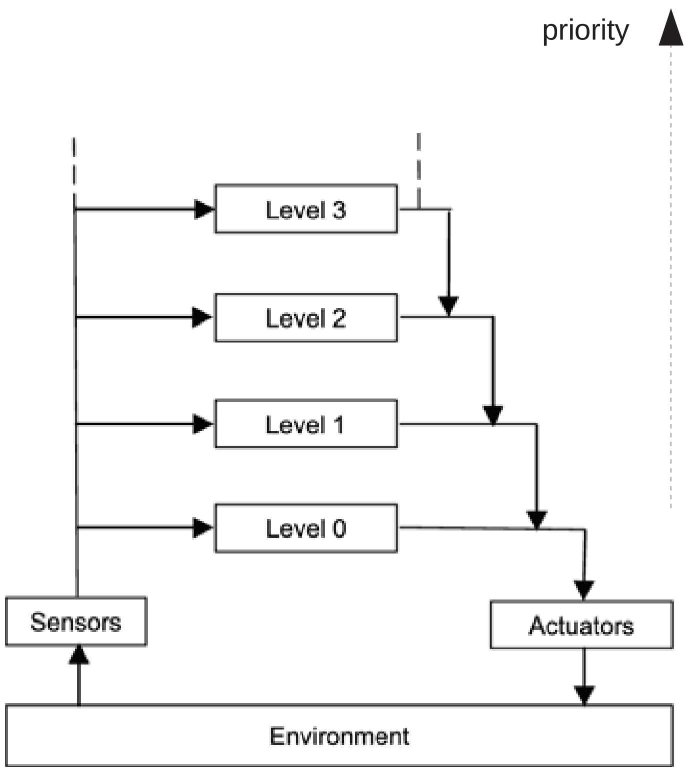


[Brooks 86] [Arkin 98]. Connecting sensors-motors, but **deadlocks** are possible !

Reactive architecture : Braitenberg, Brooks



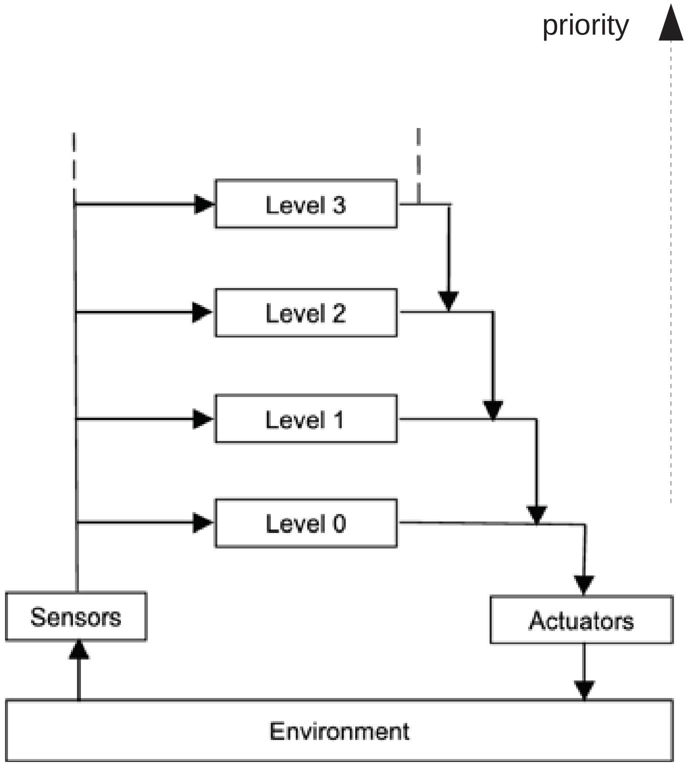
[Braitenberg 84] Vehicle (phototaxis)



[Brooks 86] Subsumption architecture

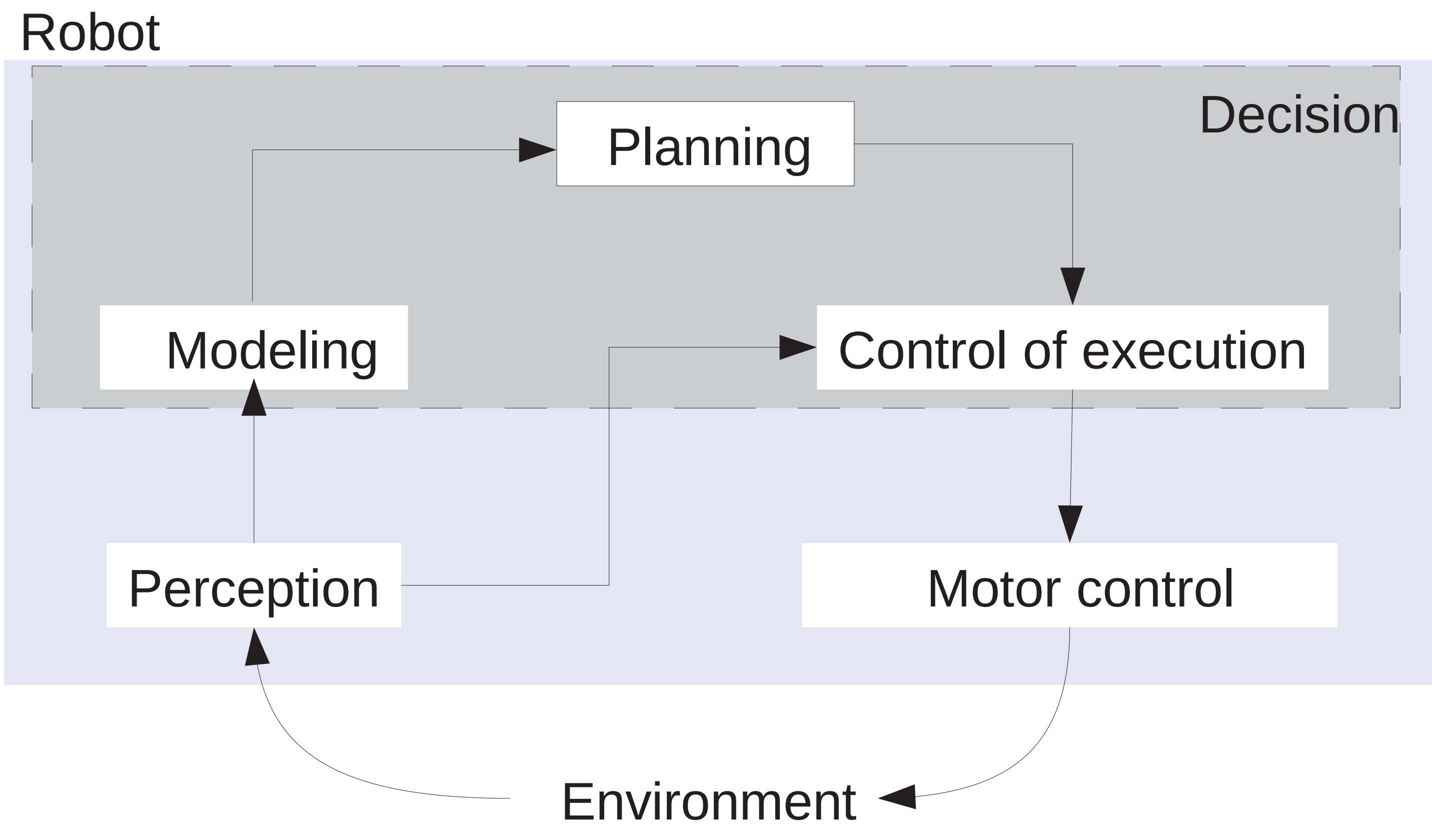
Reactive architecture : Braitenberg, Brooks

[Braitenberg 84] Vehicle (phototaxis)



[Brooks 86] Subsumption architecture

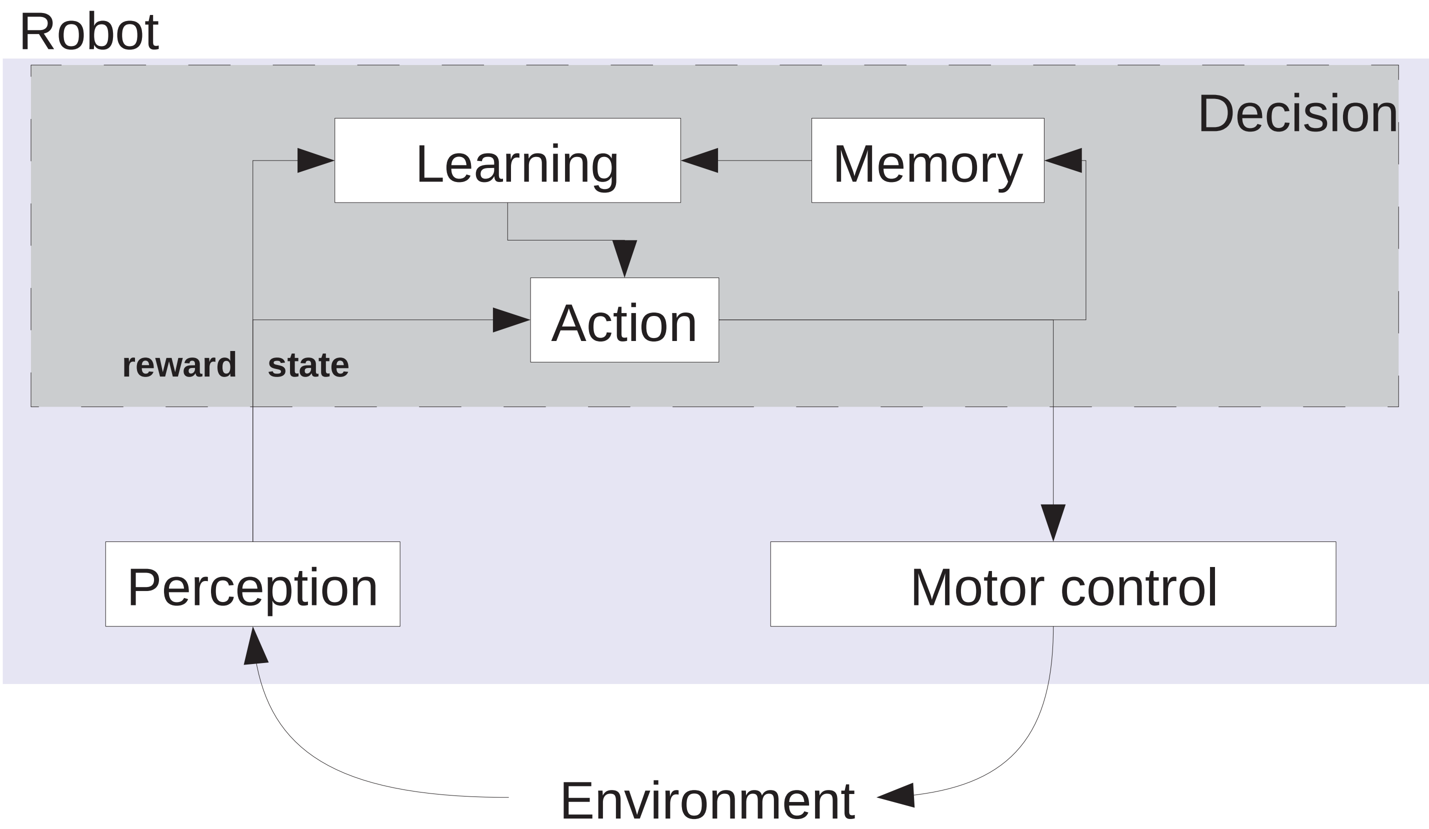
Multi-level architecture



Allow to combine fast reaction and planning

Outline

Learning architecture : non explicit knowledge repr.



Neuronal networks, Reinforcement Learning, Markov Decision Process (MDP) ..

Q-Learning [Watkins 89] (Reinforcement Learning)

Compute the quality of an action ***a*** in state ***s*** ($s \in S$, $a \in A$)

$$Q : S \times A \to \mathbb{R}$$

Each time step ***t*** the agent selects an action $\boldsymbol{a_t}$, observes a reward $\boldsymbol{r_t}$, enters in new state $\boldsymbol{s_{t+1}}$.
Then *Q(s,a) is updated :*

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

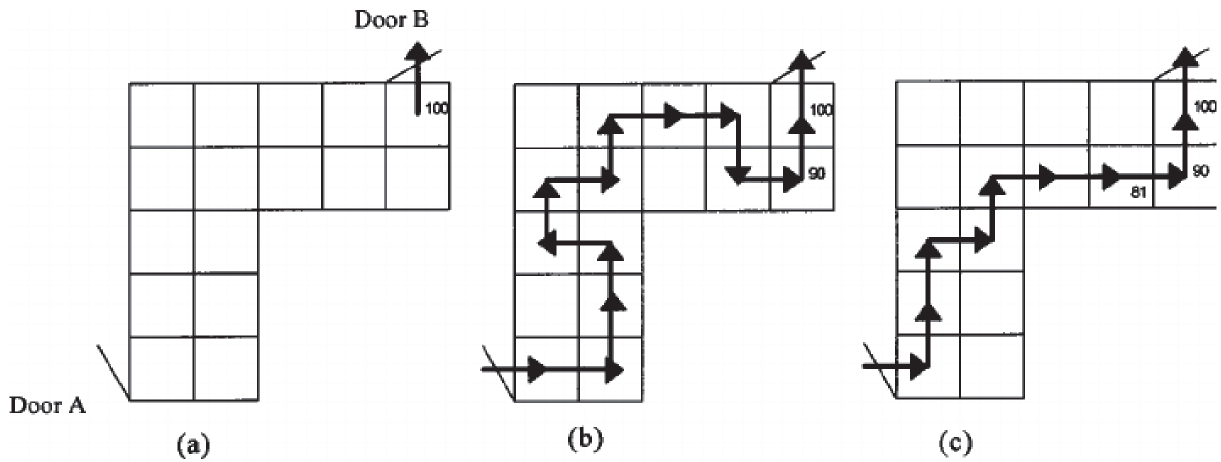
Converge to optimal policy that **maximizes the expected cumulative reward** $\sum_{t=0}^{\infty} \gamma^t r_t$
(proof [Watkins, Dayan 92])

Q-Learning [Watkins 89] (Reinforcement Learning)

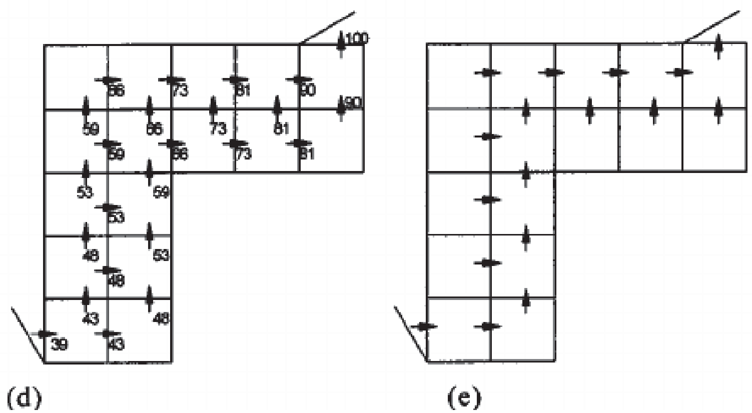
$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

The agent must explore/experiment the environment

→ exploration/exploitation dilemma



Grid example :
$\alpha = 1 \quad \gamma = 0.9$



Modeling uncertainty : Markov Decision Process

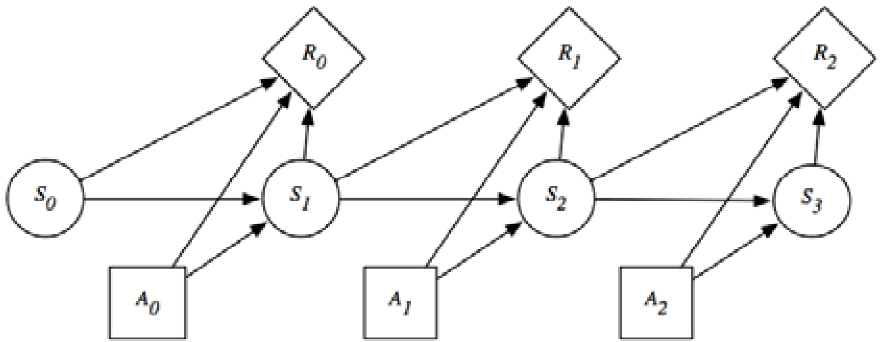
MDP : 4-tuple (S, A, P_a, R_a)

Actions' result is uncertain : that is true in robotics !

Exploit a model of the environment/system dynamics :

The probability that action ***a*** , in state ***s*** at time ***t*** will lead to state **s'** at time **t+1** is
$\boldsymbol{P_a}(s,s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$

Each action ***a*** , from state ***s*** to state **s'** gives an immediate reward $\boldsymbol{R_a}(s,s') \in \Re$



Modeling uncertainty : Markov Decision Process

MDP : 4-tuple (S, A, P_a, R_a)

Actions' result is uncertain : that is true in robotics !

Exploit a model of the environment/system dynamics :

The probability that action ***a***, in state ***s*** at time ***t*** will lead to state **s'** at time **t+1** is
$\boldsymbol{P_a}(s,s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$

Each action ***a***, from state ***s*** to state **s'** gives an immediate reward $\boldsymbol{R_a}(s,s') \in \Re$

Problem : finding a **policy π** that maximizes a cumulative function of the random rewards

$$\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})$$

$a_t = \pi(s_t)$

→ **Reinforcement Learning**

Modeling uncertainty : Markov Decision Process

MDP : 4-tuple (S, A, P_a, R_a)

Actions' result is uncertain : that is true in robotics !

Exploit a model of the environment/system dynamics :

The probability that action ***a***, in state ***s*** at time ***t*** will lead to state **s'** at time **t+1** is
$\boldsymbol{P_a}(s,s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$

Each action ***a***, from state ***s*** to state **s'** gives an immediate reward $\boldsymbol{R_a}(s,s') \in \Re$

Problem : finding a **policy π** that maximizes a cumulative function of the random rewards

Several solutions :
- Value iteration [Bellman 57]
- Policy iteration [Howard 60]
- ..

$$\pi(s) := \arg\max_a \left\{ \sum_{s'} P(s'|s,a) \left(R(s'|s,a) + \gamma V(s') \right) \right\}$$

$$V(s) := \sum_{s'} P_{\pi(s)}(s,s') \left(R_{\pi(s)}(s,s') + \gamma V(s') \right)$$

Computation can be very expensive..

Modeling uncertainty : Markov Decision Process

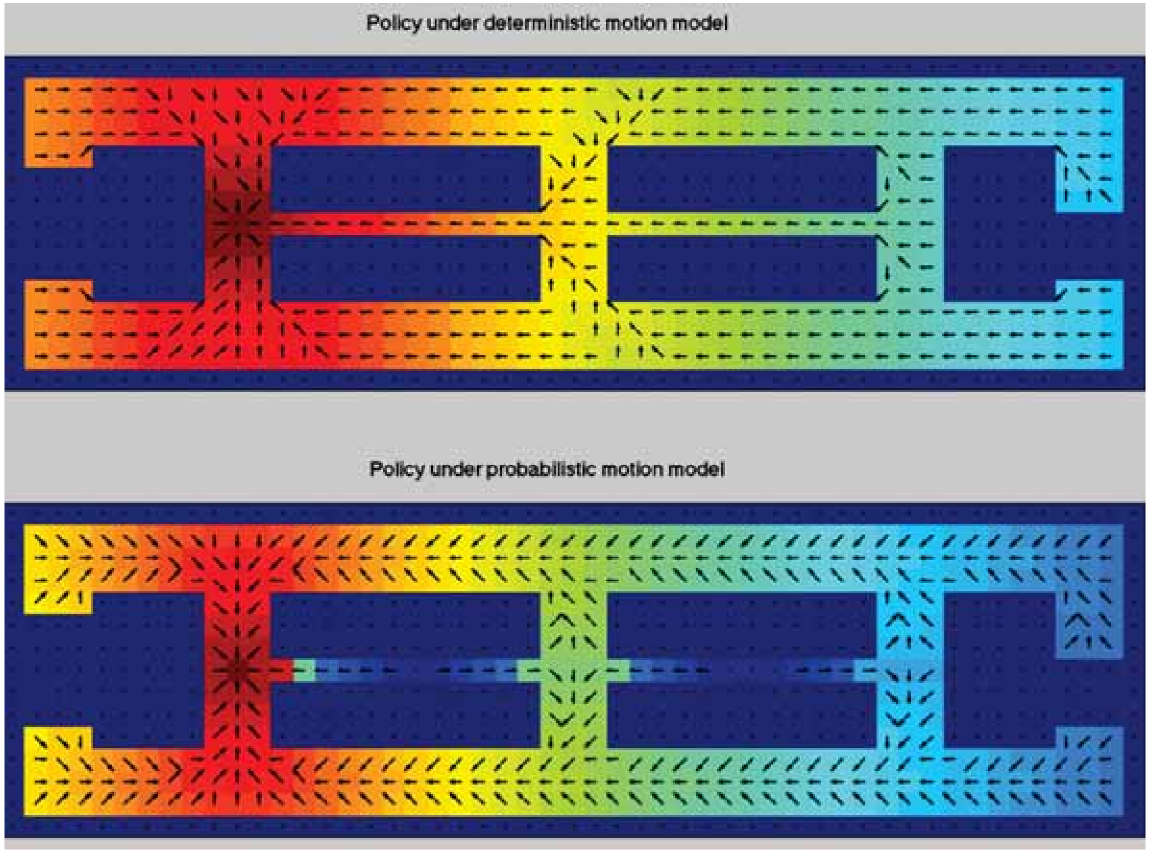
MDP : 4-tuple (S, A, P_a, R_a)

Actions' result is uncertain : that is true in robotics !

Exploit a model of the environment/system dynamics :

The probability that action ***a*** , in state ***s*** at time ***t*** will lead to state **s'** at time **t+1** is

$\boldsymbol{P_a}(s,s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$



Modeling uncertainty : Markov Decision Process

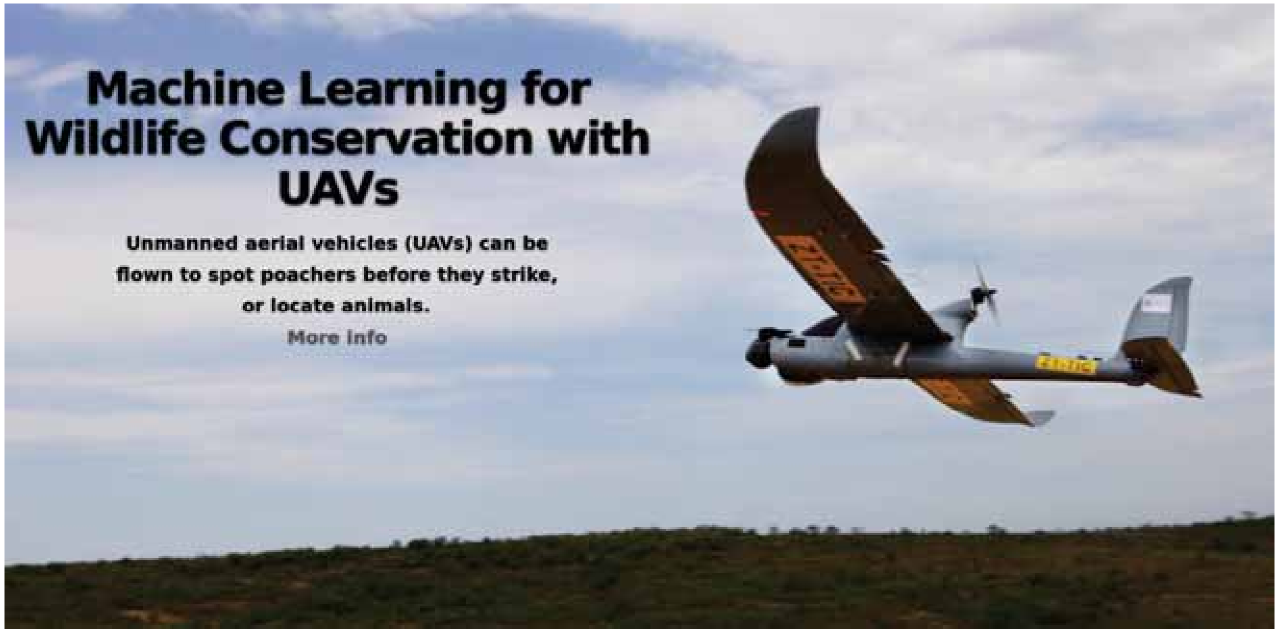
MDP : 4-tuple (S, A, P_a, R_a)

Actions' result is uncertain : that is true in robotics !

Exploit a model of the environment/system dynamics :

The probability that action ***a***, in state ***s*** at time ***t*** will lead to state **s'** at time **t+1** is

$\boldsymbol{P_a}(s,s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$

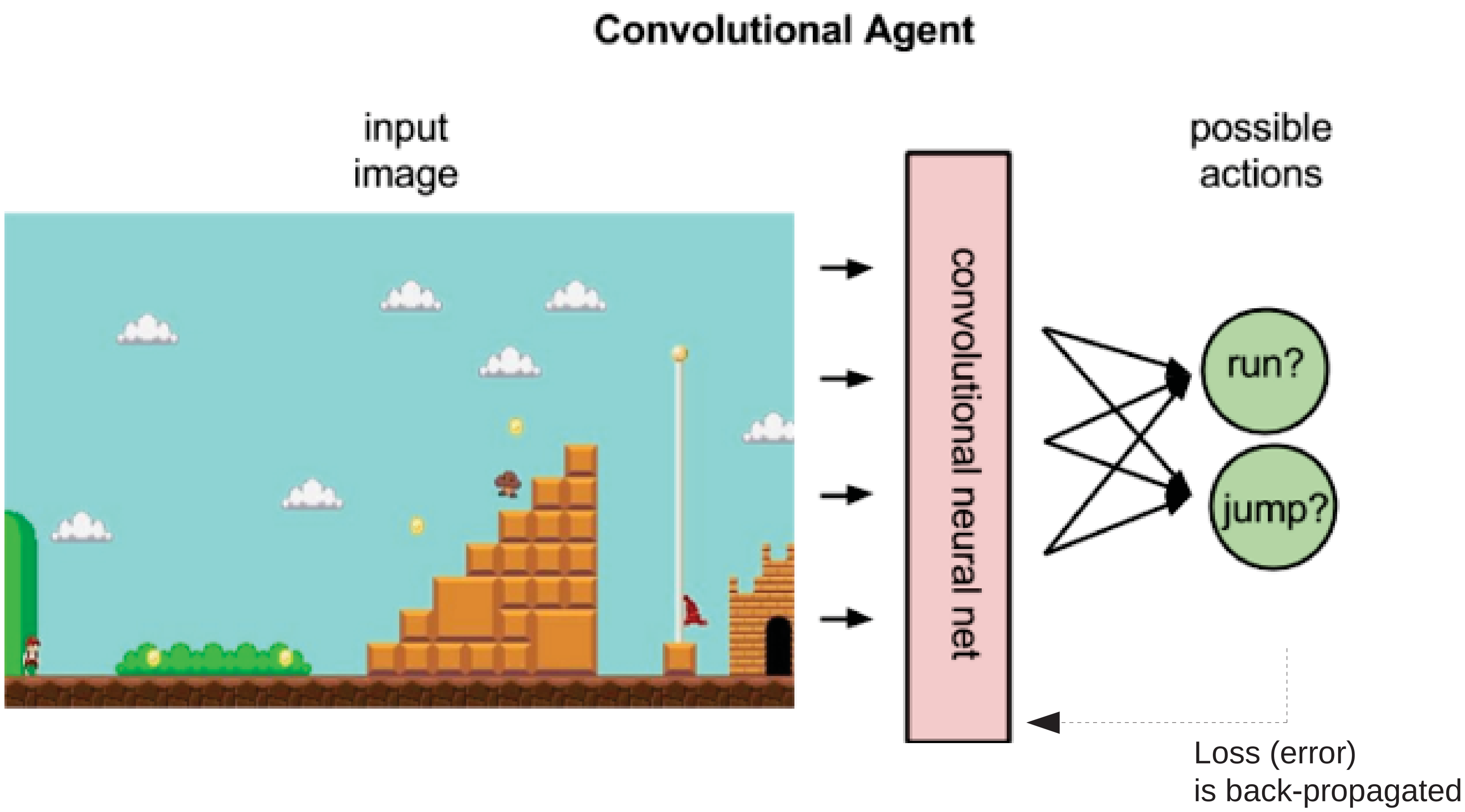


Milind Tambe team USC, CAIS

Good introduction :
Reinforcement Learning: a Survey
L. P. Kaelbling, M. L. Littman, W. Moore
1996.

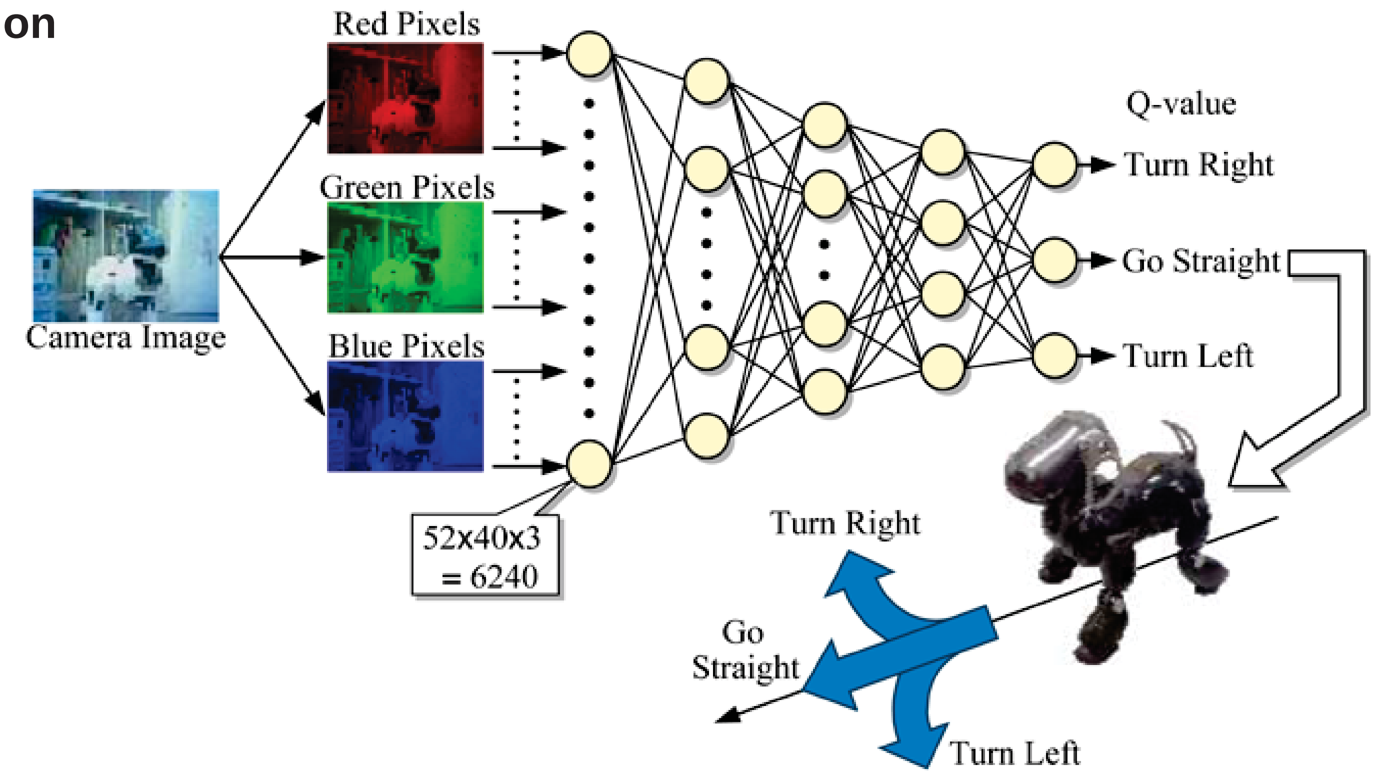
End to End Deep Learning

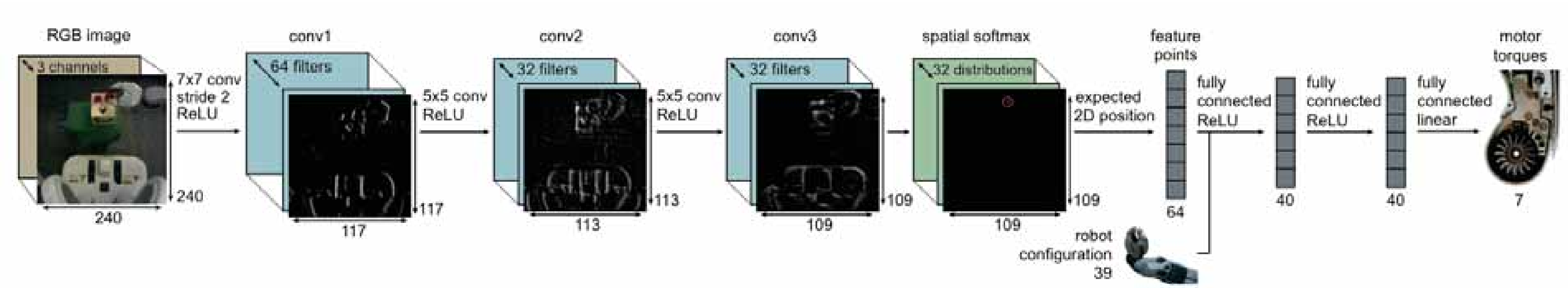
n processes → 1 single Neural Network



End to End Deep Learning

CNN allows generalization





Outline

Multi-Agent Systems (MAS)

Cooperate / Coordinate / Compete

Collective perspective (or reward) | individual perspective (or reward)

Multi-Agent Systems

Autonomous agents that interact in a common environment in order to fulfill collective and/or individual objectives

J. Ferber, Multi-Agent Systems, Addison Wesley, London, 1999

Multi-robot missions

EXPLORATION

TRANSPORT

Mapping (eg. SLAM)

Collective Trans.
(eg. box pushing)

Traffic regulation

Search, Coverage, Patrolling

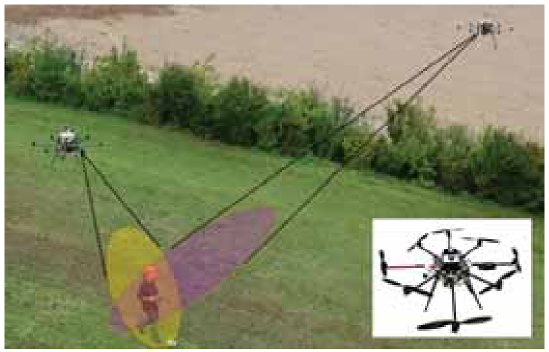
Autonomous fleet navig.

Tracking, Active perc.

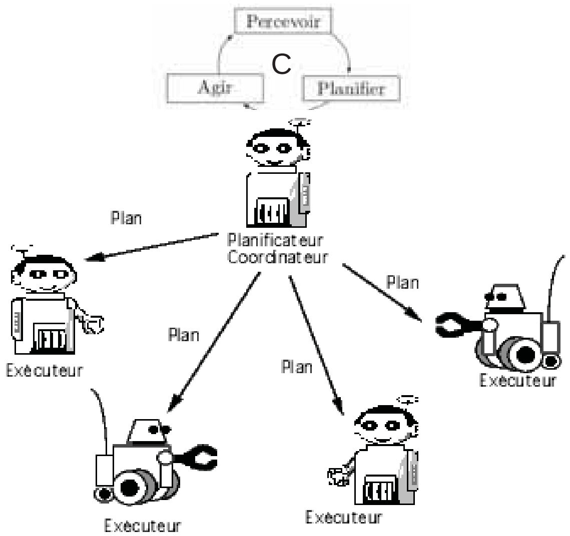
Connectivity

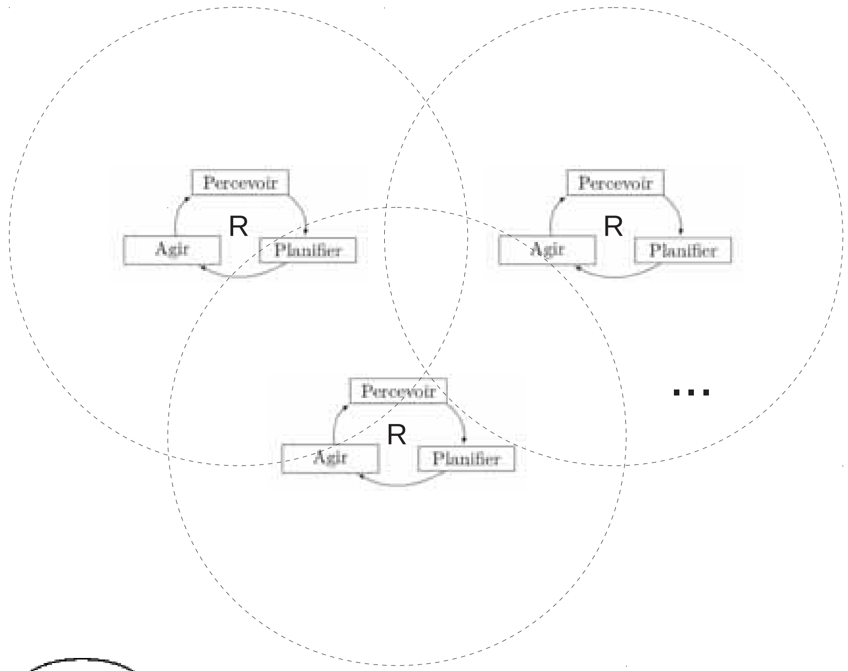
OBSERVATION

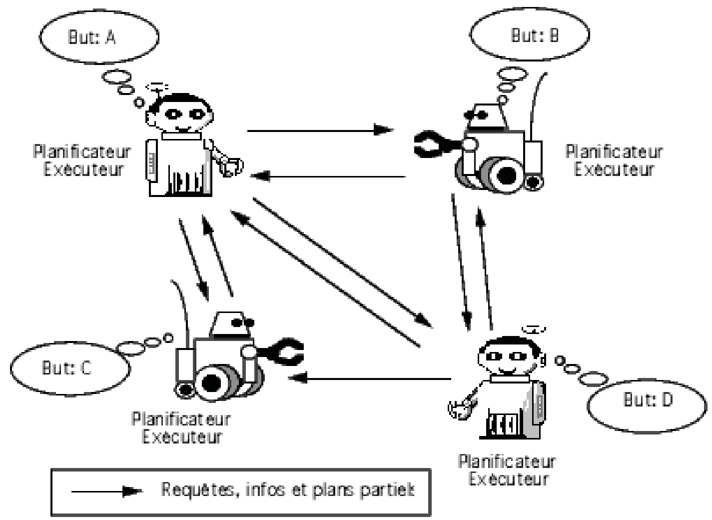
FORMATION MAINT.



Centralized, Decentralized and Distributed systems



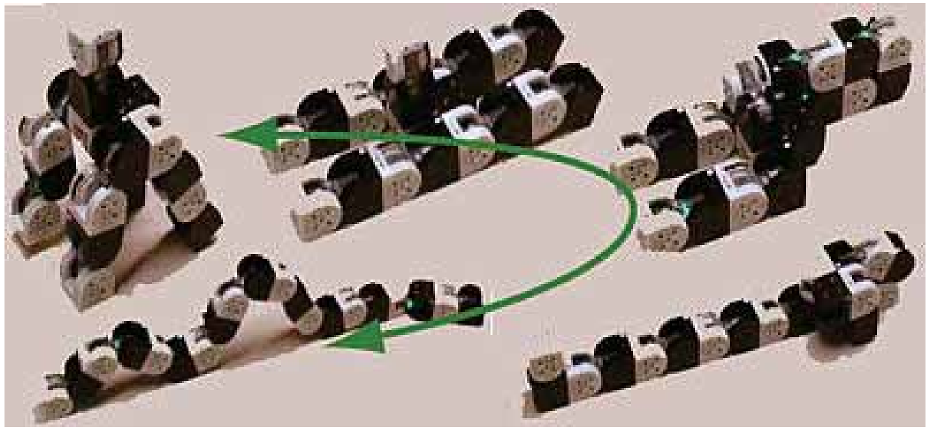




J. Ferber, Multi-Agent Systems, Addison Wesley, London, 1999

Decentralized : collective navigation and self-config.

Box-pushing 1988

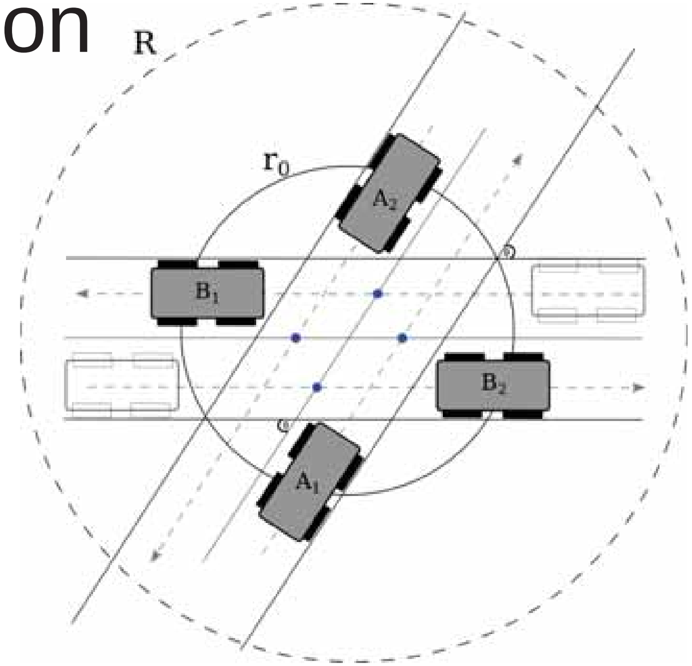


Self-configurable robots (M-TRAN III) 2005

Kilobots (2011)

CollMot project (UAV **flocking**) Pennsylvania 2012

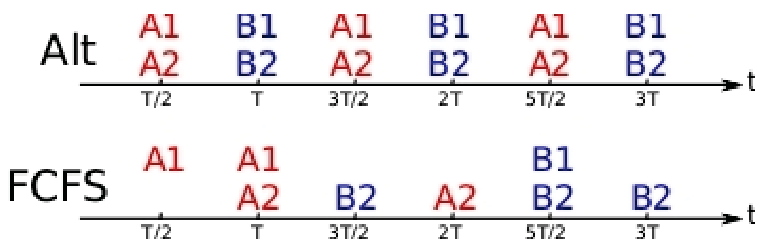
Local coordination at intersection

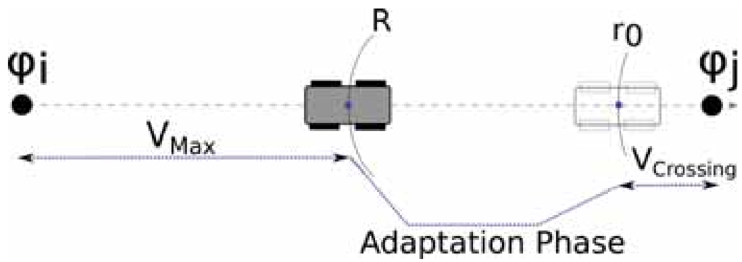


Non stop strategies

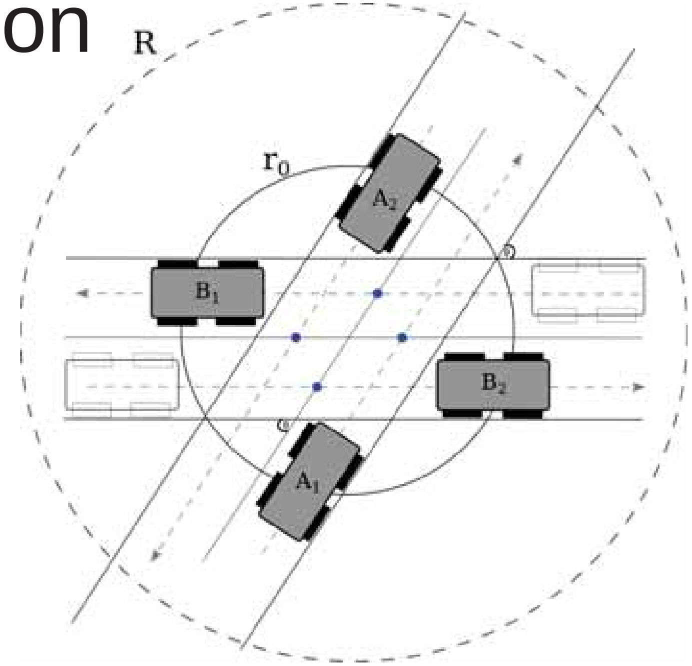
- First Come First Serve (**FCFS**)
  [Dresner Stone, 2005]

- **Alt**ernate [Tlig PhD, 2015]





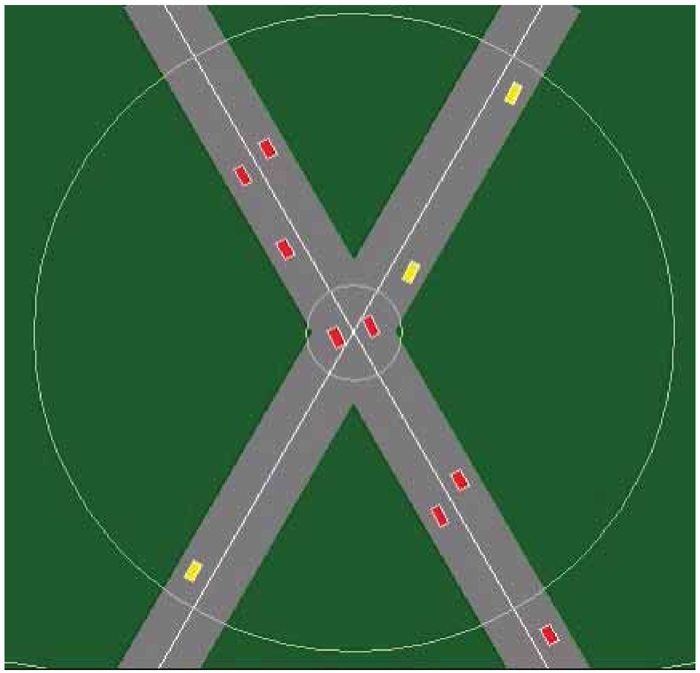
Local coordination at intersection



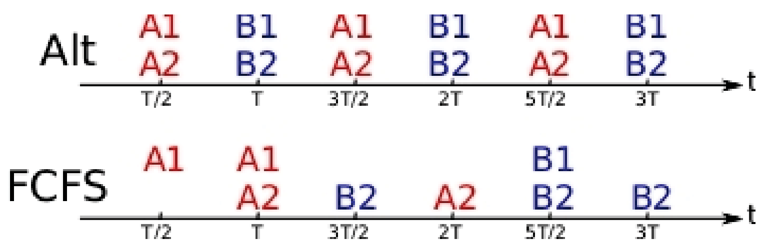
Non stop strategies

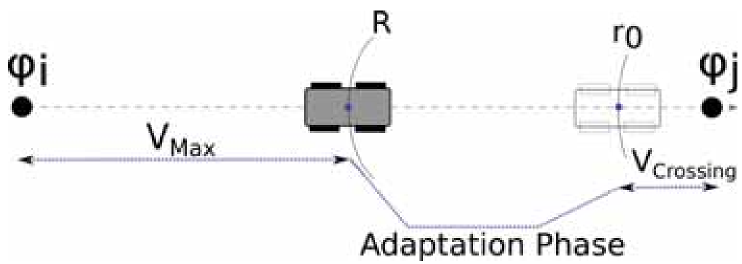
- First Come First Serve (**FCFS**)
  [Dresner Stone, 2005]

- **Alt**ernate [Tlig PhD, 2015]



Tlig, Buffet, Simonin, ECAI 2014





Local coordination at intersection

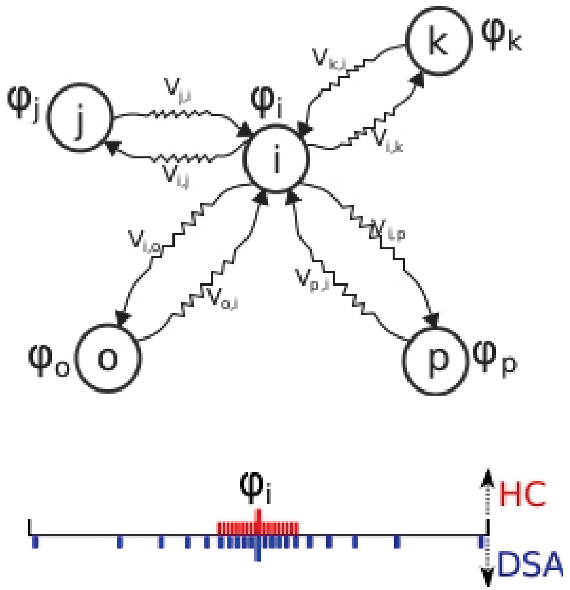
Cooperation between intersections

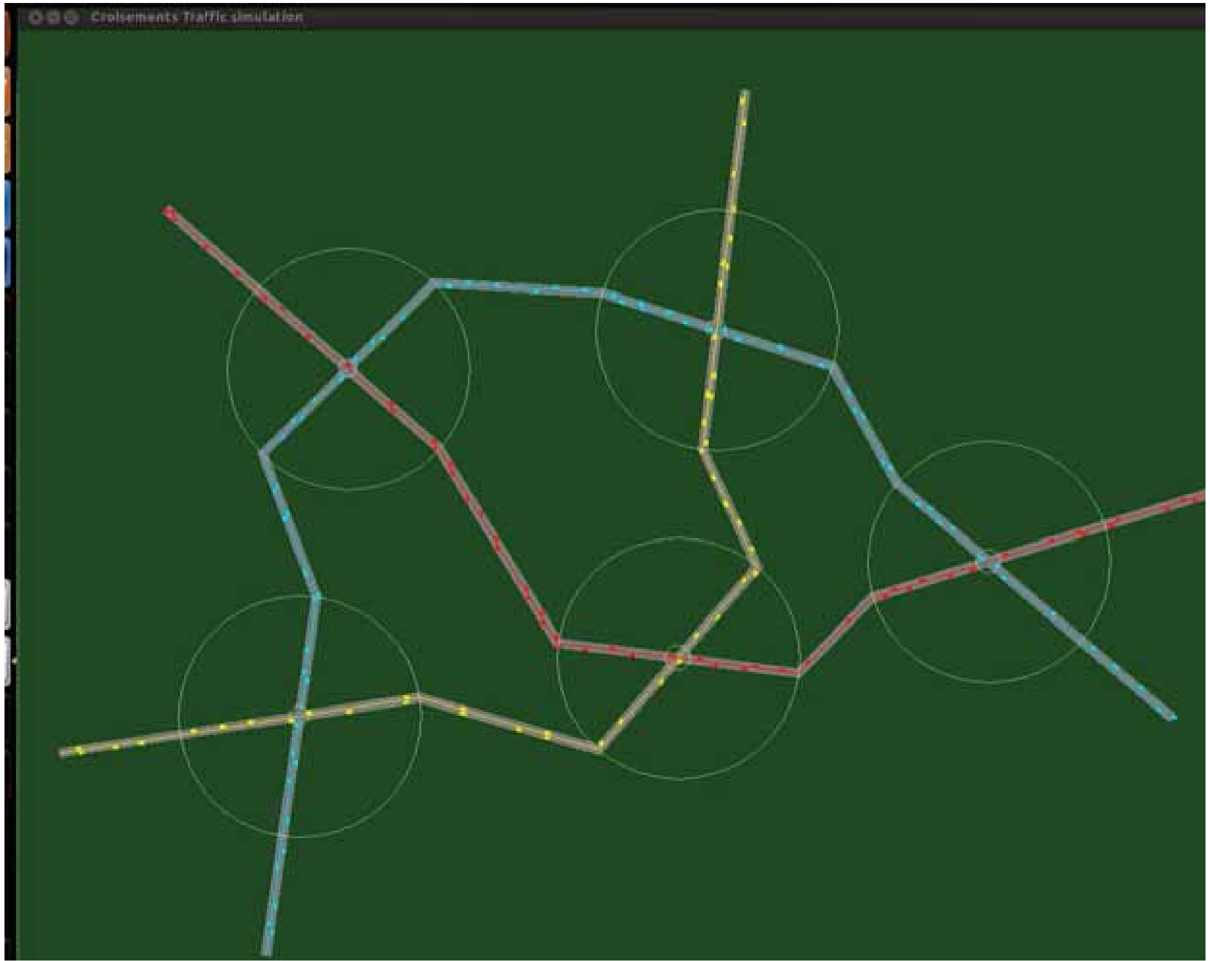
Decentralized (global) optimization → emergence of **green waves**

parameters : temporal phases φ_i

Optimizing time / energy → **online adaptation** to traffic changes

Hill Climbing, ρ-DSA

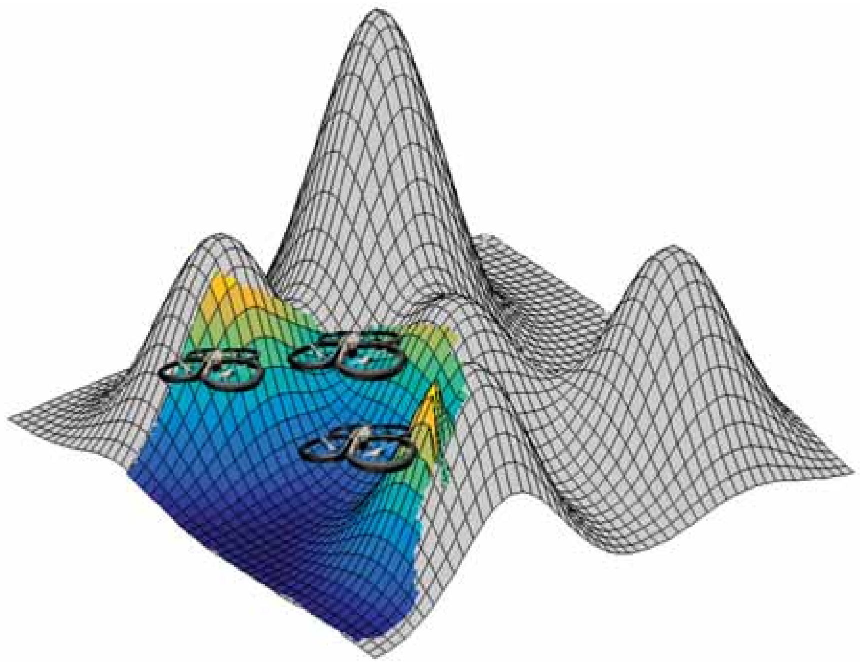




Outline

3D Mapping with a fleet of drones

A. Renzaglia, J. Dibangoye, O. Simonin

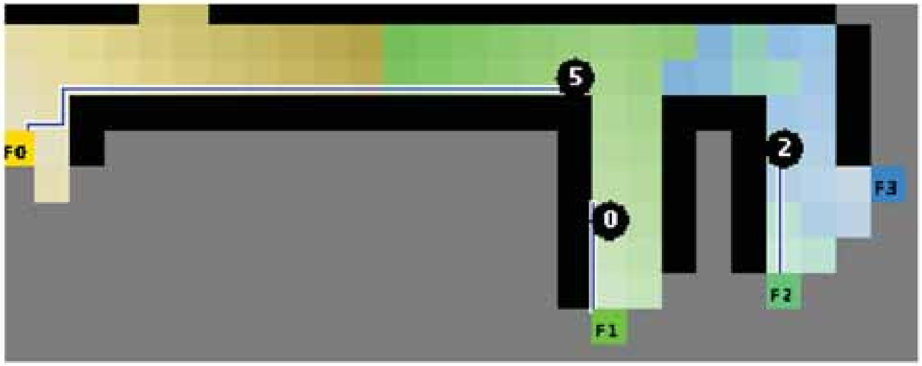


Strategy of exploration: approaches and limits

Mapping relies on visiting frontiers (known-unknown areas)

→ Repeat assignation robot-frontier : **min $\Sigma_{(i,j)}$ cost-Dist(r_i,f_j)**

[Yamauchi98] (mindist), [Burgard05] (greedy), [Bautin12] (minpos) ..

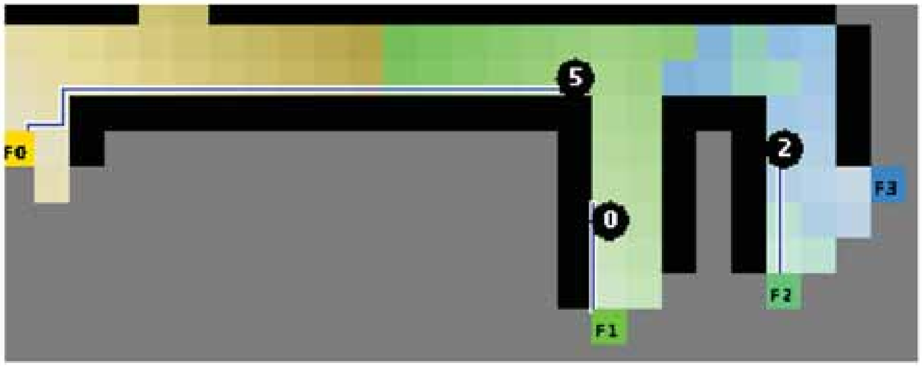


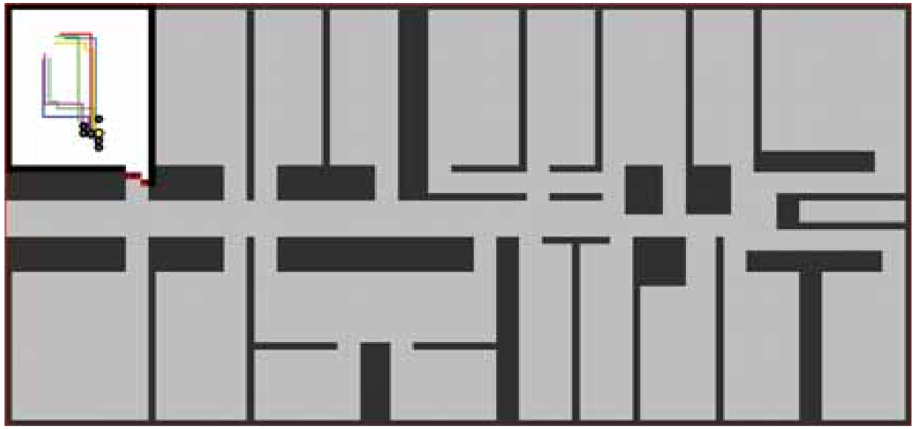
Strategy of exploration: approaches and limits

Mapping relies on visiting frontiers (known-unknown areas)

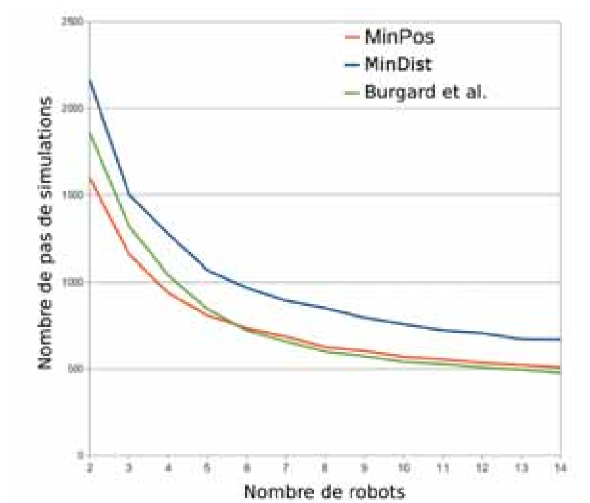
→ Repeat assignation robot-frontier : **min $\Sigma_{(i,j)}$ cost-Dist(r_i,f_j)**

[Yamauchi98] (mindist), [Burgard05] (greedy), [Bautin12] (minpos) ..





Multi-robot exploration : MinPos
[Bautin, Simonin, Charpillet, 2012], ANR Cartomatic



3D mapping : exploit optimal coverage

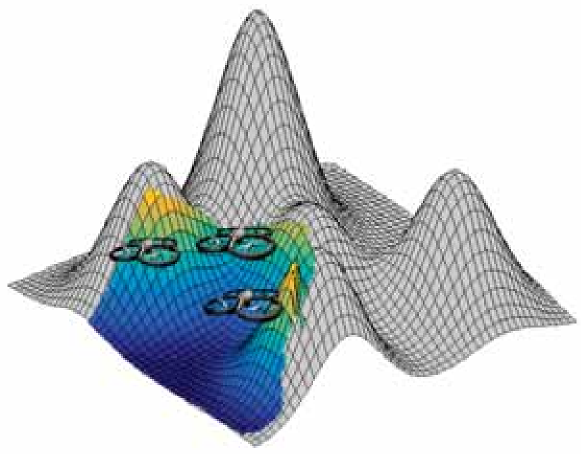
Mapping relies on visiting frontiers (known-unknown areas)

→ Repeat assignation robot-frontier : $\mathbf{min}\ \boldsymbol{\Sigma}_{(i,j)}\ \mathbf{cost\text{-}Dist(r_i, f_j)}$

[Yamauchi98] (mindist), [Burgard05] (greedy), [Bautin12] (minpos) ..

Sub-problem: **optimal coverage**

→ deploying robots such as **maximizing the observed area**



3D mapping : exploit optimal coverage

Mapping relies on visiting frontiers (known-unknown areas)

→ Repeat assignation robot-frontier : **min $\Sigma_{(i,j)}$ cost-Dist(r_i,f_j)**

[Yamauchi98] (mindist), [Burgard05] (greedy), [Bautin12] (minpos) ..

Sub-problem: **optimal coverage**

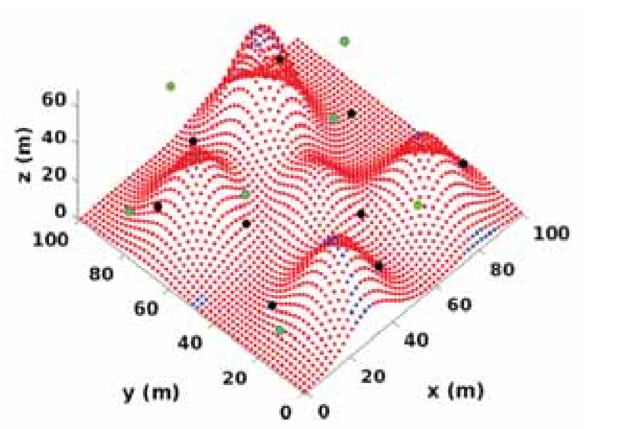
→ deploying robots such as **maximizing the observed area**

CAO (Cognitive-based Adaptive Optimization) [Renzaglia et al. 12]

Local approx. of the **objective** function : $\hat{J}(x_1(t), .. x_N(t))$ ← **measures** (short T)

Move : **Stochastic search** on $\hat{J}$

→ converge to a <u>local</u> optimum



Combining local search and frontier-based app.

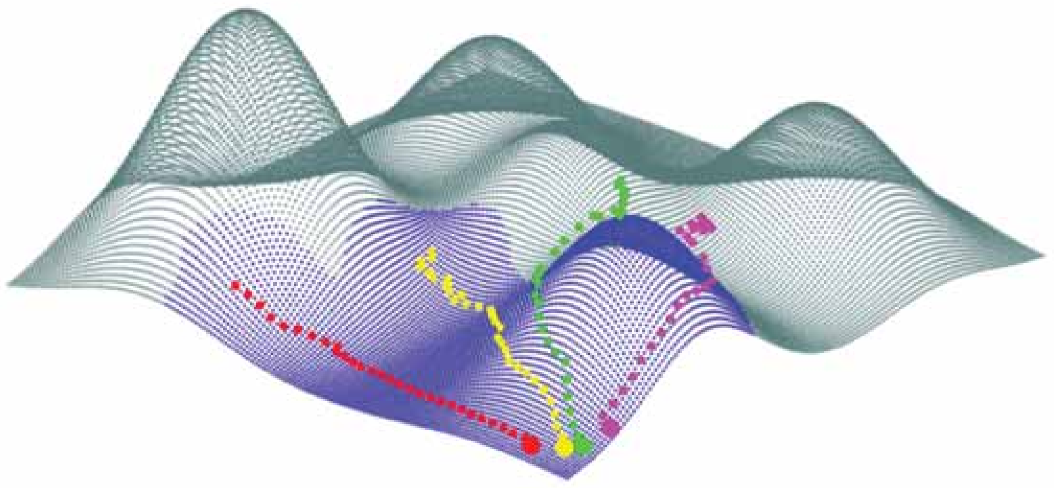
For each robot

Repeat

Follow local search

When a local minima is reached **Move** to the closest frontier

Until no more frontier



Combining local search and frontier-based app.

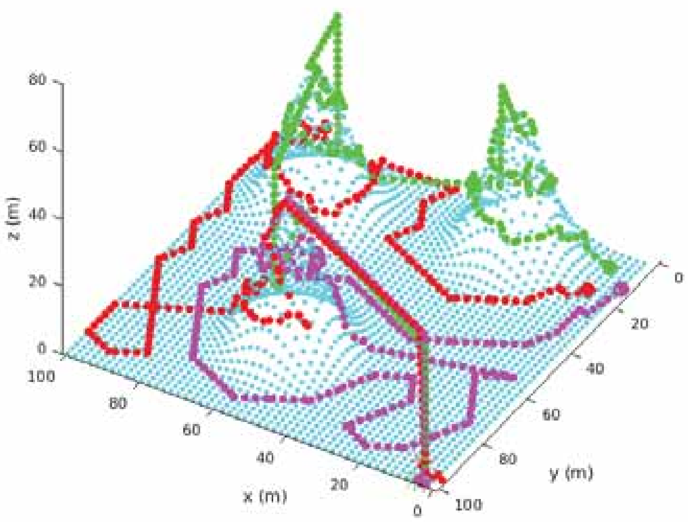
For each robot

Repeat

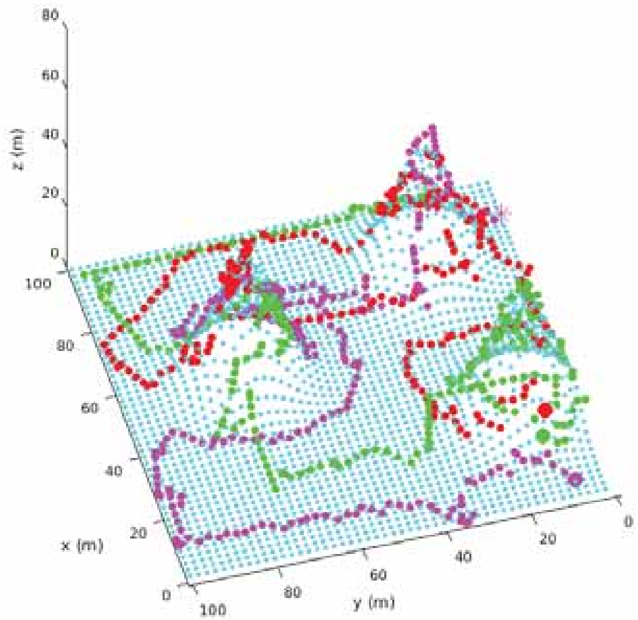
Follow local search

When a <u>local minima is reached</u> **Move** to the <u>closest frontier</u>

Until no more frontier



Only frontiers



Local search + frontiers

Combining local search and frontier-based app.

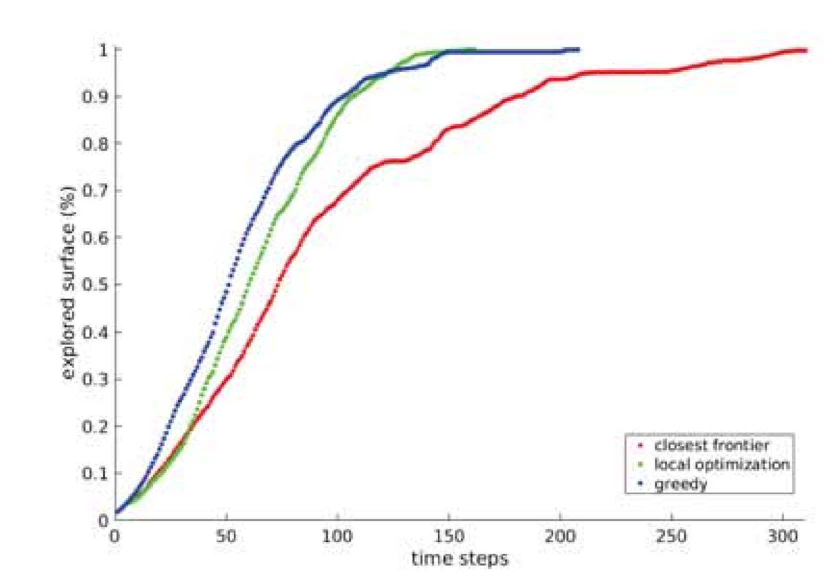
For each robot

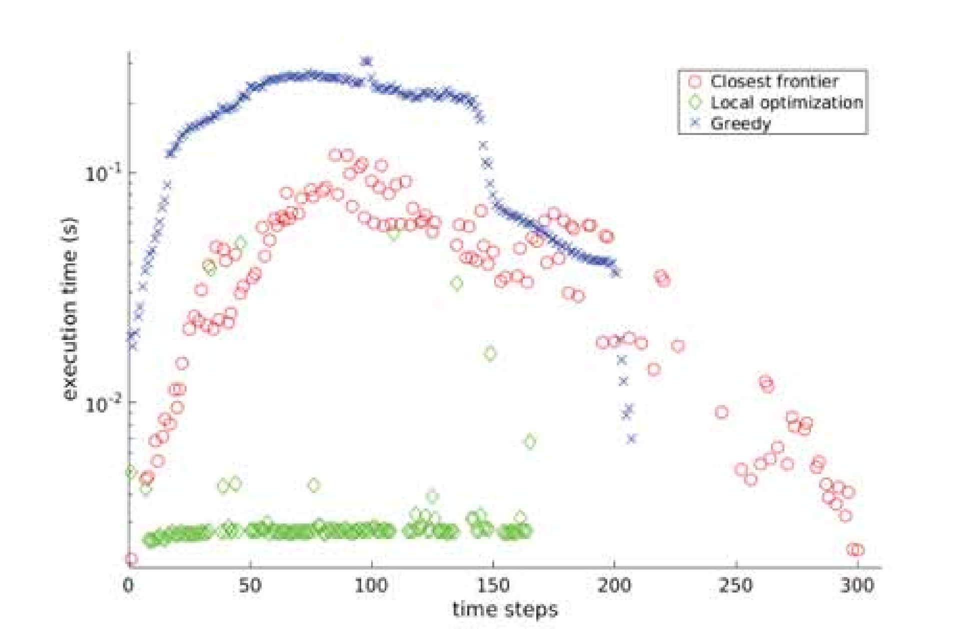
Repeat

Follow local search

When a local minima is reached **Move** to the closest frontier

Until no more frontier





Thank you !

https://team.inria.fr/chroma/