Learning Tasks in Robotics: Problems and Solutions

Nuno Lau

nunolau@ua.pt

IEETA – Institute of Electronics and Informatics Engineering of Aveiro

DETI / Universidade de Aveiro

Summary

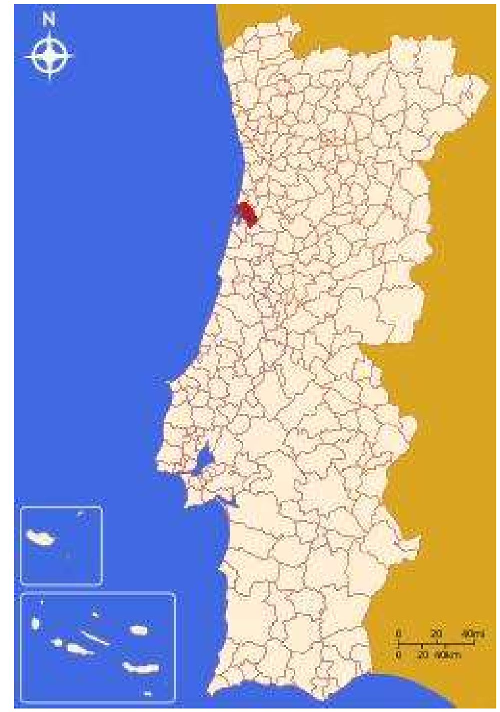
- Presentation
- Motivation
  - Robotics Learning Problems
- Some solutions
  - Gesture recognition
  - Q-Batch update rule
  - Multi-context optimization
  - User profiles and Adapted interfaces
  - Multiagent Learning
- Conclusion

Summary

- **Presentation**
- Motivation
  - Robotics Learning Problems
- Some solutions
  - Gesture recognition
  - Q-Batch update rule
  - Multi-context optimization
  - User profiles and Adapted interfaces
  - Multiagent Learning
- Conclusion

Presentation

- Aveiro, Portugal
  - Capital of Aveiro District
    - 68 km South of Oporto
    - 258 km North of Lisbon
  - Population: 78 000
  - Water channels crossing the city

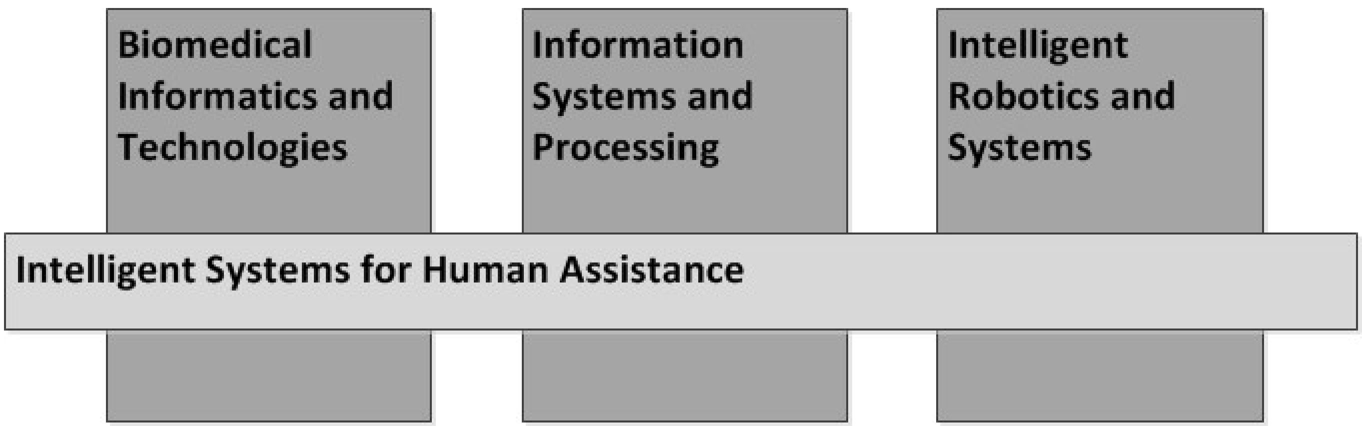


Presentation

- Universidade de Aveiro
  - Founded in 1973
  - 13 000 students
  - 13 Research Units
    - 77% Excellent or V. Good
  - Domains
    - Science and Engineering,
    - Communication and Art,
    - Social Sciences,
    - Health,
    - Humanities
    - Education

Presentation

- IEETA - Institute of Electronics and Informatics Engineering of Aveiro
  - Mission:
    Multidisciplinary research and advanced development in Electronics and Telematics



Summary

- Presentation
- **Motivation**
  - Robotics Learning Problems
- Some solutions
  - Gesture recognition
  - Q-Batch update rule
  - Multi-context optimization
  - User profiles and Adapted interfaces
  - Multiagent Learning
- Conclusion

Motivation

Programming Robots is a hard task

- No high-level programming language
- Sensors and actuators are noisy
- Robotics is moving towards increasingly unstructured environments

If only **robots could learn** how **to perform tasks** by themselves…

$\Rightarrow$ **Machine Learning in Robotics**

Motivation

Table-Tennis

Robots

Mülling + Peters

Humans

We need **learning** and **adaptation** to improve robot skills!

Motivation

Machine Learning in Robotics can be used for:

- Robot **Perception**
- Robot **Decision**
- Robot **Actuation** (Behaviors)
- Multi-robot **Coordination**
- Adapt **Human-Robot Interaction**

Motivation

Challenges in Robot Learning

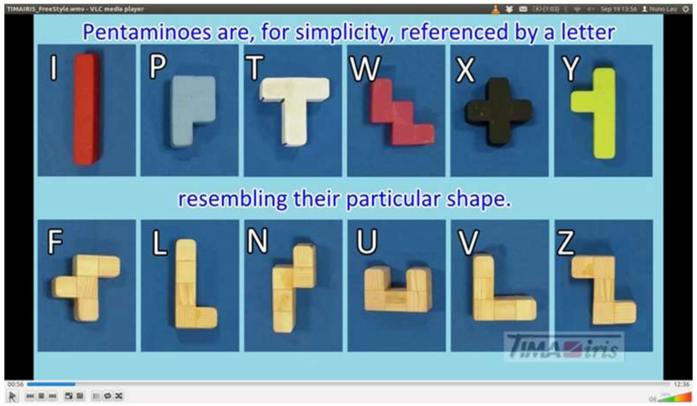
- Cost of experimentation
- Cost of failure
- Limited data
- Generalization
- Curse of dimensionality
- Real time requirements
- Changes in environment
- Changes in task specification

Summary

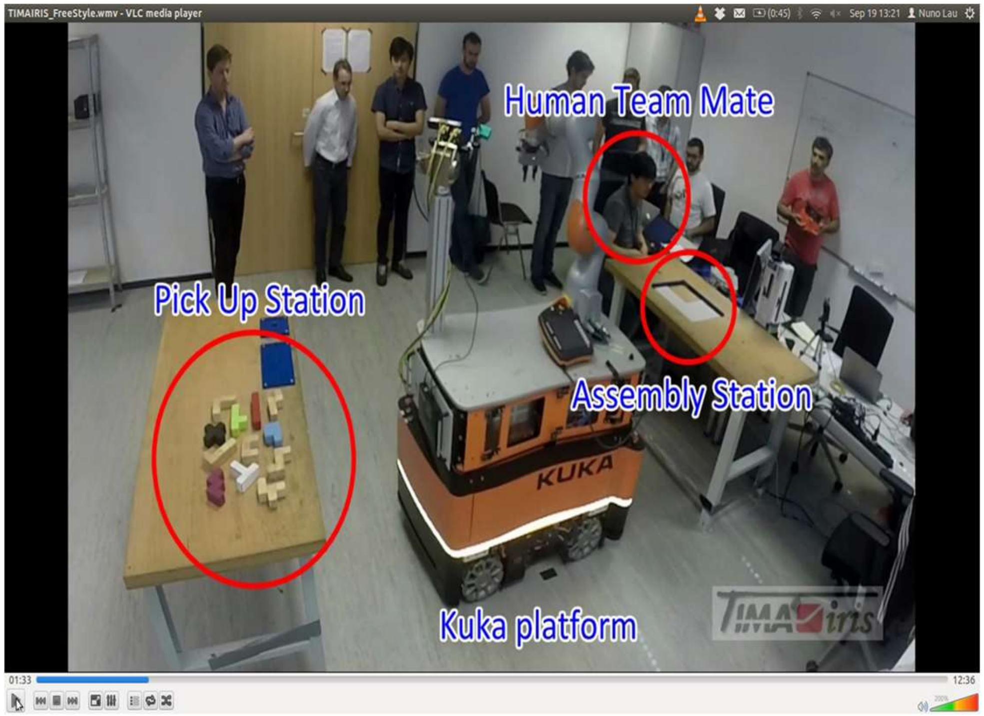
- Presentation
- Motivation
  - Robotics Learning Problems
- Some solutions
  - **Gesture recognition**
  - Q-Batch update rule
  - Multi-context optimization
  - User profiles and Adapted interfaces
  - Multiagent Learning
- Conclusion

Gesture Recognition

Task: **Assembling a puzzle cooperatively by a human and a robot** (EuRoC Project)

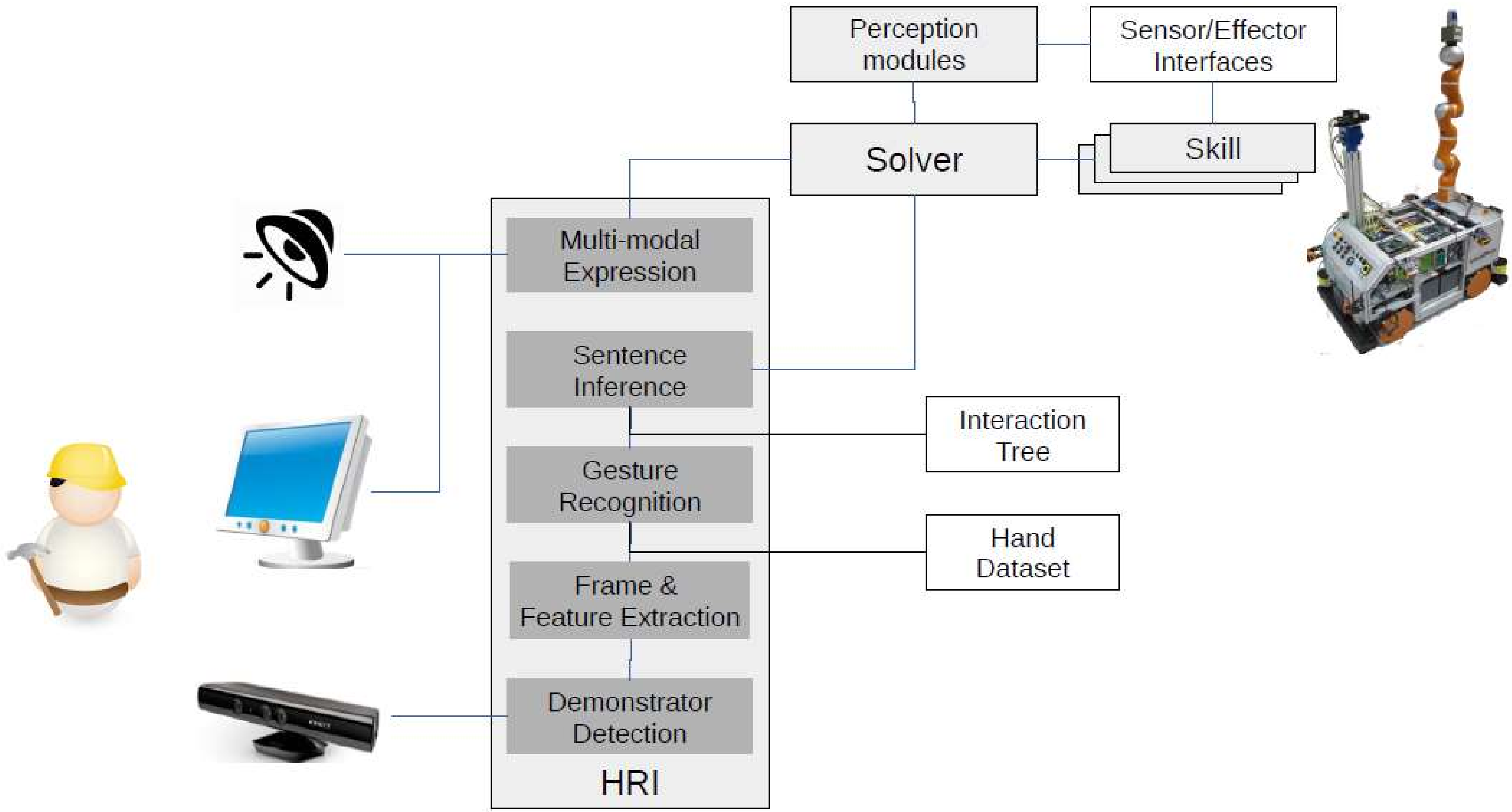


Set of 12 pentomino pieces



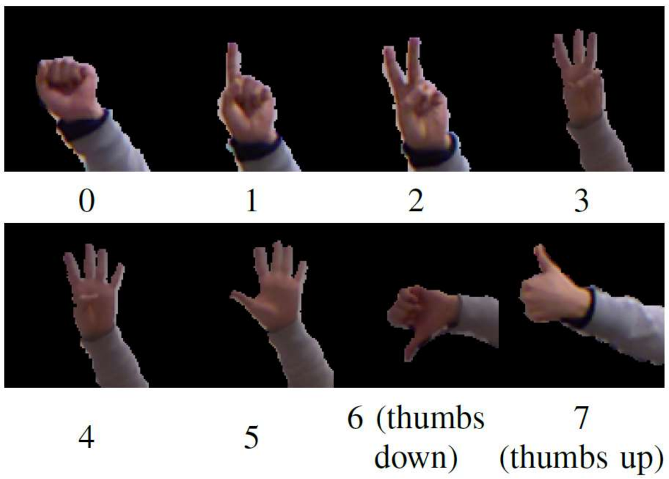
Task environment

Gesture Recognition



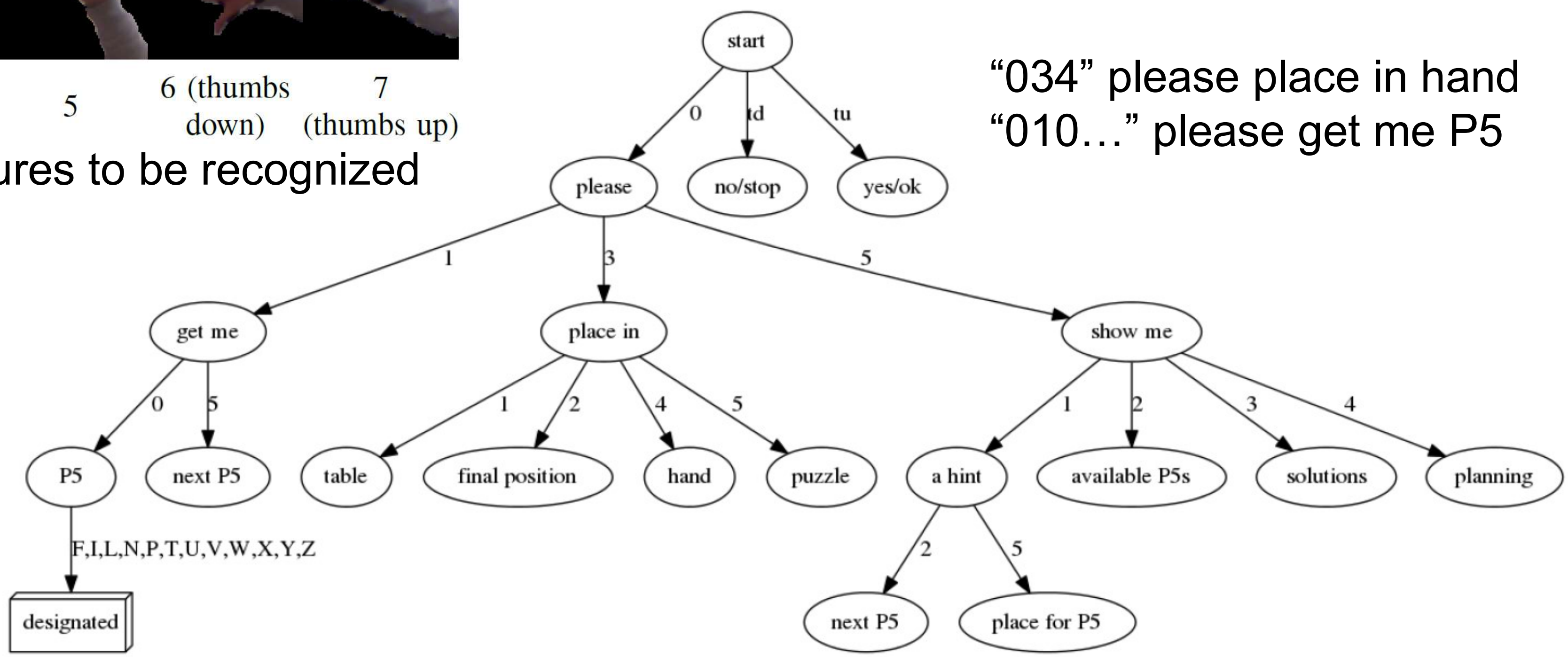
HRI architecture

Gesture Recognition



Gestures to be recognized

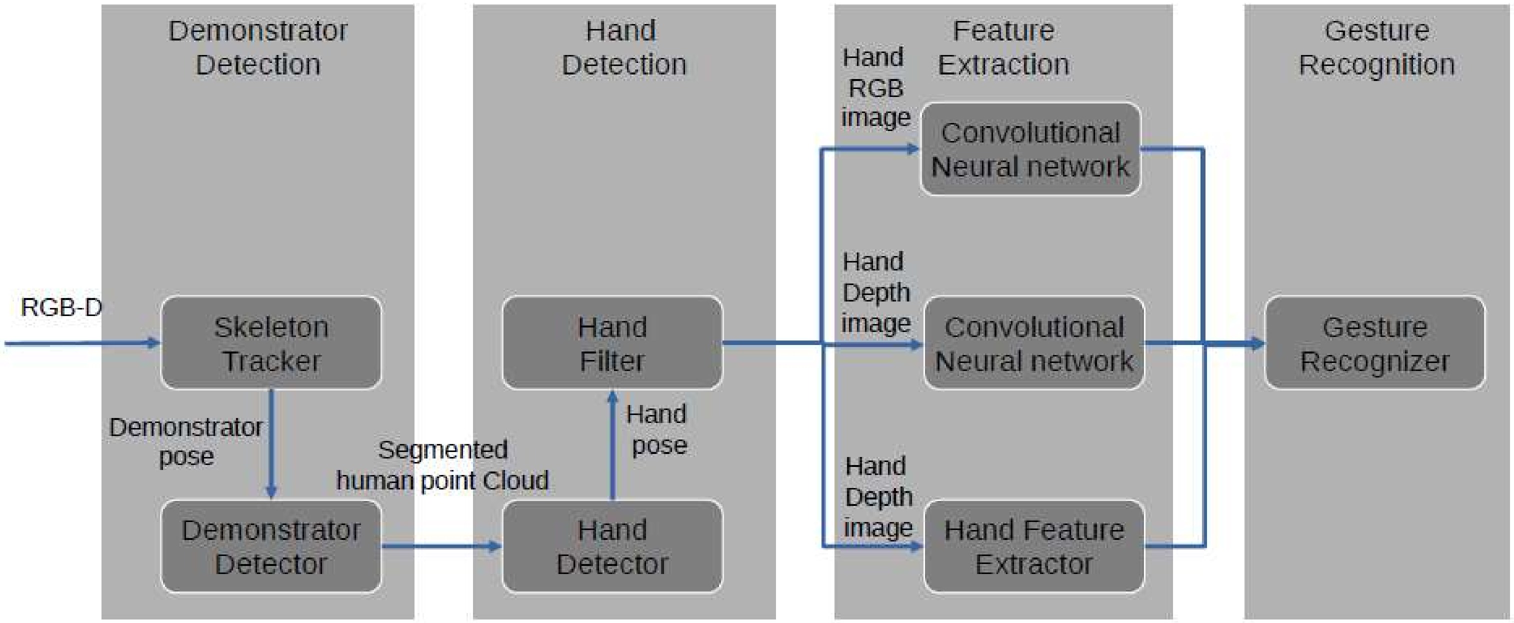
“034” please place in hand
“010…” please get me P5



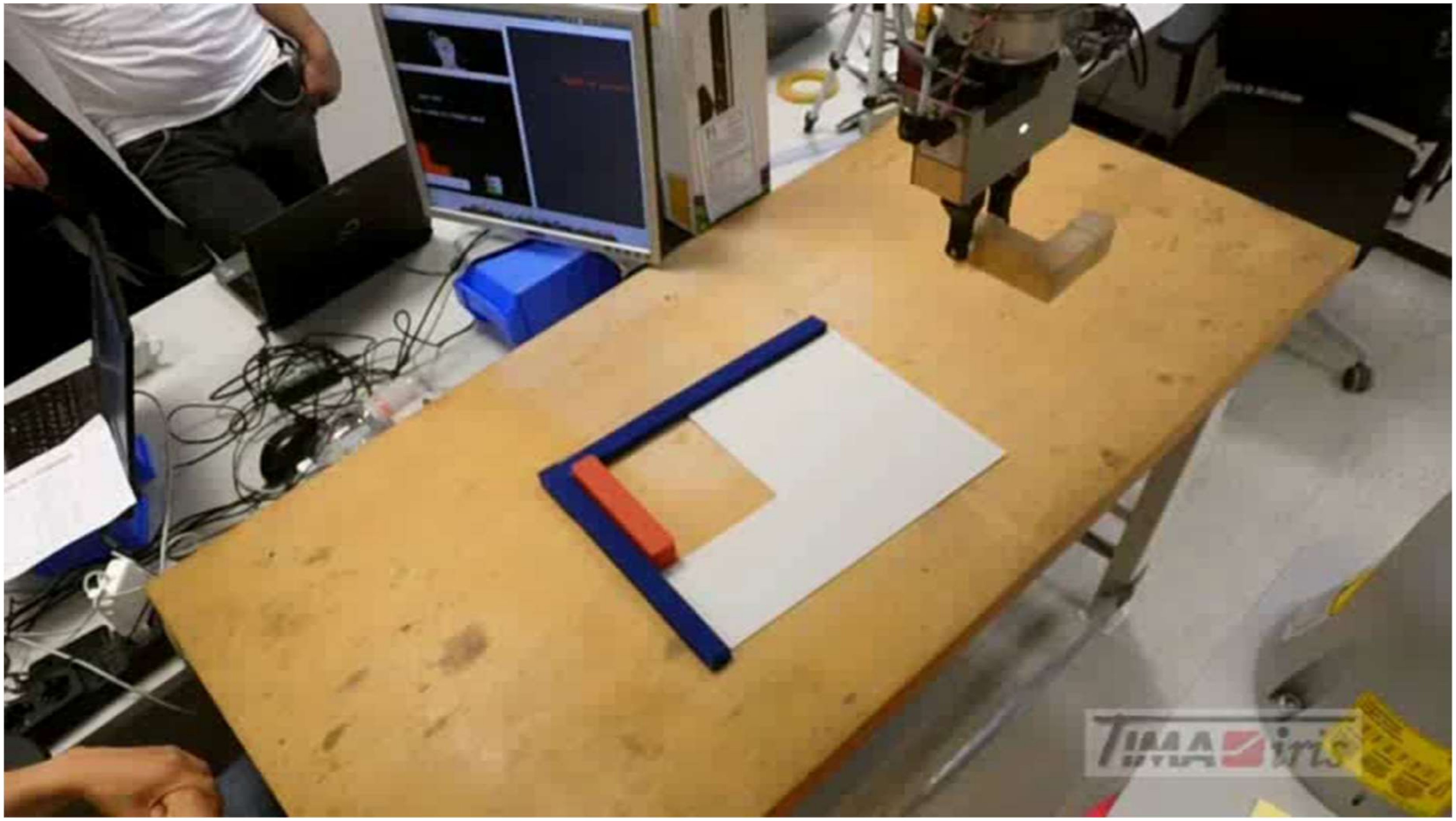
Interaction tree

Gesture Recognition

- Learning Task: **Recognize Gestures**
- Approach:
  - 1st : Use Deep Learning
  - 2nd : Mix Deep Learning with other features



Gesture Recognition

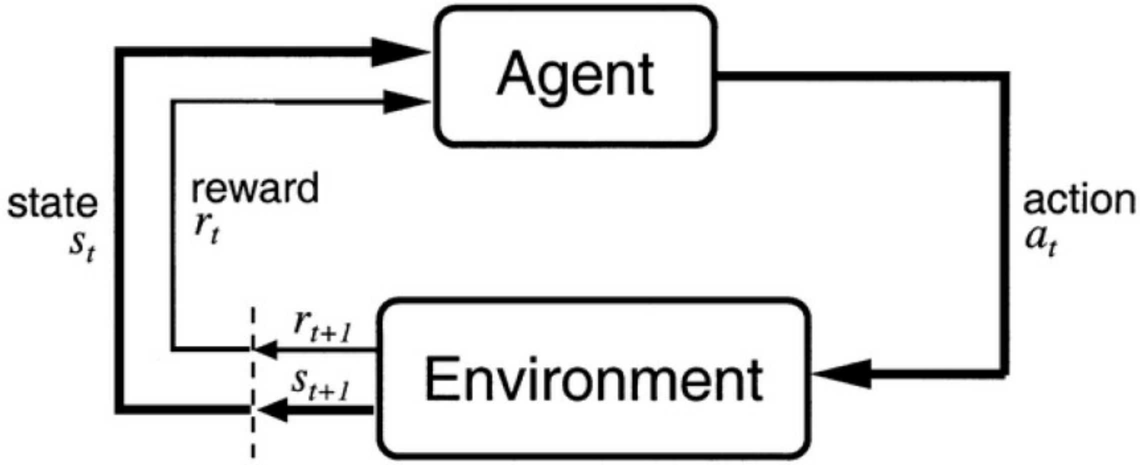


Presentation outline

- Presentation
- Motivation
  - Robotics Learning Problems
- Some solutions
  - Gesture recognition
  - **Q-Batch update rule**
  - Multi-context optimization
  - User profiles and Adapted interfaces
  - Multiagent Learning
- Conclusion

Q-Batch update Rule

- **Reinforcement Learning**



- Goal: Determine the **policy** that maximizes *Return*

$$R_t = \sum_{k=0}^{+\infty} \gamma^k r_{k+t+1}$$

Q-Batch update Rule

Three **main RL classes** of methods

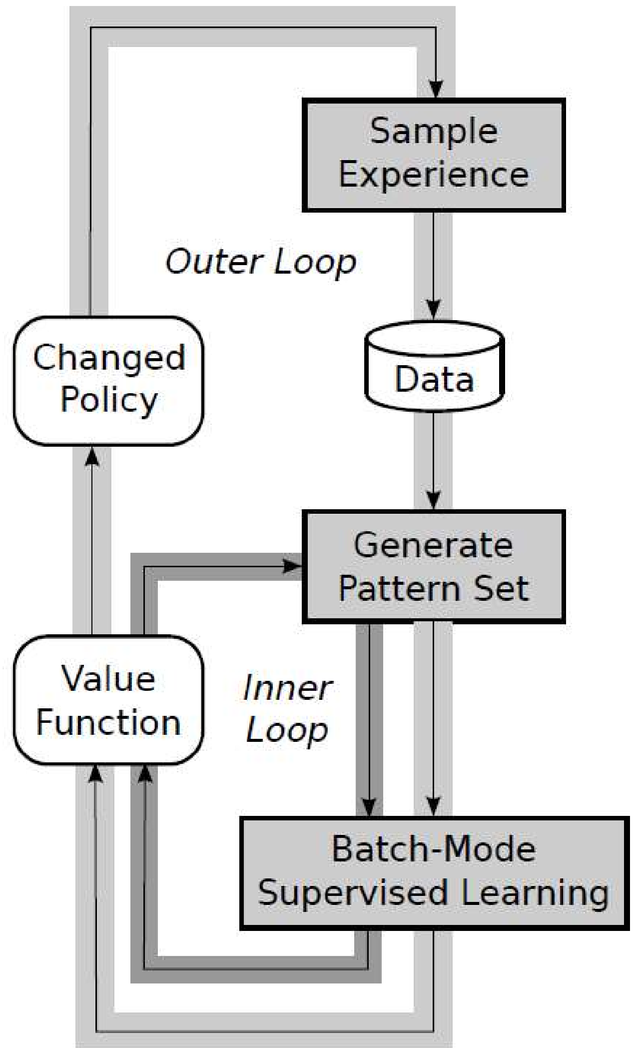
- **Value Function based** methods
  - No policy representation
  - Policy obtained by evaluating the value function directly
- **Policy Search** methods
  - No value function
  - Optimization of a parametrized policy directly on policy-space
- **Actor-Critic** methods
  - Value function (critic)
  - Explicit Policy representation (actor)

- **Batch RL is a sub-class of Value Function based methods**

Batch Reinforcement Learning

- **Batch RL** estimates value functions by processing **a set of interactions**
- The value function is updated synchronously
- Application of function approximators
- Collected experience is **not discarded**
- Data **efficient**
- Fitted Q iteration:

$$\bar{Q}_i = r_i + \gamma \max_b \hat{Q}(s_{i+1}, b)$$



Q-Batch update rule

- **Still:**
  - Q-Learning is **transition based**
  - **Not considering trajectories**
  - **Many inner-loops** for reward propagation

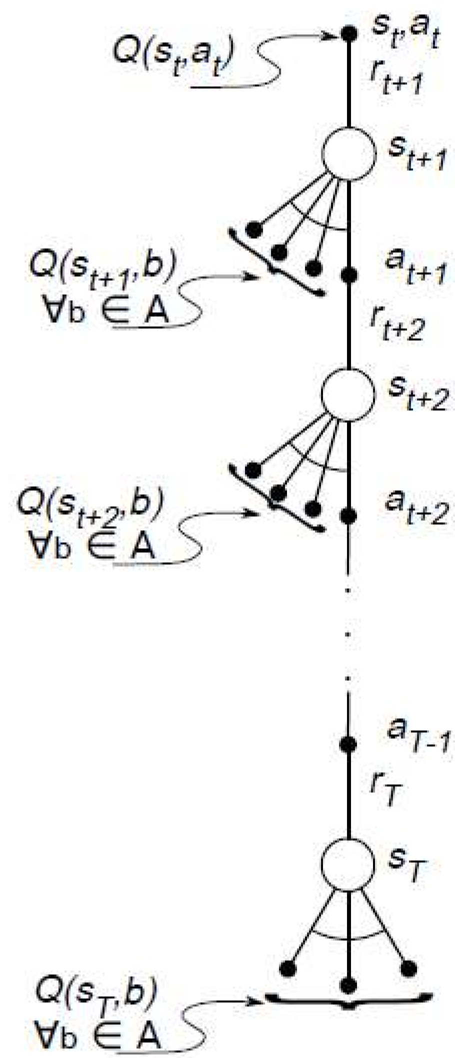
- In Batch RL **all data is available**

⇒ Q-Batch update rule

- Find largest n-step return

$$\bar{Q}(s_{i,j}, a_{i,j}) = \max_k R_{i,j}^{(k)}$$

$$= \max_k \left(\sum_{l=0}^{k-1} (\gamma^l r_{i,j+1+l}) + \gamma^k \max_{b \in A} \hat{Q}(s_{i,j+k}, b) \right)$$



João Cunha, et al.. Batch Reinforcement Learning for Robotic Soccer Using the Q-Batch Update-Rule. Journal of Intelligent & Robotic Systems, vol. 80, no. 3, p. 385-399, December 2015

Q-Batch update rule

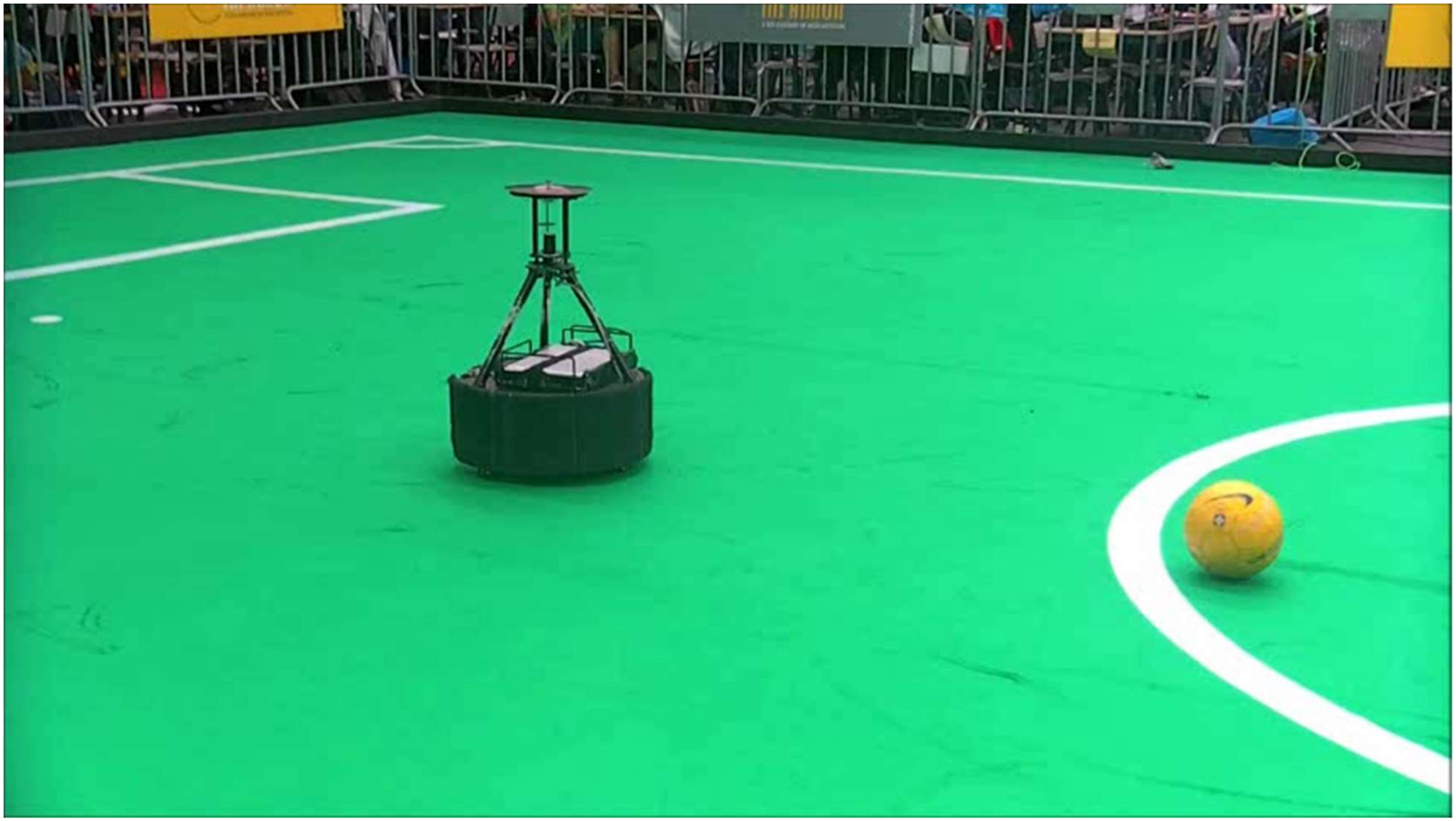
- Results on Simulated Inverted Pendulum

Deterministic	best policy	interaction time first suitable policy (in minutes)	number of suitable policies
Q-learning	0.41 ± 0.01	7.05 ± 1.07	352.0 ± 32.3
Watkins-Q(1)	0.40 ± 0.01	17.65 ± 15.58	306.0 ± 74.5
Q-Batch	$\mathbf{0.40 \pm 0.01}$	$\mathbf{10.67 \pm 6.64}$	$\mathbf{359.3 \pm 22.1}$

Stochastic	best policy	interaction time first suitable policy (in minutes)	number of suitable policies
Q-learning	1.03 ± 0.18	20.51 ± 35.48	67.3 ± 81.4
Watkins-Q(1)	1.12 ± 0.20	67.22 ± 50.03	74.0 ± 118.4
Q-Batch	$\mathbf{0.89 \pm 0.02}$	$\mathbf{17.83 \pm 16.48}$	$\mathbf{228.8 \pm 58.8}$

Q-Batch update rule

Q-Batch update rule

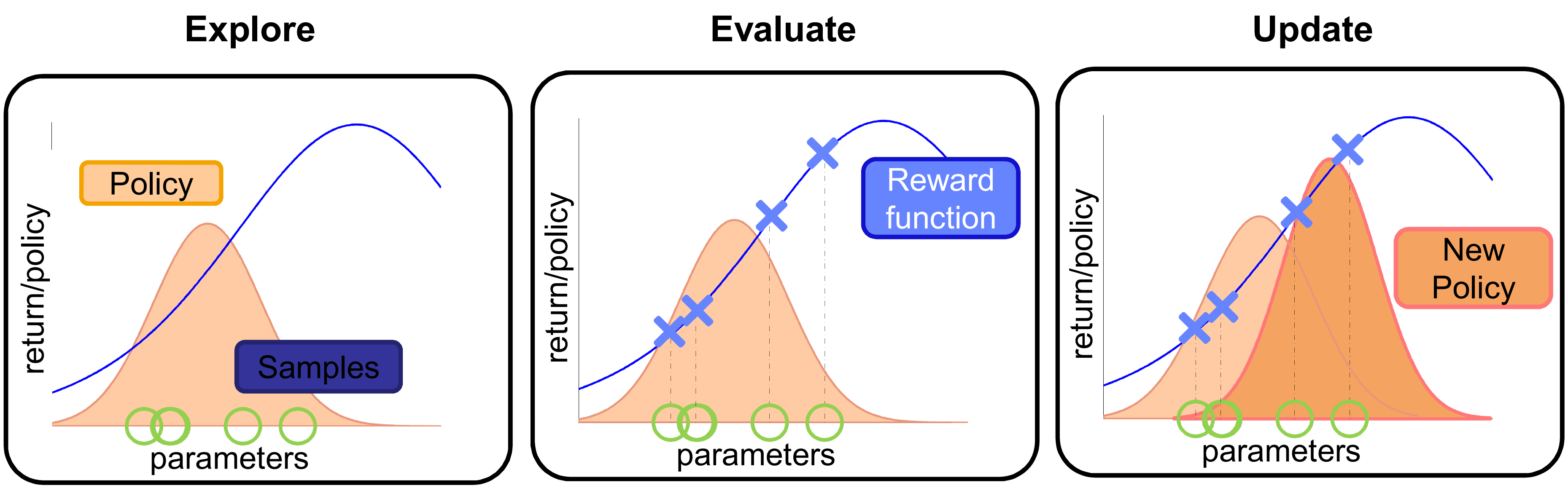


Presentation outline

- Presentation
- **Motivation**
  - Robotics Learning Problems
- Some solutions
  - Gesture recognition
  - Q-Batch update rule
  - **Multi-context optimization**
  - User profiles and Adapted interfaces
  - Multiagent Learning
- Conclusion

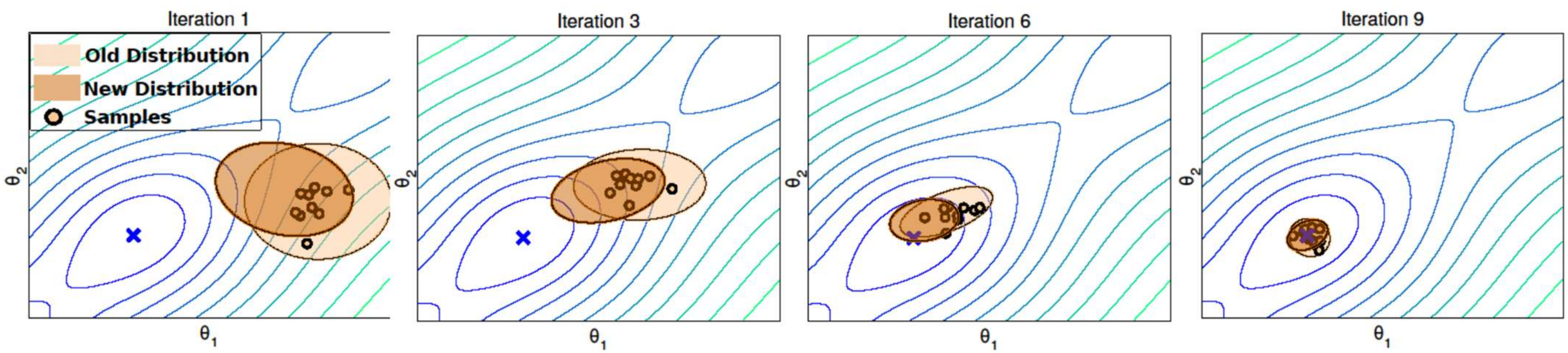
Stochastic Search

- **Use Search-Distribution:** $\pi(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- **Objective:** Find search distribution $\pi(\boldsymbol{w})$ that maximizes $J_\pi = \int \pi(\boldsymbol{w}) R(\boldsymbol{w}) d\boldsymbol{w}$



Stochastic Search

- **Use Search-Distribution:** $\pi(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- **Objective:** Find search distribution $\pi(\boldsymbol{w})$ that maximizes $J_\pi = \int \pi(\boldsymbol{w}) R(\boldsymbol{w}) d\boldsymbol{w}$



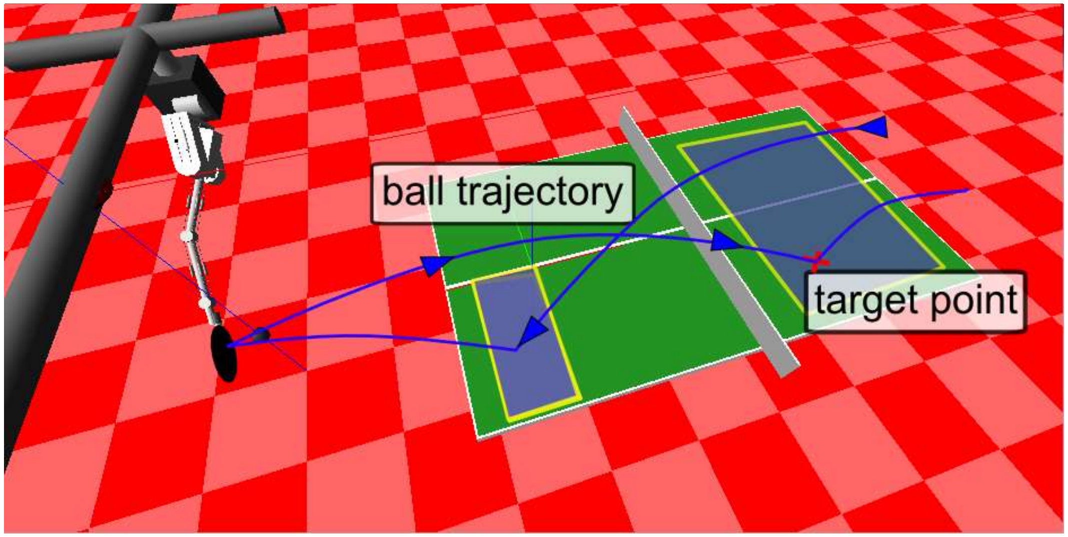
Contextual Stochastic Search

Goal: Adapt parameters $\boldsymbol{w}$ to different situations

- Different ball trajectories
- Different target locations

Introduce context vector $\boldsymbol{s}$

- Continuous valued vector
- Characterizes environment and objectives of agent



Learn contextual search policy $\pi(\boldsymbol{w}|\boldsymbol{s})$

Abdolmaleki, et. al, *Model-Based Relative Entropy Stochastic Search,* NIPS 2015
Kupcsik, et. al, *Model-based Contextual Policy Search for Data-Efficient Generalization of Robot Skills*, *Artificial Intelligence, 2015*

Adaptation of Skills

Contextual distribution:

$$\pi(\boldsymbol{w}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{s}^T\boldsymbol{M} + \boldsymbol{m}, \boldsymbol{\Sigma})$$

Compatible Function Approximation:

$$R(\boldsymbol{s}, \boldsymbol{w}) \approx \boldsymbol{w}^T\boldsymbol{A}\boldsymbol{w} + \boldsymbol{s}^T\boldsymbol{B}\boldsymbol{w} + \boldsymbol{a}^T\boldsymbol{w} + a_0$$

Contextual distribution update:

1. Maximize **expected** return

$$\arg\max_{\pi} \mathbb{E}_{p(\boldsymbol{s})}\left[\int \pi(\boldsymbol{w}|\boldsymbol{s})R(\boldsymbol{s}, \boldsymbol{w})d\boldsymbol{w}\right]$$

2. Bound **expected** information loss

$$\text{s.t.:} \quad \mathbb{E}_{p(\boldsymbol{s})}\left[\mathrm{KL}\big(\pi(\cdot|\boldsymbol{s})||\pi_{\text{old}}(\cdot|\boldsymbol{s})\big)\right] \leq \epsilon$$

3. Bound entropy loss

$$\underbrace{H(\pi_{\text{old}}) - H(\pi)}_{\text{loss in entropy}} \leq \gamma$$

New distribution:

$$\pi(\boldsymbol{w}|\boldsymbol{s}) \propto \pi_{\text{old}}(\boldsymbol{w}|\boldsymbol{s})^{\frac{\eta}{\eta+\omega}} \exp\left(\frac{R(\boldsymbol{s}, \boldsymbol{w})}{\eta + \omega}\right)$$

$$\propto \mathcal{N}(\boldsymbol{s}^T\boldsymbol{M}_{\text{new}} + \boldsymbol{m}_{\text{new}}, \boldsymbol{\Sigma}_{\text{new}})$$

⬅ **Compatible Function Approximation**

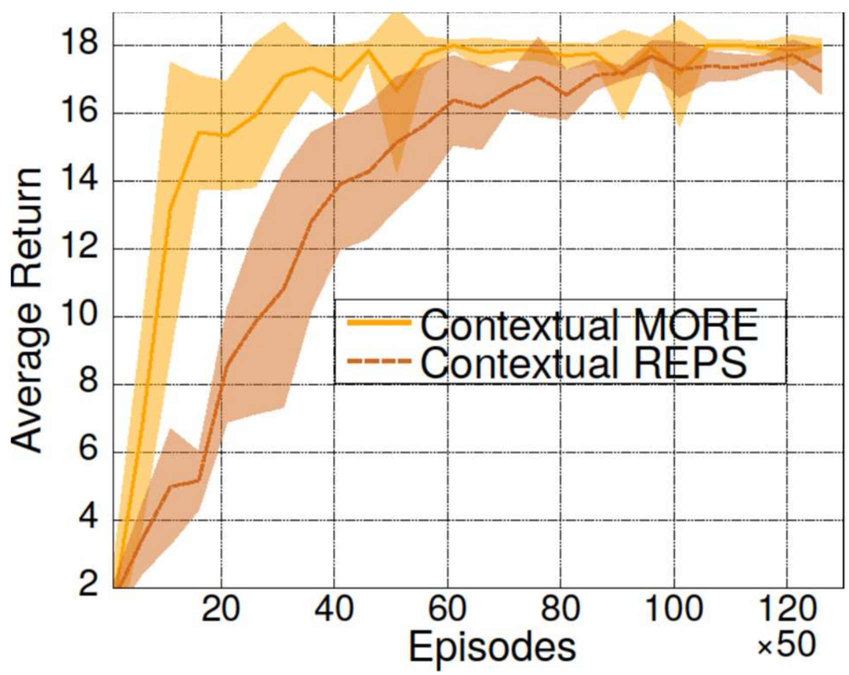
Adaptation of Skills: Table Tennis

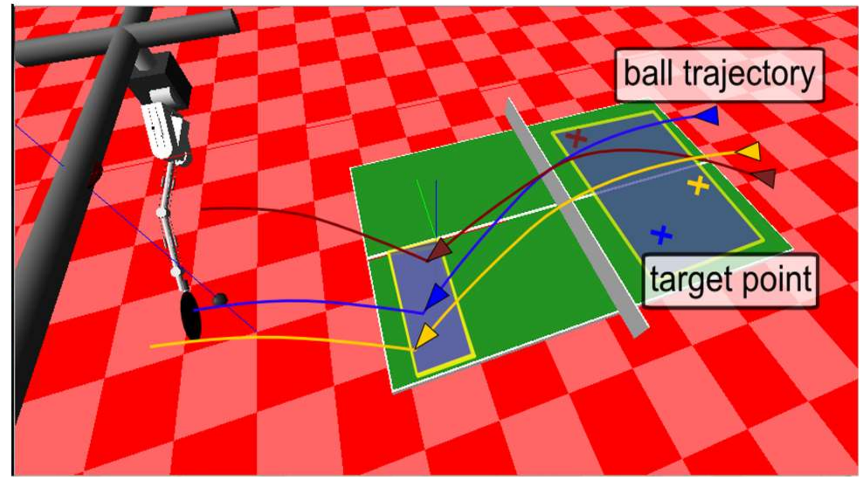
Contextual Stochastic Search:

- Context: Initial ball velocity

Reward:

- Hit ball
- Ball impacts at target position



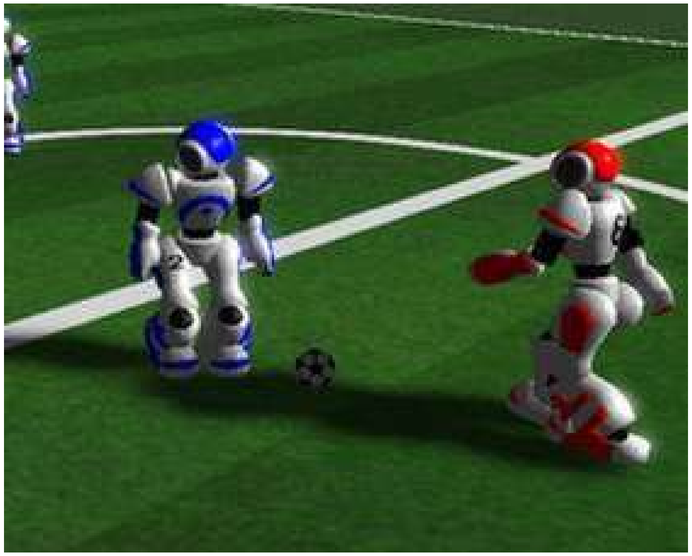


Skills Improvement:

- ✓ Hot-start with imitation
- ✓ Continuous-valued decision making
- ✓ Low number of samples
- ✓ Adaptation

Skill Improvement: Controlled Kick

- **Task**
  - Develop a **kick with controlled kicking distance**
  - From 10 different positions in the soccer field (with distances ranging from 3m to 12m), kick the ball so that it stops in the center of the field
- **Classical approach**
  - Optimize for each distance
- **Contextual approach**
  - Optimize for all distances in a single process
  - Use all data to improve performance
  - Generalize for unknown contexts



Skill Improvement: Controlled Kick

Abbas Abdolmaleki et al. Learning a Humanoid Kick With Controlled Distance. RoboCup 2016: Robot World Cup XX, Springer, July 2016

Presentation outline

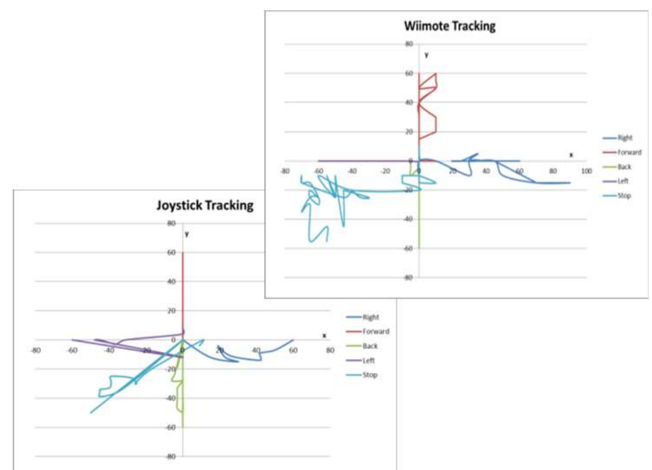
- Motivation
  - Challenges for Robotics Learning
- Q-Batch update rule
- Multi-context optimization
- **User profiles and Adapted interfaces**
- Multiagent Learning
- Robot motion planning
- Conclusion

User profiles and Adapted interfaces

- **Users** of Intelligent Wheelchairs **have very different skills**
- **command interface** provided for each user **should be adapted to his/her capabilities**
  - **User profiling** provides relevant information
  - **automatically generate command language** adapted to the user for driving the IW

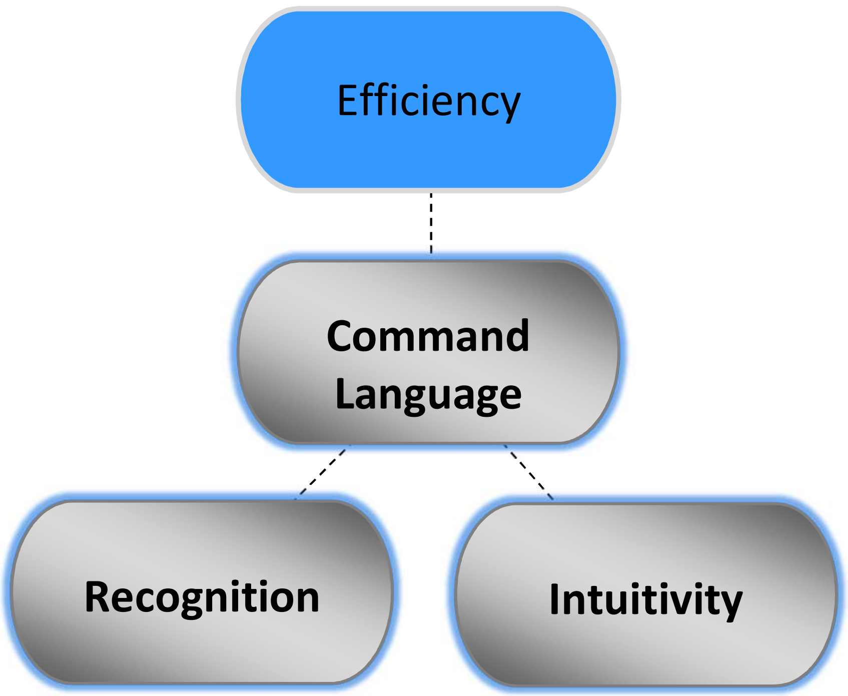
User profiles and Adapted interfaces

- User Profiling Experiments
  - 11 cerebral palsy users
  - Level IV (27.3%) and V (72.7%) GMFM
  - Voice Inputs
    - “Go”, “Front”, “Forward”, “Back”, “Right”, “Left”, “Turn”, “Spin” and “Stop”
  - Joystick and the Head Movements



User profiles and Adapted interfaces

- Command Language

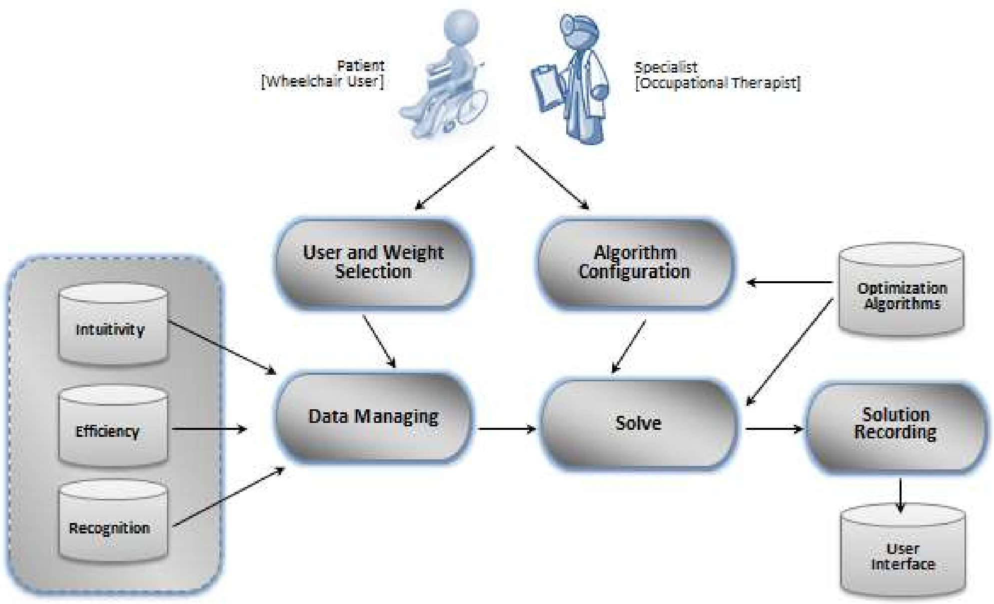


User profiles and Adapted interfaces

- Command Language

Maximizes the function composed by the total time efficiency, total recognition and intuitiveness

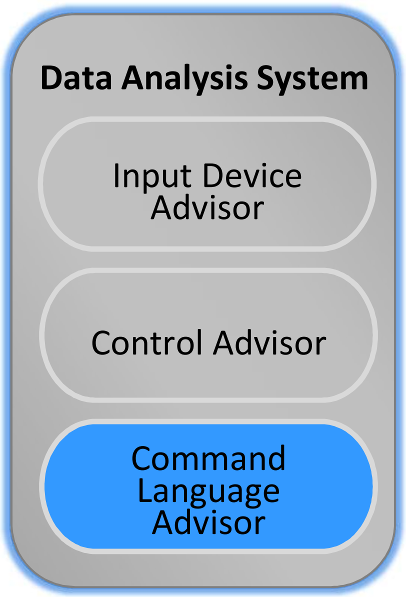
$$\arg\max_{T_{eff}, T_{reg}, T_{\text{int}}} (\alpha T_{eff} + \beta T_{reg} + \gamma T_{\text{int}})$$




```
(w_rec, w_time, w_intu) = weights; evaluation ← 0
for ncom = 1 to NC do
   recVal ← 1; timeVal ← 0;  intuVal ← 1
   for nseq = 1 to NS do
      inpDev ← inputDevice(solution[ncom][nseq])
      inp ← input(newSolution[ncom][nseq])
      if inpDev = NULL then break
      else
         recVal ← recVal * rec[inpDev][inp]
         timeVal ← timeVal + time[inpDev][inp]
         intuVal ← intuVal * intu[ncom][inpDev][inp]
      endif
   endfor
   evalComm ← w_rec* recVal + w_time*1/(timeVal+1)
                  + w_intu*intuVal
   evaluation ← evaluation  +  evalComm
endfor
return evaluation
```

User profiles and Adapted interfaces

- Command Language Advisor



Mean of DAS evaluation higher than mean of evaluation of the command language recommended by specialist (p value = 0.002)

		Command Language for Patients				
Patient	Evaluation	Forward	Left	Right	Back	Stop
P1						
Specialist	4.53	wiimote	joystick	joystick	joystick	joystick
IDAS	4.57	joystick	joystick	joystick	joystick	joystick
P2						
Specialist	4.18	joystick	joystick	joystick	joystick	voice ("stop")
IDAS	4.85	joystick	joystick	joystick	joystick	voice ("go")
P3						
Specialist	3.33	voice ("forward")	wiimote	wiimote	joystick	voice ("stop")
IDAS	4.51	wiimote	wiimote	wiimote	wiimote	voice ("go")
P4						
Specialist	4.50	voice ("forward")	joystick	joystick	joystick	voice ("stop")
IDAS	4.60	joystick	joystick	joystick	joystick	voice ("stop")
P5						
Specialist	4.14	voice ("front")	wiimote	wiimote	joystick	voice ("stop")
IDAS	4.40	wiimote	wiimote	voice ("turn")	joystick	voice ("stop")
P6						
Specialist	4.13	wiimote	joystick	joystick	joystick	joystick
IDAS	4.38	wiimote	wiimote	wiimote	wiimote	wiimote
P7						
Specialist	4.49	voice ("front")	joystick	joystick	joystick	voice ("stop")
IDAS	4.60	joystick	joystick	joystick	voice ("back")	voice ("stop")
P8						
Specialist	3.51	wiimote	joystick	joystick	joystick	joystick
IDAS	4.20	wiimote	wiimote	wiimote	wiimote	wiimote
P9						
Specialist	3.70	voice ("forward")	wiimote	wiimote	joystick	voice ("stop")
IDAS	4.75	joystick	joystick	joystick	joystick	joystick
P10						
Specialist	4.11	voice ("forward")	voice ("left")	voice ("right")	voice ("turn")	voice ("stop")
IDAS	4.80	joystick	joystick	voice ("turn")	joystick	voice ("go")
P11						
Specialist	4.29	joystick	wiimote	wiimote	joystick	joystick
IDAS	4.30	wiimote	wiimote	wiimote	wiimote	wiimote

Brígida Mónica Faria, el al. A Methodology for Creating an Adapted Command Language for Driving an Intelligent Wheelchair. Journal of Intelligent & Robotic Systems, vol. 80, no. 3, December 2015

Presentation outline

- Presentation
- Motivation
  - Robotics Learning Problems
- Some solutions
  - Gesture recognition
  - Q-Batch update rule
  - Multi-context optimization
  - User profiles and Adapted interfaces
  - **Multiagent Learning**
- Conclusion

Multiagent Learning

- Learning Coordination among several agents
- Multiagent reward based learning challenges
  - Non static environment
  - Complexity exponential to number of agents

- Double Deep Q Networks used for multiagent paradigm

Multiagent Learning

- Learning Coordination among several agents
- Multiagent reward based learning challenges
  - Non static environment
  - Complexity exponential to number of agents

- Double Deep Q Networks used for multiagent paradigm

  $\Rightarrow$ **Multiagent Double Deep Q-Networks**

Multiagent Learning

- Joint-Action Multiagent Double DQN

Input: Learning rate η, mini-batch size k, discount factor γ, network update period τ, replay memory $\mathcal{D}$ with capacity N, action-value function Q with random weights θ

for iteration = $1, M$ **do**
    for agent $p = 1, P$ **do**
        Sample state $s_{1,p}$
    end for
    Compute ϕ_1
    for step $t = 1, T$ **do**
        for agent $p = 1, P$ **do**
            Select random action $a_{t,p}$ with probability ϵ, otherwise best action $a_{t,p} = \max_a Q^*(\phi(s_t), a; \theta)$
            Execute $a_{t,p}$
            Observe image $s_{t+1,p}$ and reward r_t
        end for
        Compute ϕ_{t+1}
        Store transition $(\phi_t, a_{t,1}, \ldots, a_{t,p}, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Sample random mini-batch of k transitions $(\phi_j, a_{j,1}, \ldots, a_{j,b}, r_t, \phi_{j+1})$ from $\mathcal{D}$
        for transition $i = 1, k$ **do**
            Update $\theta \leftarrow \theta + \eta \nabla_{\theta_i} L_i(\theta_i)$
        end for
        Update network weights $\theta_{target} \leftarrow \theta$ every τ time-steps
    end for
end for

Multiagent Learning

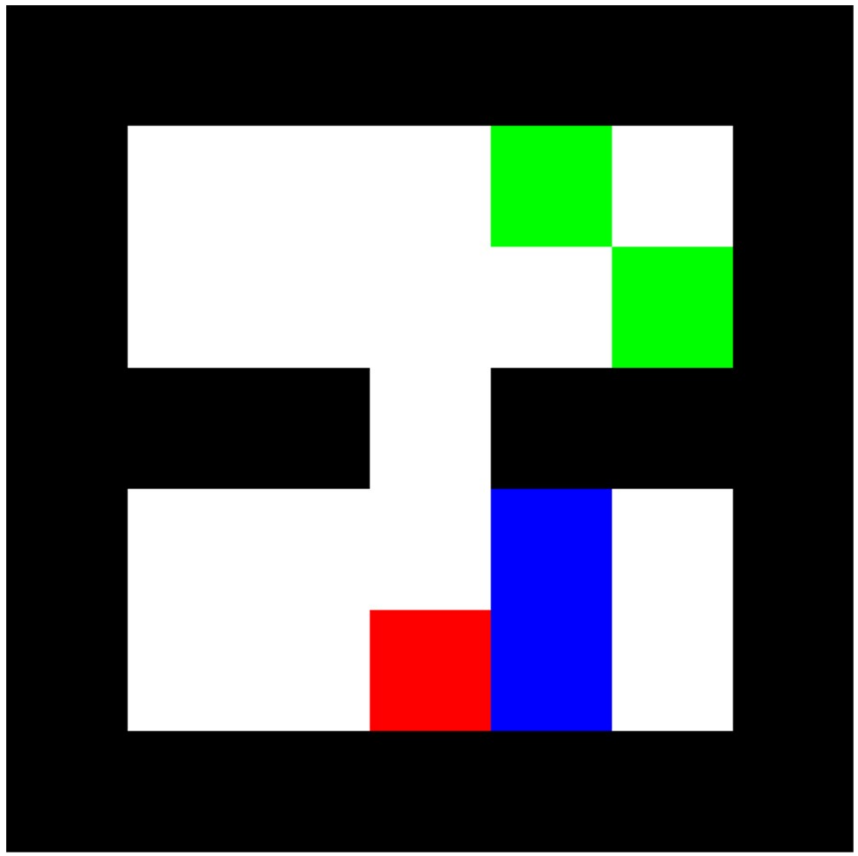
- Independent Learners Multiagent Double DQN

Input: Learning rate η, mini-batch size k, discount factor γ, network update period τ, replay memory $\mathcal{D}$ with capacity N, action-value function Q with random weights θ

for iteration = $1, M$ **do**
    for agent $p = 1, P$ **do**
        Sample state $s_{1,p}$ and compute $\phi_{1,p}$
    end for
    for step $t = 1, T$ **do**
        for agent $p = 1, P$ **do**
            Select random action $a_{t,p}$ with probability ϵ, otherwise best action $a_{t,p} = \max_a Q^*(\phi(s_t), a; \theta)$
            Execute $a_{t,p}$
            Observe image $s_{t+1,p}$ and reward r_t
            Compute $\phi_{t+1,p}$
            Store transition $(\phi_{t,p}, a_{t,p}, r_t, \phi_{t+1,p})$ in $\mathcal{D}$
        end for
        Sample random mini-batch of k transitions $(\phi_{j,b}, a_{j,b}, r_t, \phi_{j+1,b})$ from $\mathcal{D}$
        for transition $i = 1, k$ **do**
            Update $\theta \leftarrow \theta + \eta \nabla_{\theta_i} L_i(\theta_i)$
        end for
        Update network weights $\theta_{target} \leftarrow \theta$ every τ time-steps
    end for
end for

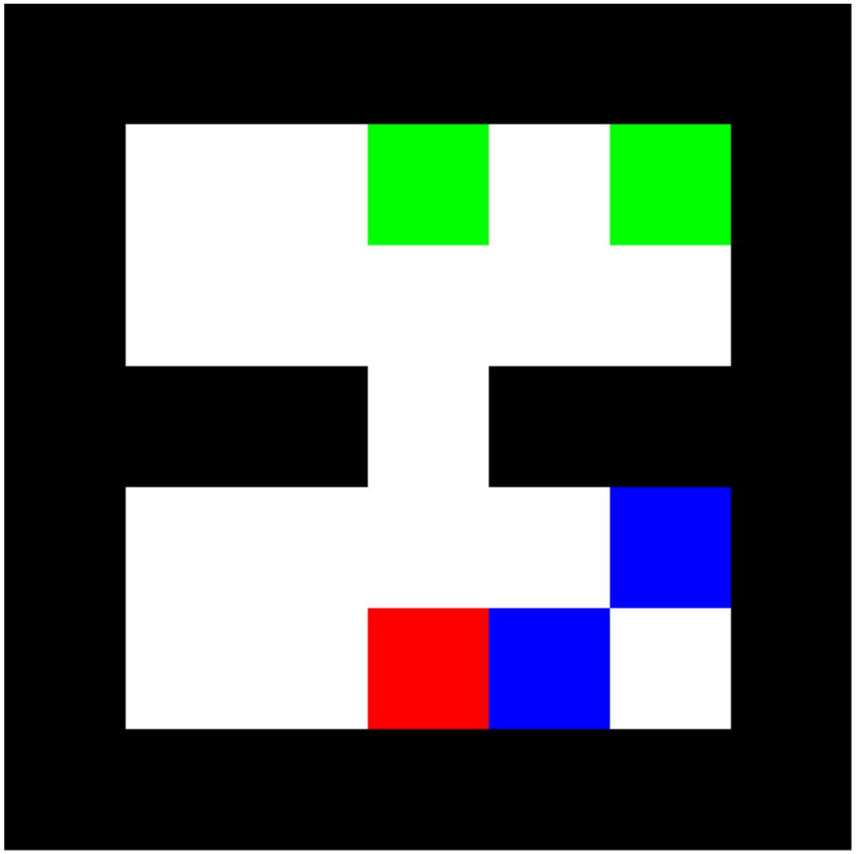
Multiagent Learning

- Foraging task: 2 agents; 2 berries
- 10k iterations



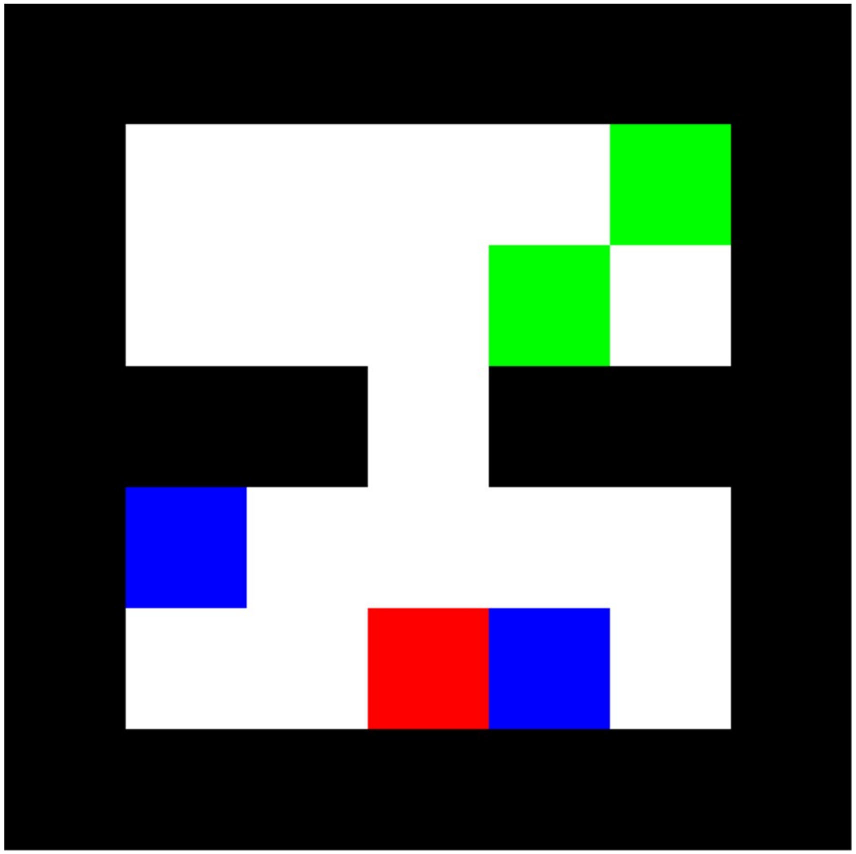
Multiagent Learning

- Foraging task: 2 agents; 2 berries
- 100k iterations



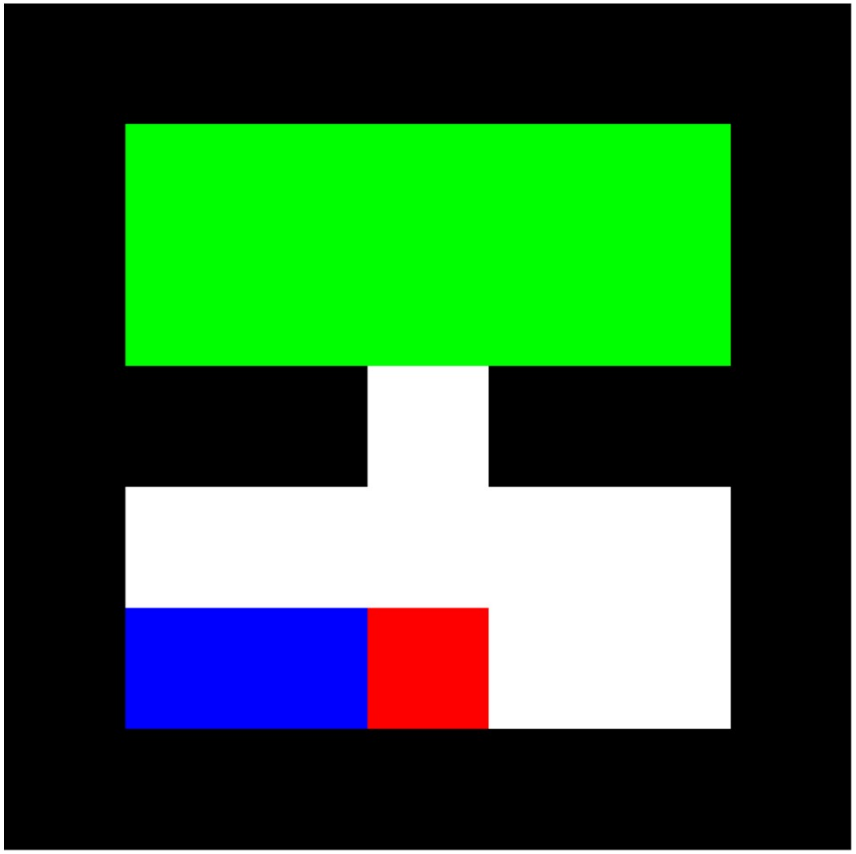
Multiagent Learning

- Foraging task: 2 agents; 2 berries
- 200k iterations



Multiagent Learning

- Foraging task: 2 agents; 10 berries
- Transfer Learning



Multiagent Coordination

- Formation specification

Presentation outline

- Presentation
- Motivation
  - Robotics Learning Problems
- Some solutions
  - Gesture recognition
  - Q-Batch update rule
  - Multi-context optimization
  - User profiles and Adapted interfaces
  - Multiagent Learning
- **Conclusion**

Conclusion

- Broad range of learning techniques applied to different areas of Robotics:
  - Perception
  - Behavior development
    - Value based
    - Contextual policy search
  - Adapting Human-Robot Interfaces
  - Coordination of Robot teams
- Learning can be applied to Robotics, but:
  - Data should be used as efficiently as possible
  - Take advantage of data structure
  - Combine different approaches, if needed
  - Use simulation in the first learning steps

I thank all people that contributed to these results, namely **Abbas Abdolmaleki**, **Brígida Mónica Faria**, **David Simões**, **João Cunha** and **Gi Hyun Lim** and also all people from **CAMBADA**, **EuRoC** and **FC Portugal** projects

Thank you for your attention.
Questions?

Learning Tasks in Robotics: Problems and Solutions

Nuno Lau (nunolau@ua.pt)

IEETA – Institute of Electronics and Informatics Engineering of Aveiro
DETI / Universidade de Aveiro