



SAPIENZA
UNIVERSITÀ DI ROMA

# Multi-Robot Search for a Moving Target: Integrating World Modeling, Task Assignment and Context

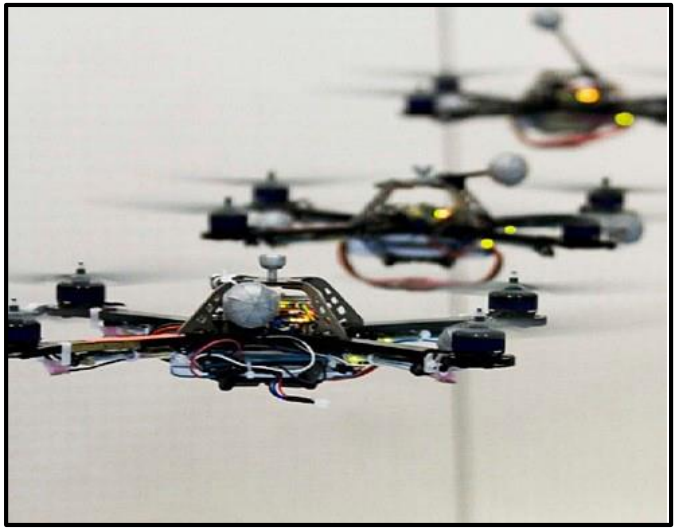
Francesco Riccio, Emanuele Borzi, Guglielmo Gemignani, Daniele Nardi

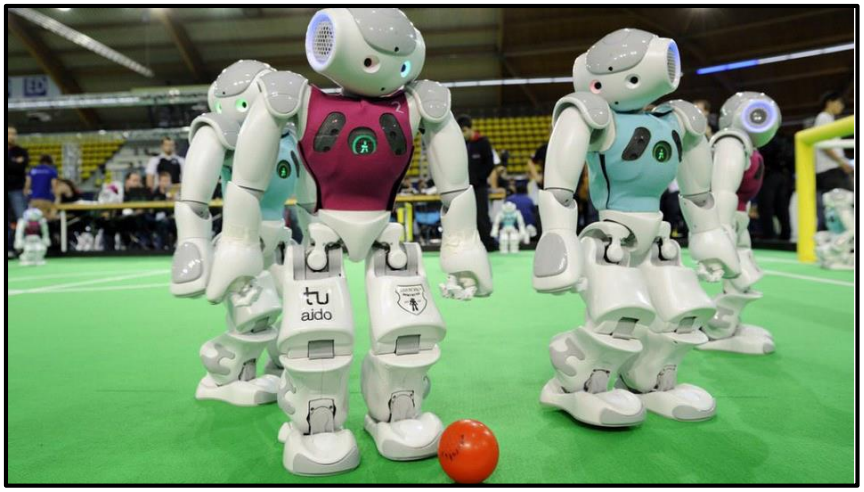
Department of Computer, Control, and Management Engineering

Antonio Ruberti

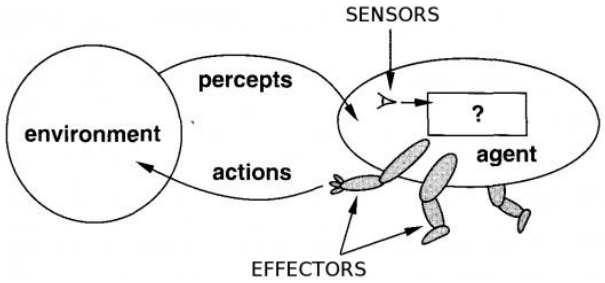
email: riccio@dis.uniroma1.it

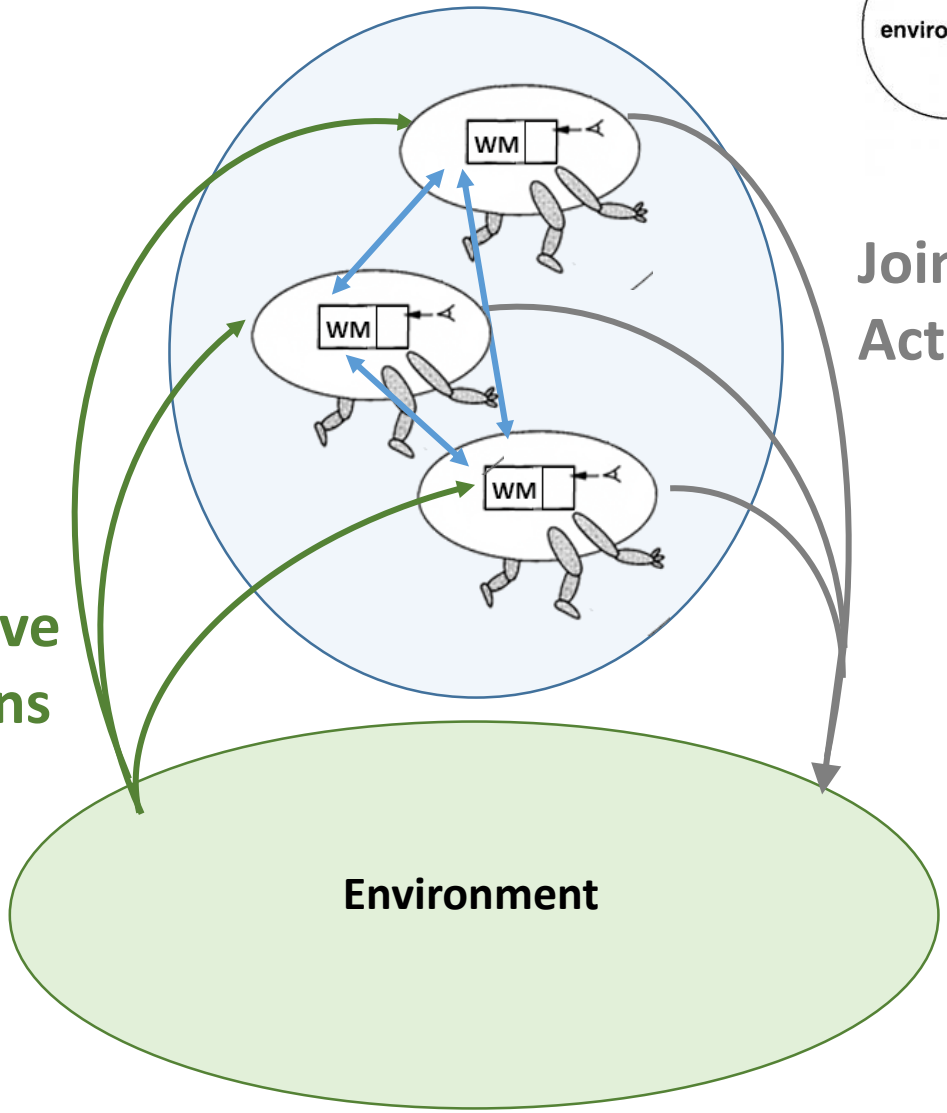
# Multi-Robot Systems





# Multi-Robot Systems



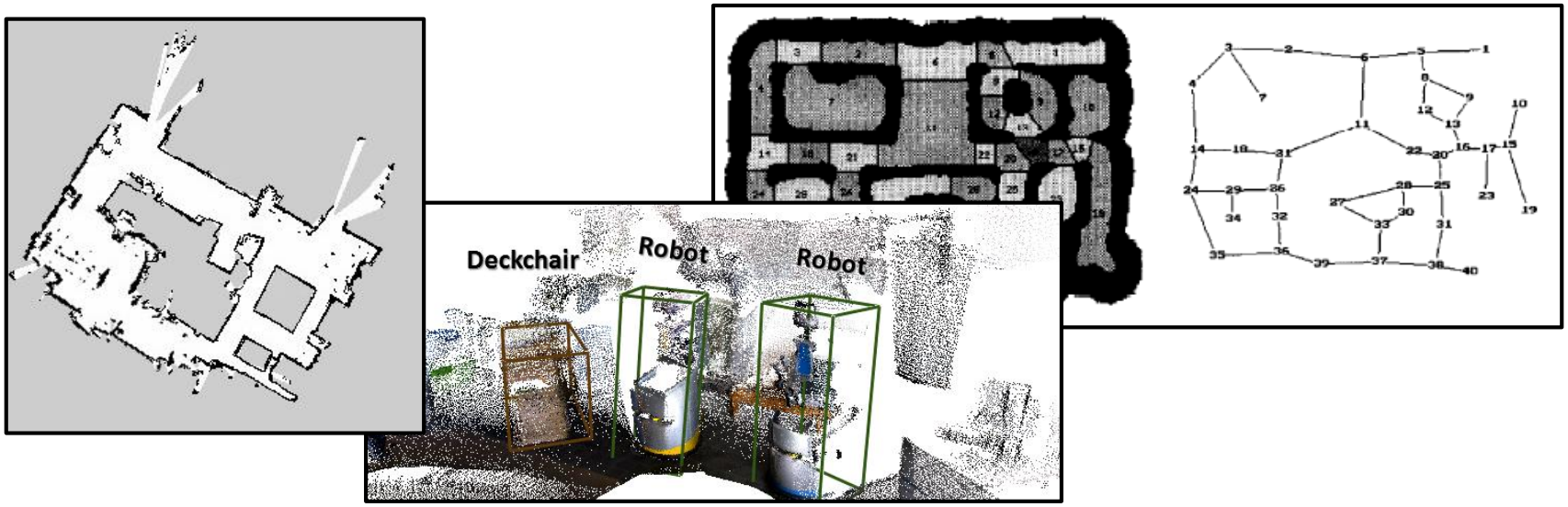



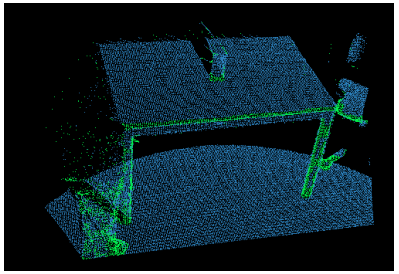

**Joint Actions**

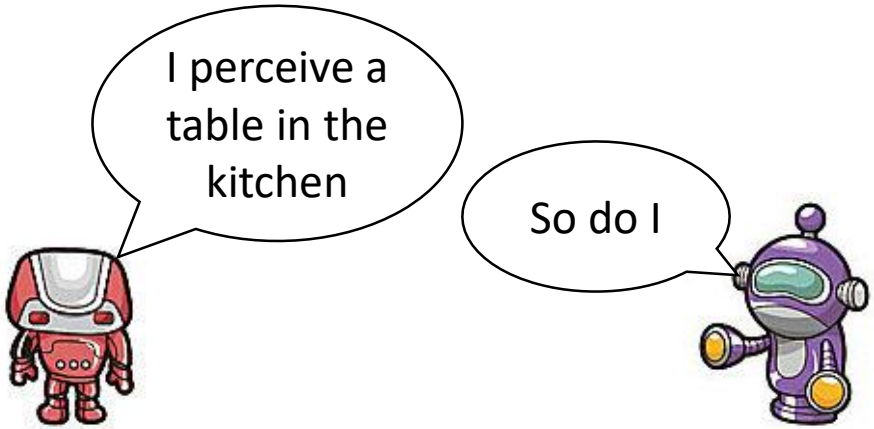
**Cooperative Perceptions**

# World modeling

**modeling**

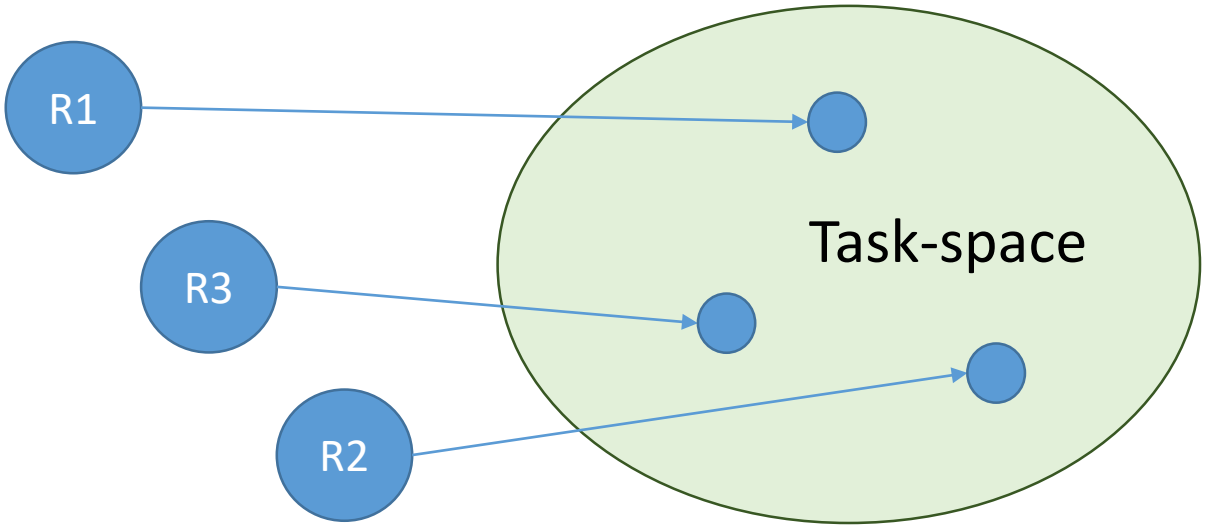


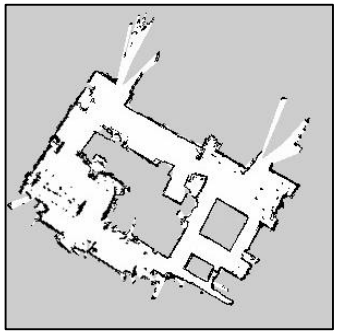



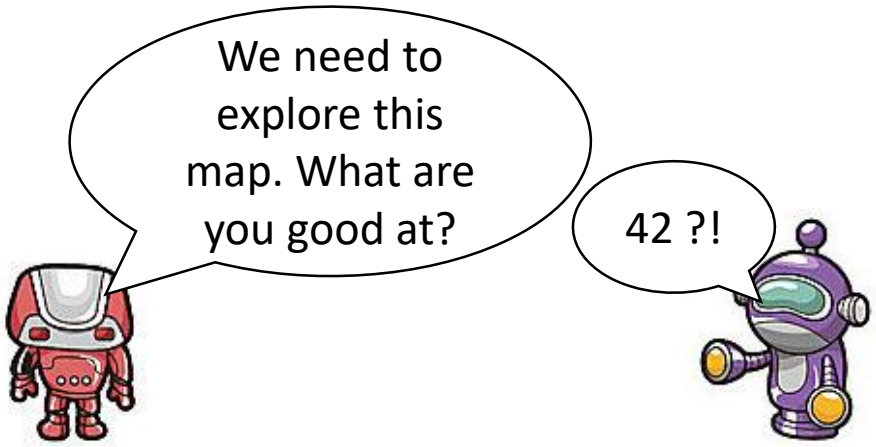


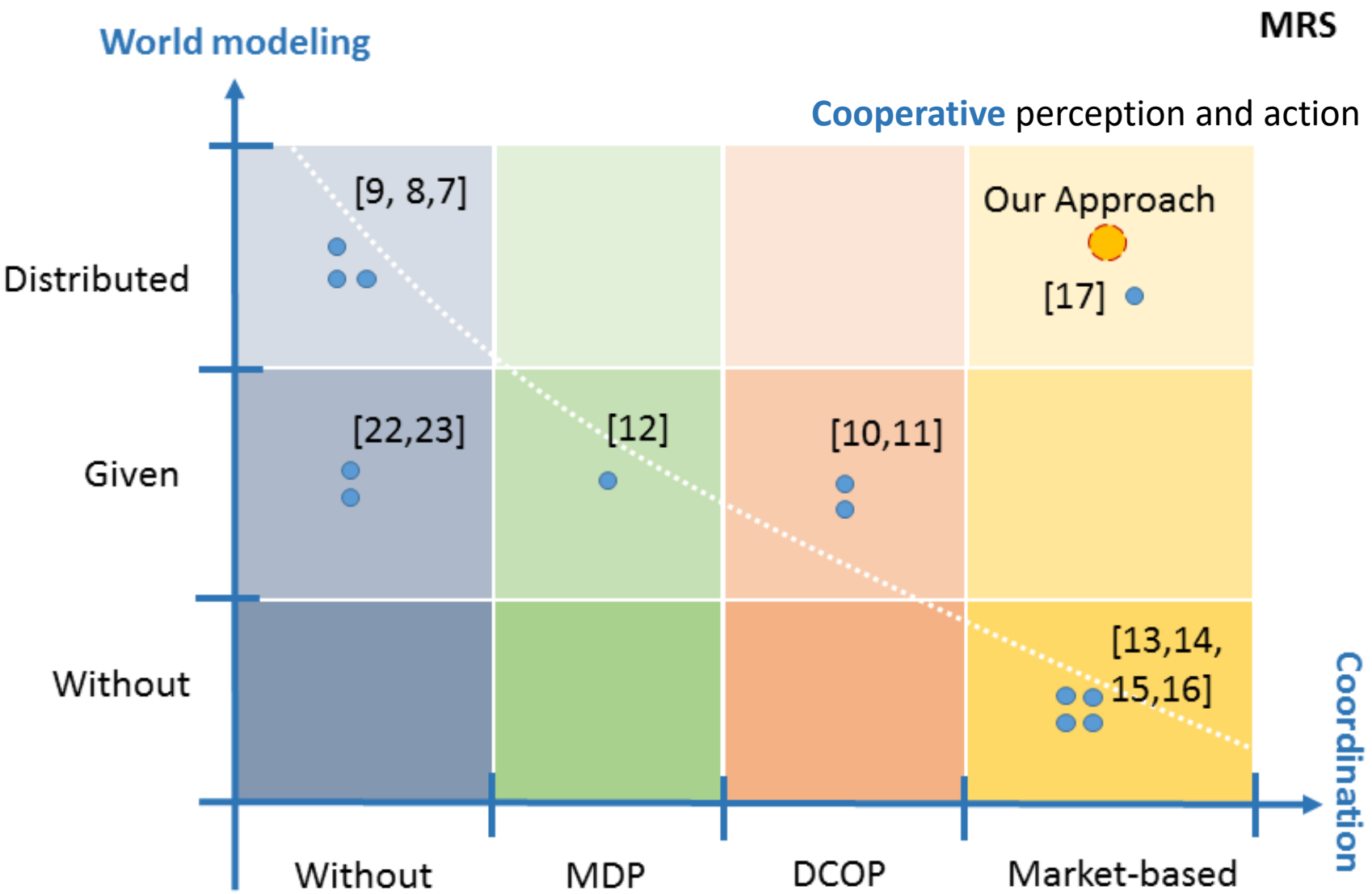

# Task Assignment

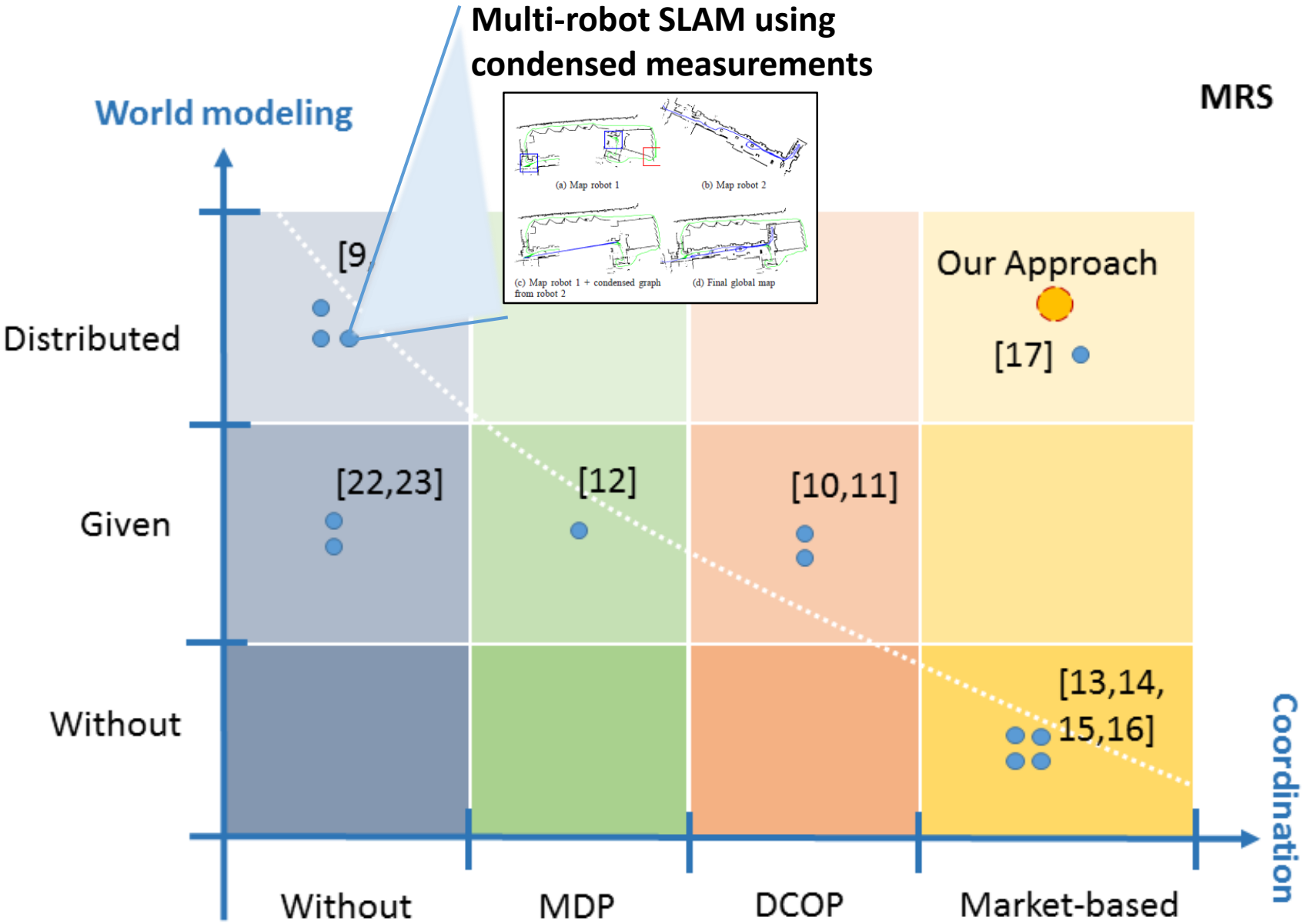
**coordination**

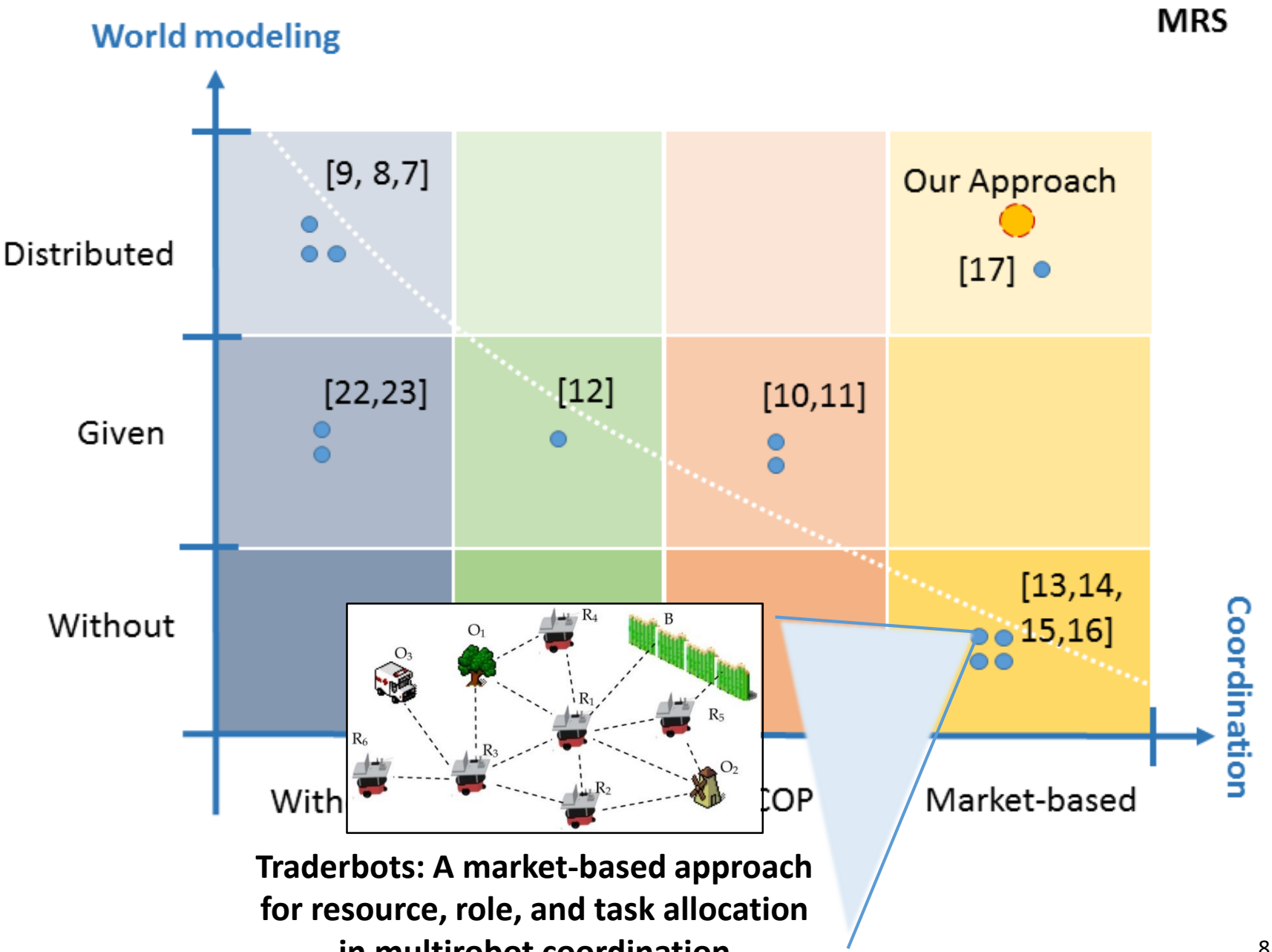


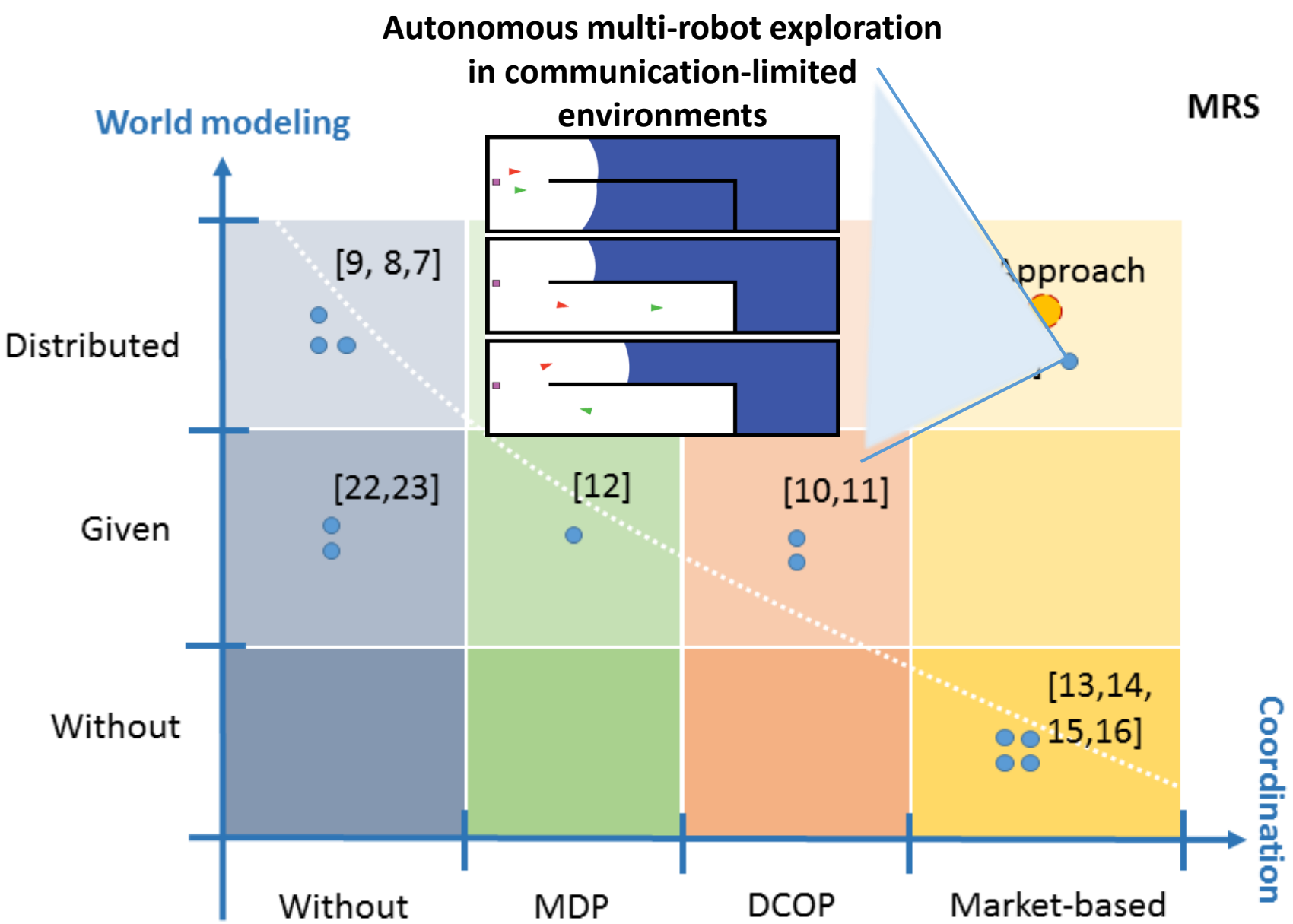



MRS
World modeling
Cooperative perception and action
Distributed
Given
Without
[9, 8,7]
Our Approach
[17]
[22,23]
[12]
[10,11]
[13,14,
15,16]
Coordination
Without
MDP
DCOP
Market-based


Multi-robot SLAM using condensed measurements
MRS
World modeling
(a) Map robot 1
(b) Map robot 2
(c) Map robot 1 + condensed graph from robot 2
(d) Final global map
[9
Our Approach
[17]
Distributed
[22,23]
[12]
[10,11]
Given
[13,14, 15,16]
Without
Coordination
Without
MDP
DCOP
Market-based




**Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination**


Autonomous multi-robot exploration in communication-limited environments
MRS
World modeling
Distributed
Given
Without
[9, 8,7]
[22,23]
[12]
[10,11]
[13,14, 15,16]
pproach
Without
MDP
DCOP
Market-based
Coordination

# Motivation and Application

Environment requires adaptive coordination

➡ **Context modeling**

Search for non-adversarial target (Hollinger et al. 2009)

# Distributed Task Assignment

**utility estimations**

Each robot *i:*

- **estimates**

$$UEV_i(t) = [b_{(i,1)}(t), \ldots, b_{(i,m)}(t)]$$

- **collects**

$$UEM_i(t) = [UEV_1\ (t), \ldots, UEV_n\ (t)]^T$$

- **evaluates**

$$< r_i, \tau_j > = \arg\max_i utility(\tau_j)$$

# Search for a Non-adversarial Target

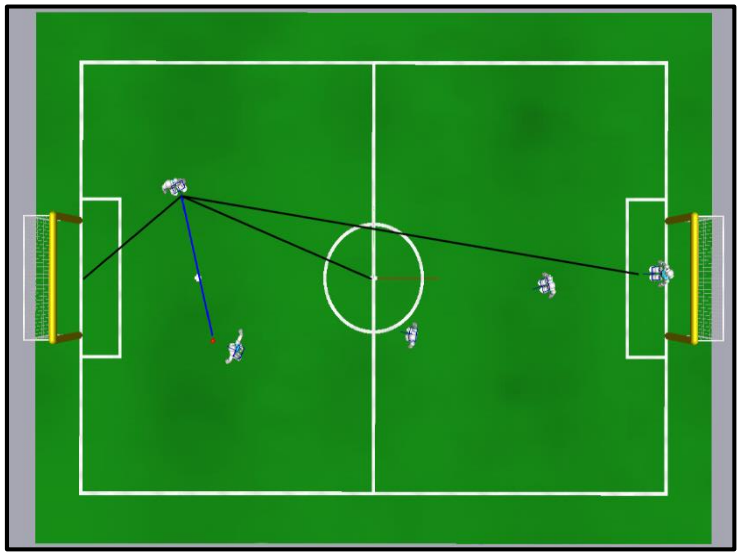
## Task Assignment

$$UEV_i(t) = [b_{(i,1)}(t), \ldots, b_{(i,m)}(t)]$$

$$UEM_i(t) = [UEV_1\ (t), \ldots, UEV_n\ (t)]^T$$

$$< r_i, \tau_j > = \arg\max_i utility(\tau_j)$$

| | T0 | T1 | T2 | T3 |
|---|---|---|---|---|
| R0 | 11939 | 0 1104 | 571 | 872 225 |
| R1 | 11940 | 0 393 | 1312 | 755 691 |
| R2 | 11941 | 0 742 | 416 | 1273 114 |
| R4 | 11942 | 0 284 | 869 | 650 784 |



# Distributed World Modeling

**suitable representation of the environment**

**Local model update**

Interpret environmental changes through multi-casted **events**

$$\overline{LM_j}(t) = \varphi(e(t), LM_j(t))$$

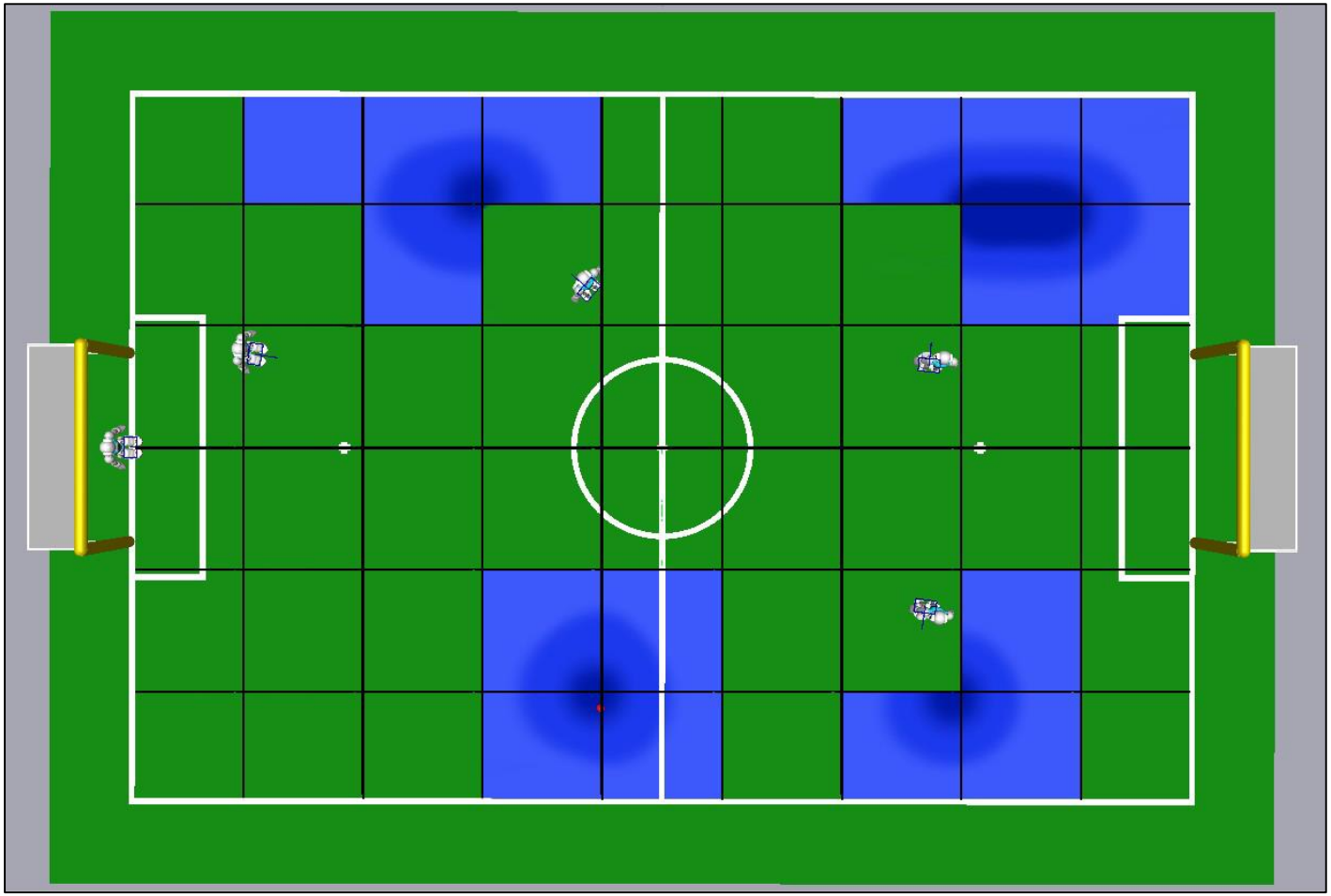
**Distributed Modeling**

Given $\boldsymbol{n}$ robot local models $\boldsymbol{\overline{LM_j}(t)}$, we **reconstruct** $\boldsymbol{DWM_j(t)}$

$$DWM_j(t) = f\left(\left\{\overline{LM_j}\right\}_{j=1}^{n}, t\right)$$

# Search for a Non-adversarial Target

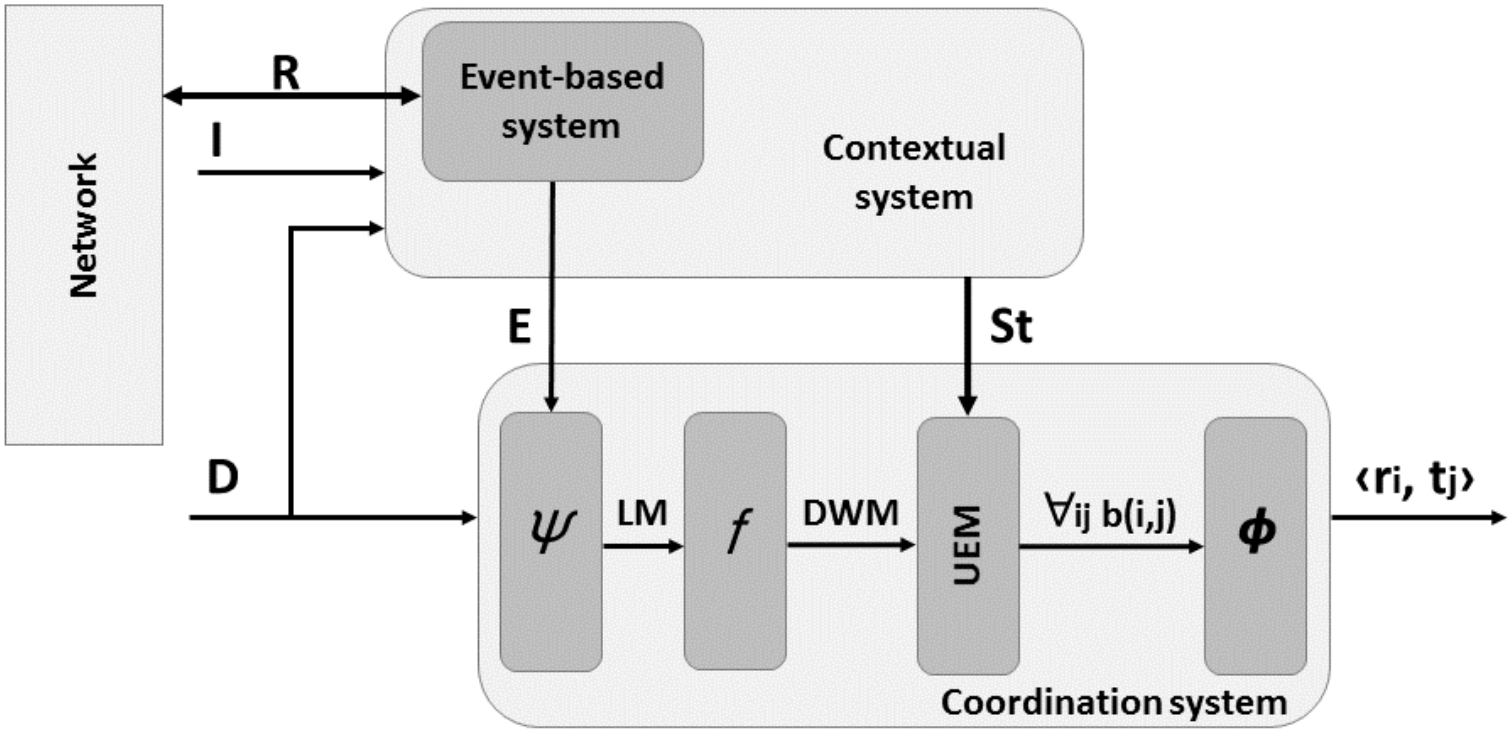
## World Representation



# Context System

**Context System** outputs the best **strategy** in accordance with the current world state

$$CS: [\, D \times I \,] \rightarrow St$$



# Algorithm

1. $S_t \leftarrow updateContextKnowledge(I, D)$

2. $\{TS_j\}_{j=1}^{N} \leftarrow getTeamState(R)$
3. $\{\overline{LM_j}\}_{j=1}^{N} \leftarrow updateLMs(\{TS_j\}_{j=1}^{N})$

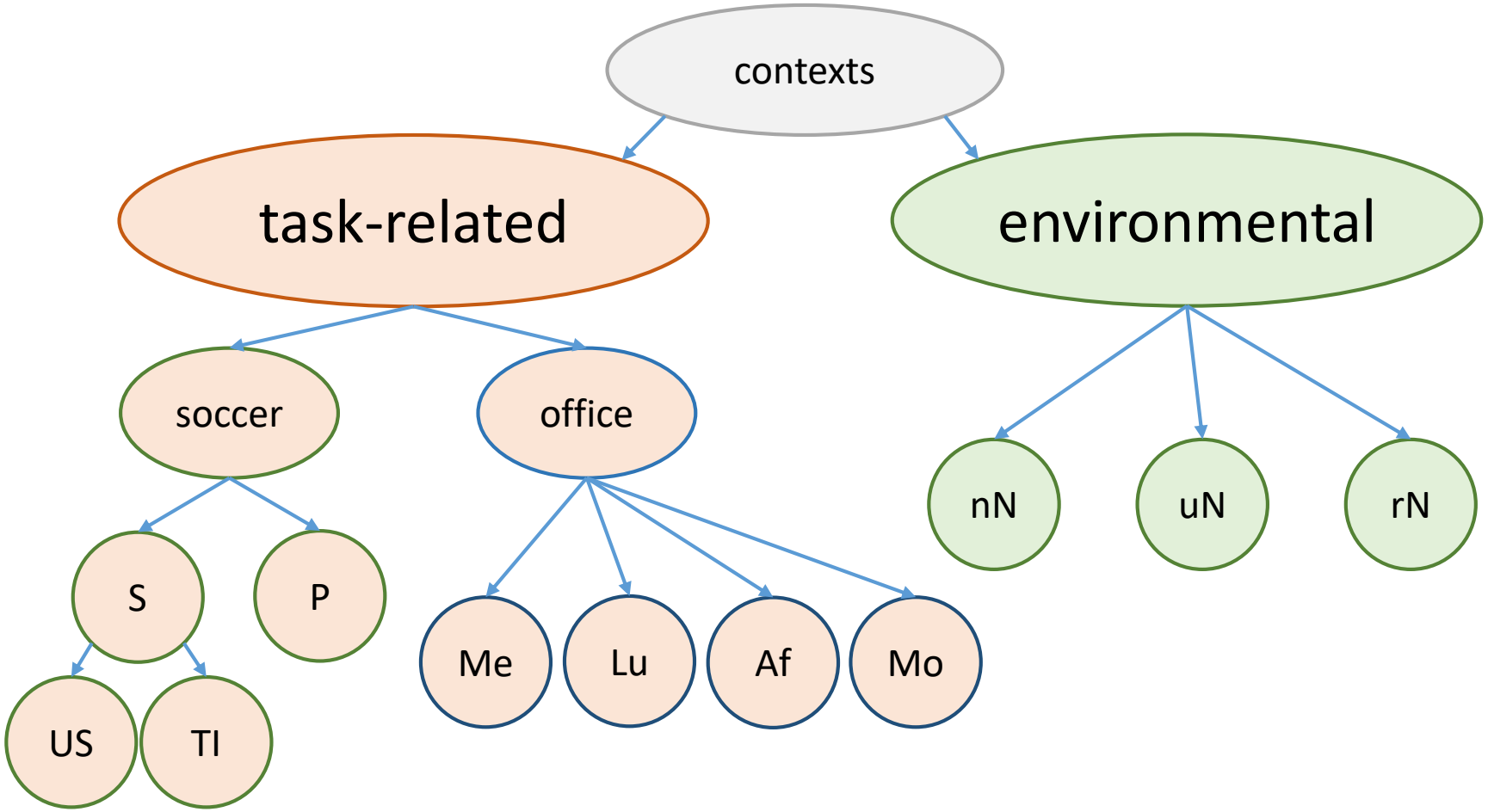
4. $DWM_j \leftarrow \boldsymbol{f}(\{\overline{LM_j}\}_{j=1}^{N})$

5. $UEV_i \leftarrow computeUtilityVector(DWM_j, S_t)$

6. $send\ (UEV_i)$
7. $\{UEV_j\}_{j=1}^{N} \leftarrow getUtilities(R)$

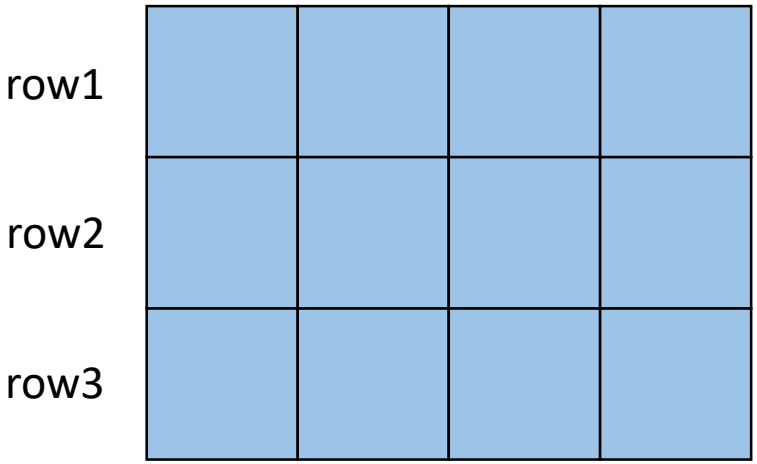
8. $\overline{UEM_i} \leftarrow computeUtilityMatrix(\{UEV_j\}_{j=1}^{N})$
9. $T_i \leftarrow mapping(\overline{UEM_i})$

# Context System



# Search for a Non-adversarial Target

## World Modeling



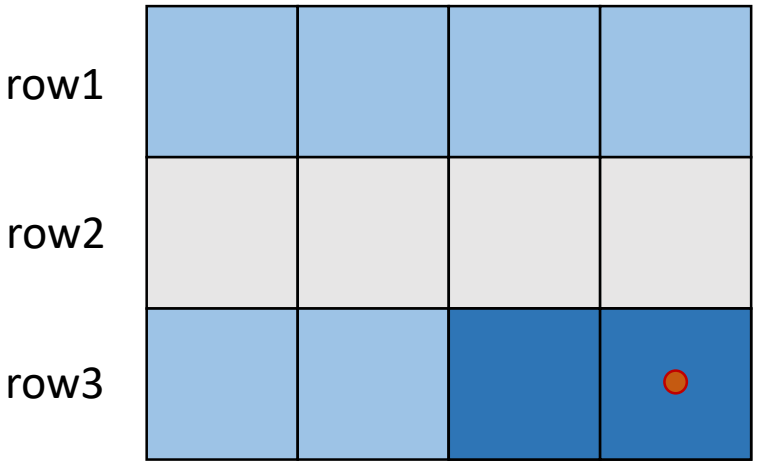

Reconstruction function ***f***:

$$DWM_i = \forall x, y \quad cell_i^{<x,y>}$$
$$= arg \min_{cell_i \in LM_j} \{score(cell_j^{<x,y>})\}$$

# Search for a Non-adversarial Target

## World Modeling

## Throw-in context



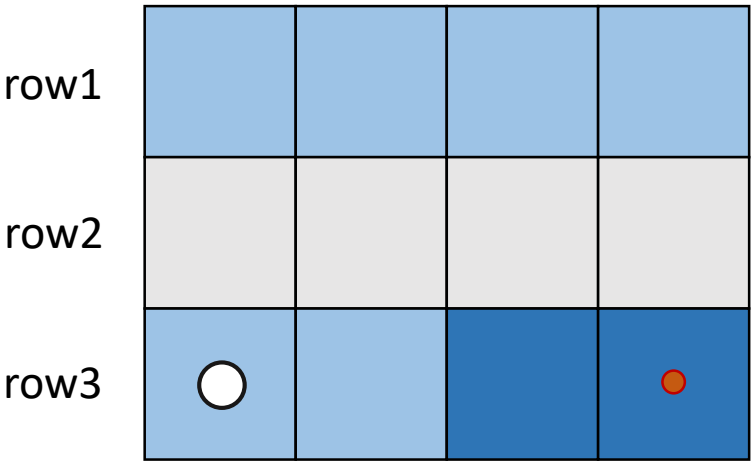

Reconstruction function ***f***:

$$DWM_i = \forall x, y \quad cell_i^{<x,y>}$$
$$= arg \min_{cell_i \in LM_j} \{score(cell_j^{<x,y>})\}$$

# Search for a Non-adversarial Target

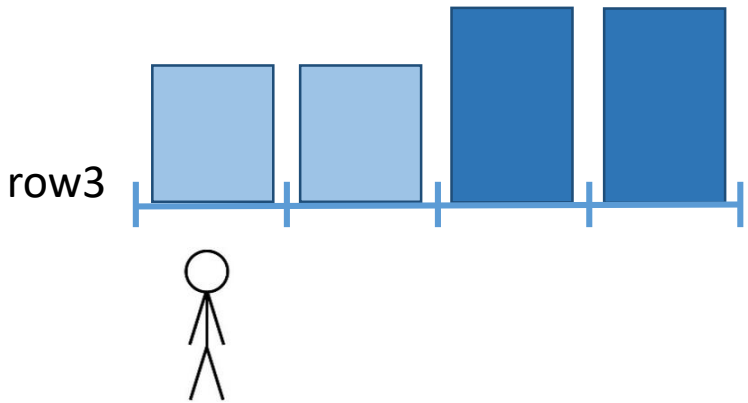
## World Modeling

## Throw-in context




Reconstruction function ***f***:

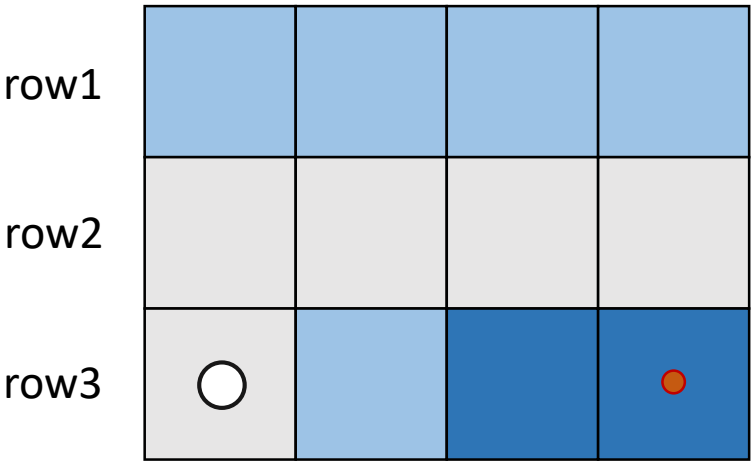
$$DWM_i = \forall x, y \quad cell_i^{<x,y>}$$
$$= arg \min_{cell_i \in LM_j} \{score(cell_j^{<x,y>})\}$$



# Search for a Non-adversarial Target

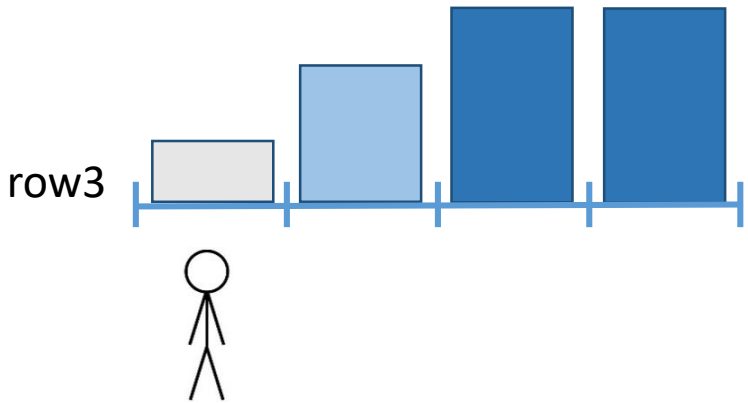
## World Modeling

## Throw-in context




Reconstruction function ***f***:

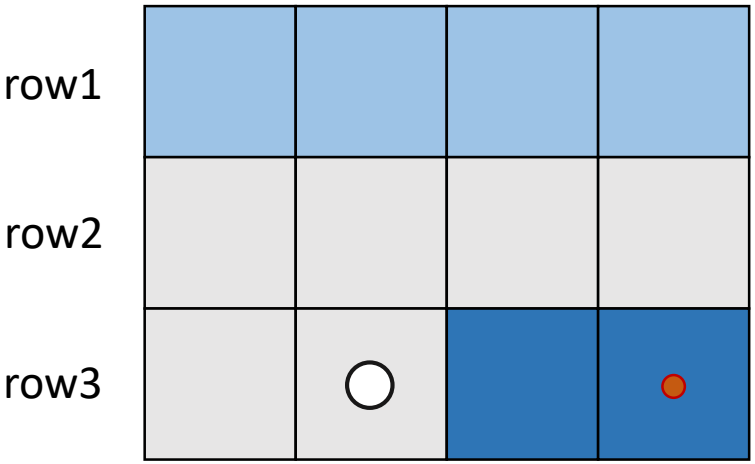
$$DWM_i = \forall x, y \quad cell_i^{<x,y>}$$
$$= arg \min_{cell_i \in LM_j} \{score(cell_j^{<x,y>})\}$$



# Search for a Non-adversarial Target

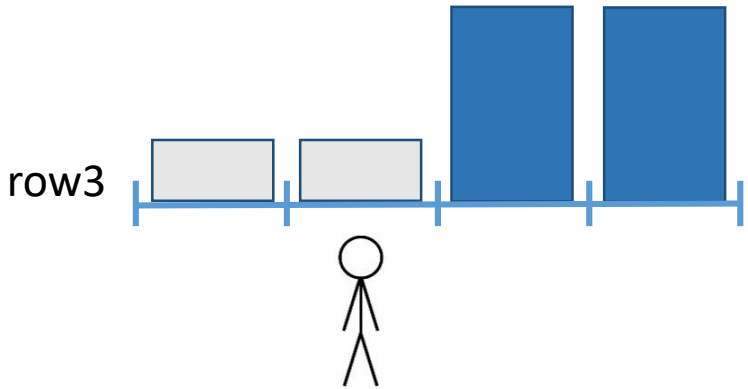
## World Modeling

## Throw-in context




Reconstruction function ***f***:

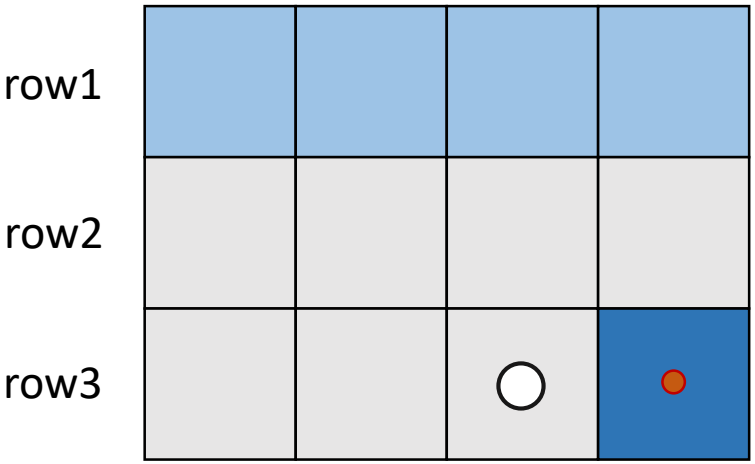
$$DWM_i = \forall x, y \quad cell_i^{<x,y>}$$
$$= arg \min_{cell_i \in LM_j} \{score(cell_j^{<x,y>})\}$$



# Search for a Non-adversarial Target

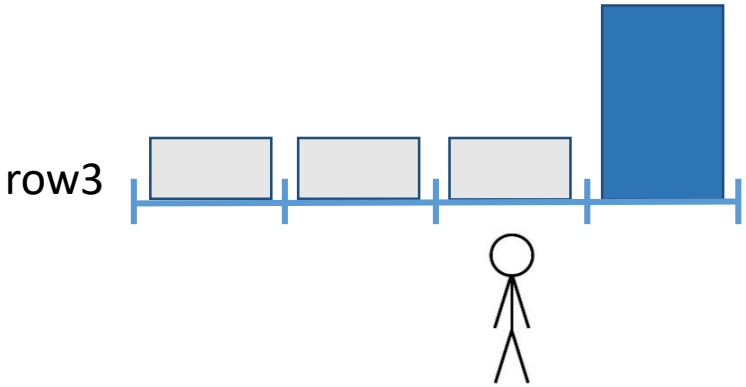
## World Modeling

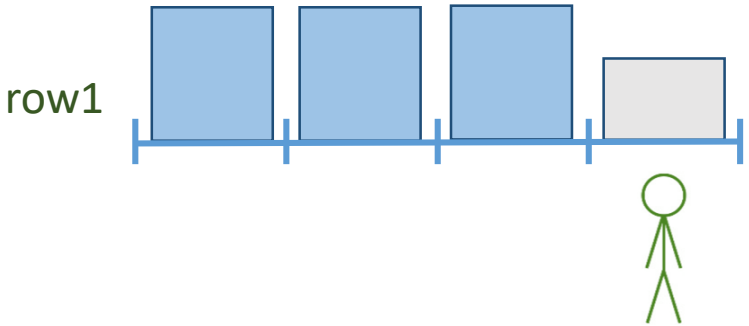
## Throw-in context




Reconstruction function ***f***:

$$DWM_i = \forall x, y \quad cell_i^{<x,y>}$$
$$= arg \min_{cell_i \in LM_j} \{score(cell_j^{<x,y>})\}$$

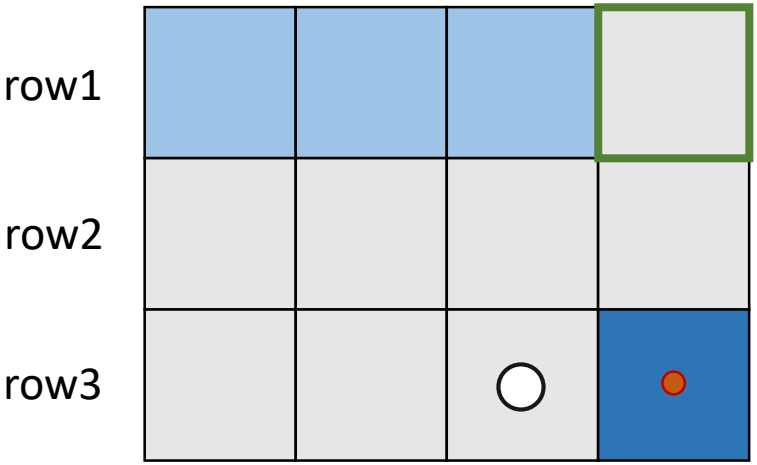


# Search for a Non-adversarial Target

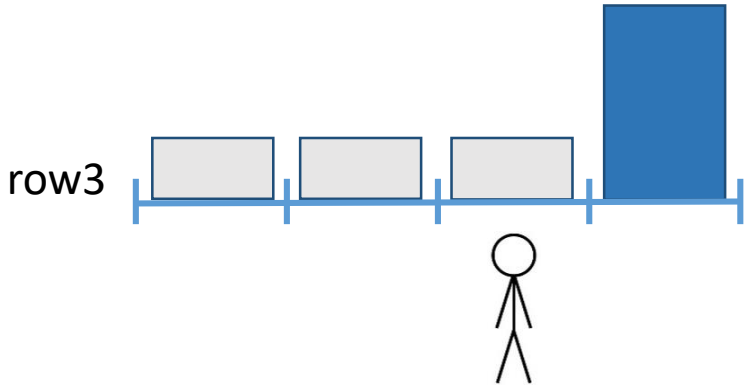
## World Modeling

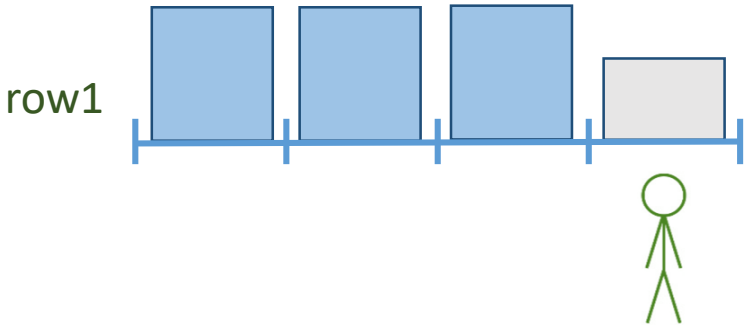
## Throw-in context




Reconstruction function ***f***:

$$DWM_i = \forall x, y \quad cell_i^{<x,y>}$$
$$= arg \min_{cell_i \in LM_j} \{score(cell_j^{<x,y>})\}$$

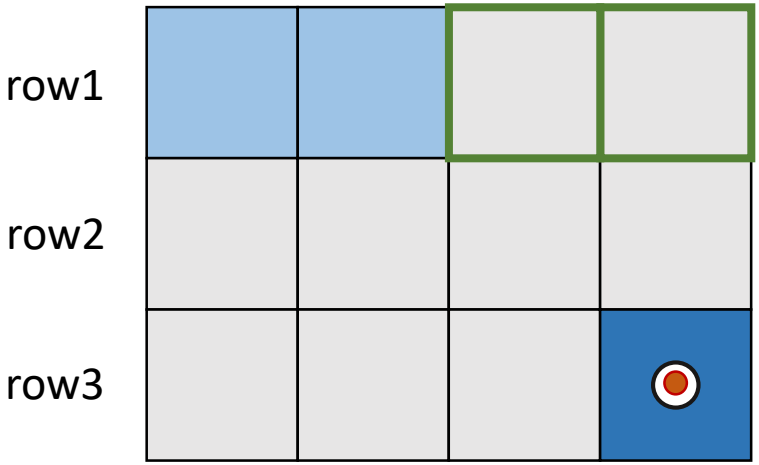


# Search for a Non-adversarial Target

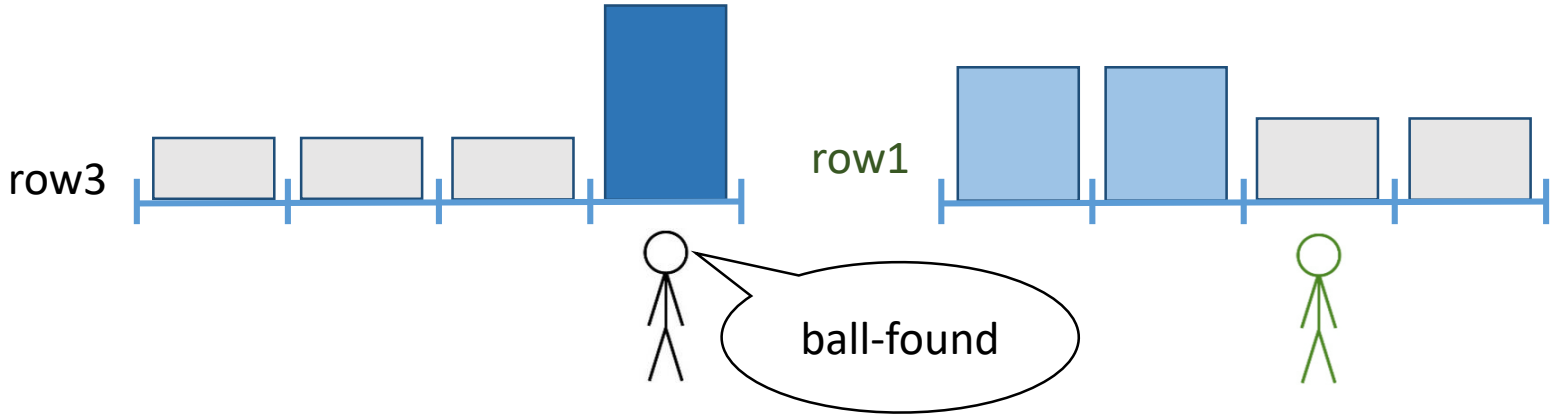
## World Modeling

## Throw-in context

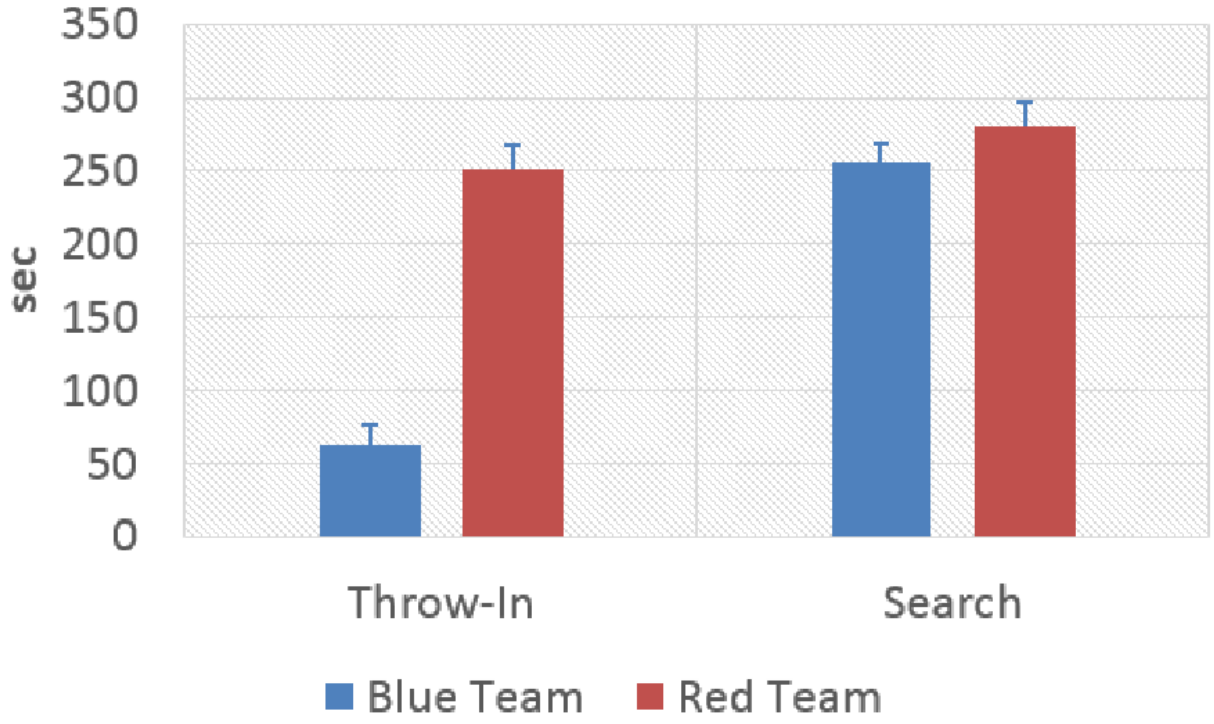



Reconstruction function ***f***:

$$DWM_i = \forall x, y \quad cell_i^{<x,y>}$$
$$= arg \min_{cell_i \in LM_j} \{score(cell_j^{<x,y>})\}$$

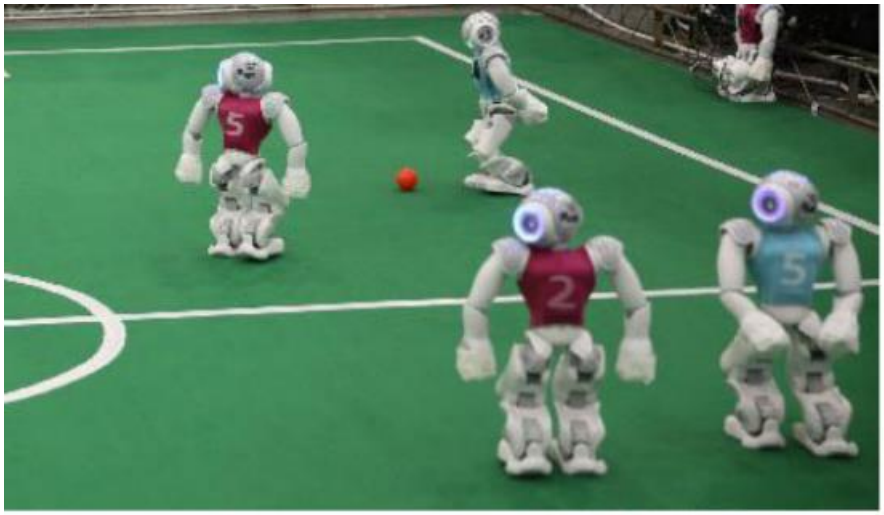


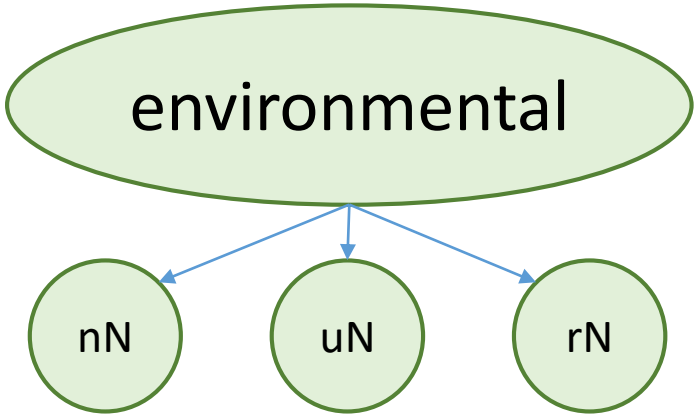
# Experimental Results




Average Cumulative time during which the **ball was not seen** in a game for the contexts **Throw-In** and **Uninformed Search** (100 simulated tests).

# Experimental Results






**detect network-contexts**

varying the **reliability of network** communication

| Team | Wins | Losts | Ties | Games |
|---|---|---|---|---|
| Blue | 95 | 36 | 42 | 173 |

# Search for a Non-adversarial Target

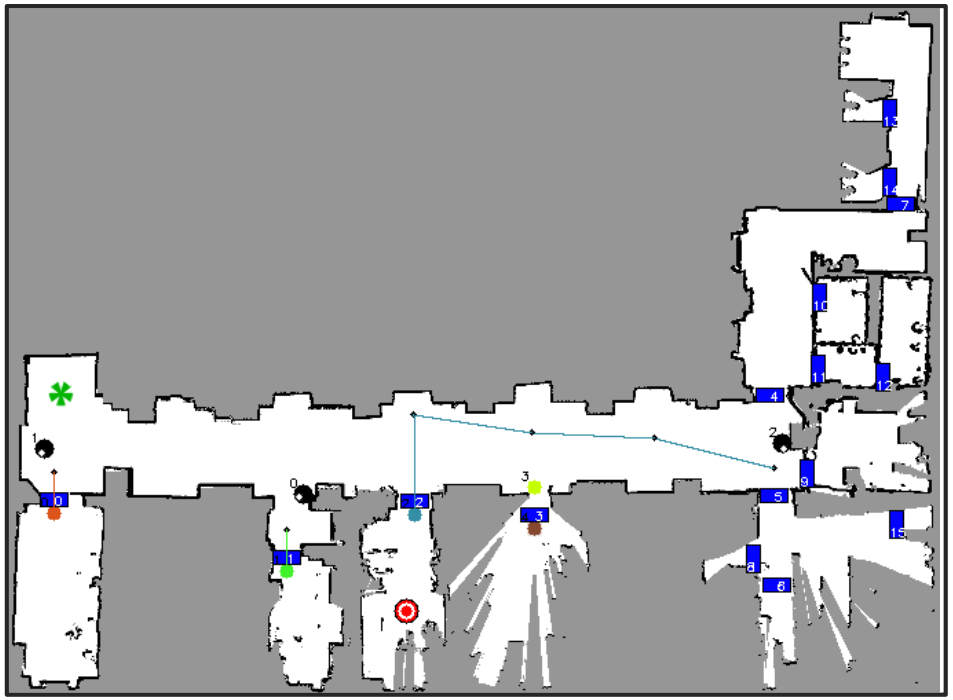
## Task Assignment

$$UEV_i(t) = [b_{(i,1)}(t), \ldots, b_{(i,m)}(t)]$$

$$UEM_i(t) = [UEV_1\ (t), \ldots, UEV_n\ (t)]^T$$

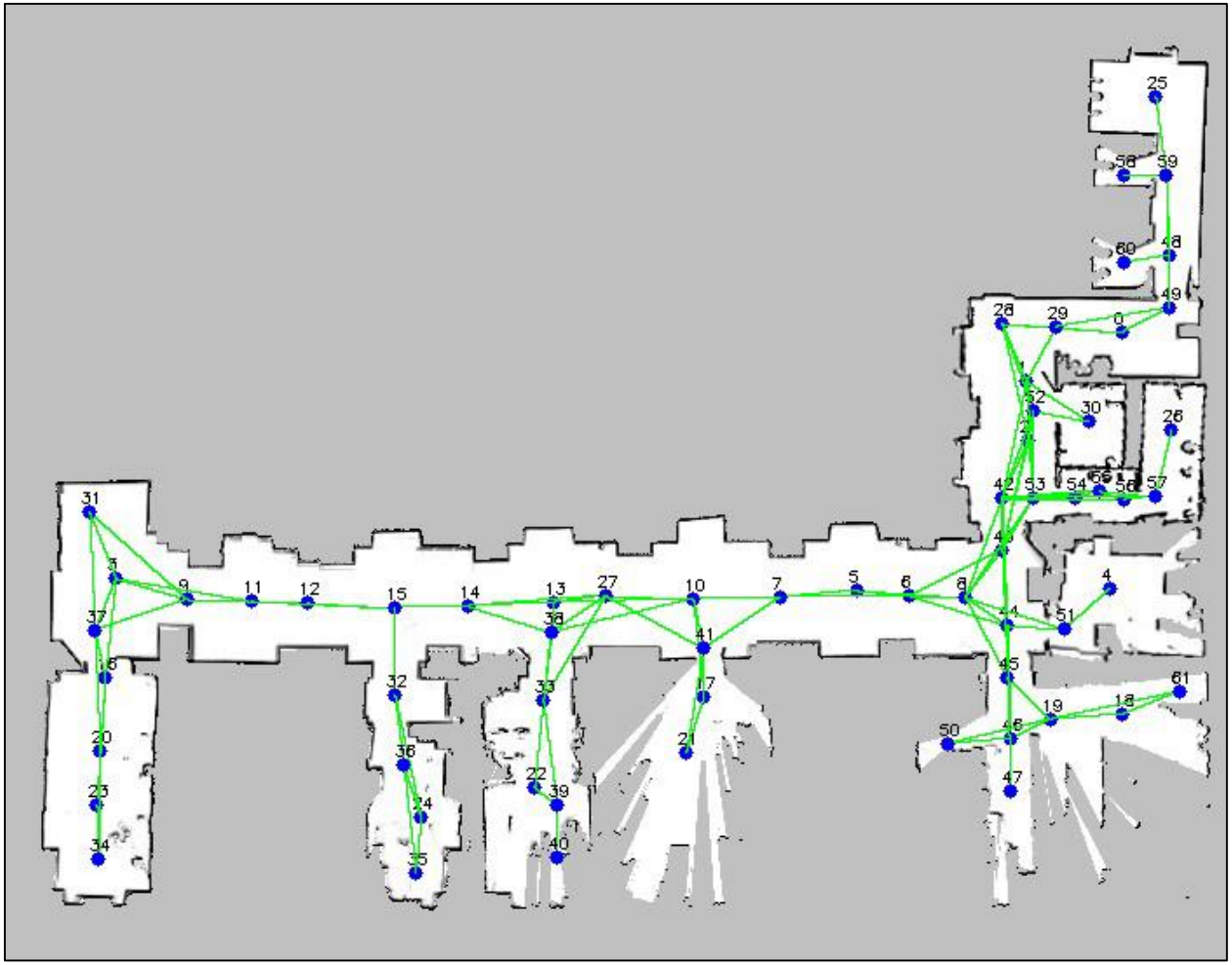
$$< r_i, \tau_j > = \arg\max_i utility(\tau_j)$$

| | T0 | T1 | T2 | T3 |
|---|---|---|---|---|
| R0 | 11939 0 1104 | 571 | 872 | 225 |
| R1 | 11940 0 393 | 1312 | 755 | 691 |
| R2 | 11941 0 742 | 416 | 1273 | 114 |
| R4 | 11942 0 284 | 869 | 650 | 784 |



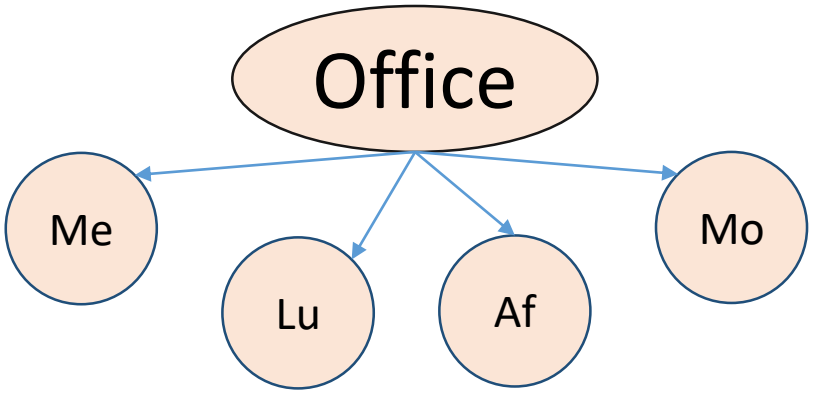
# Search for a Non-adversarial Target

## World Representation



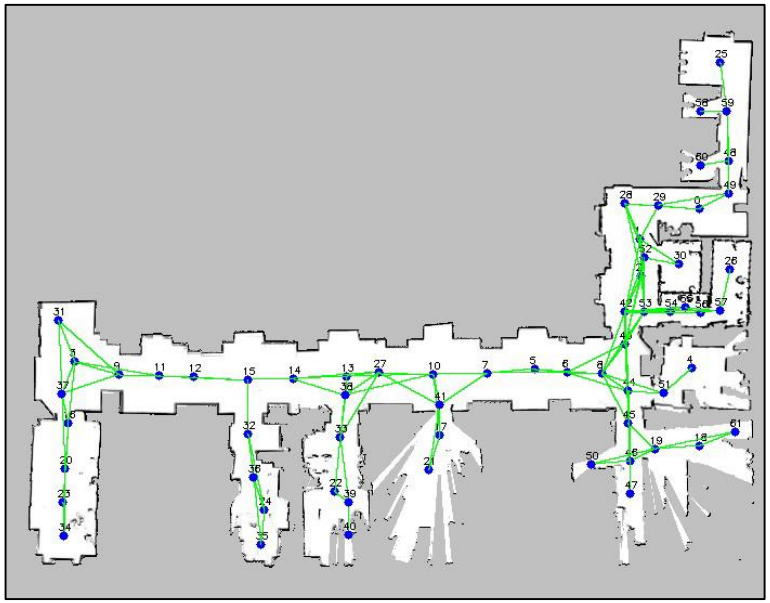
# Search for a Non-adversarial Target

## World Modeling




**Events:**

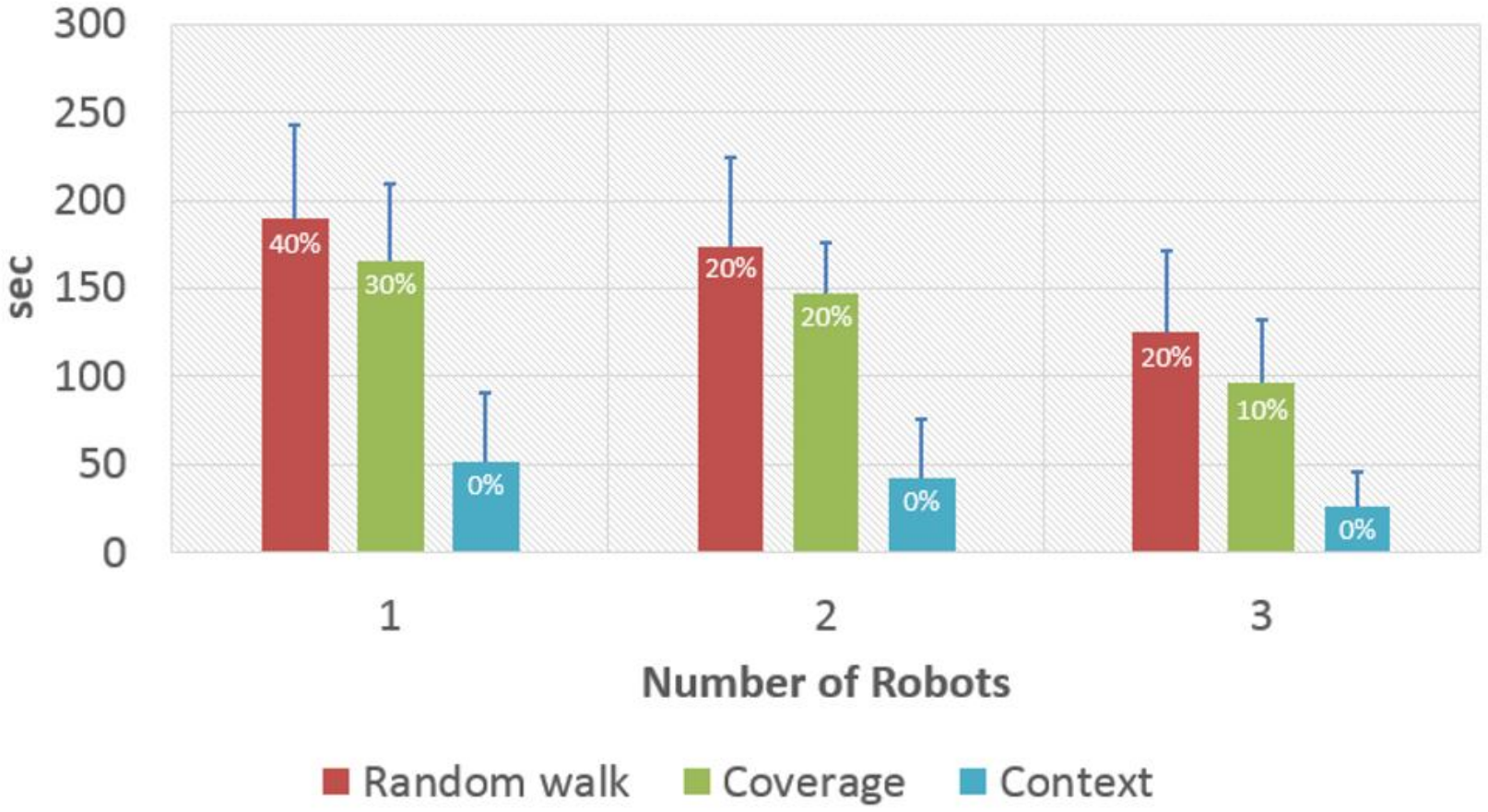
- target near location
- door opened/closed
- clear area
- person found



Reconstruction function ***f***:

$$DWM_i = \forall n \quad node_i^n$$
$$= arg \min_{node_i \in LM_j} \{score(node_j^n)\}$$

# Experimental Results




Average time needed to **locate the target**

The percentages represent the ratio of failed tasks

SAPIENZA
UNIVERSITÀ DI ROMA

# Context-Aware Multi-Robot Coordination

F. Riccio, G. Gemignani, D. Nardi

# Concluding Remarks

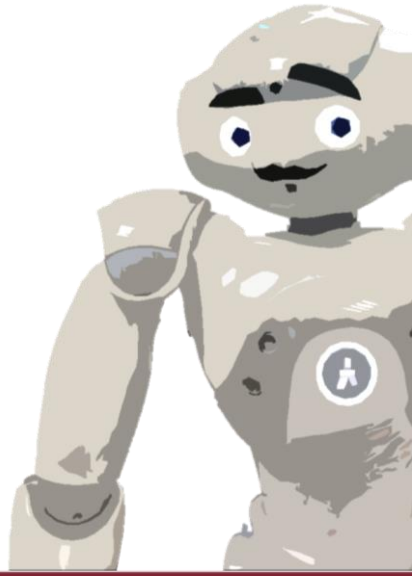
## Contribution

- **Integration of Distributed** world modeling, task assignment and contextual knowledge
- Active **adaptation** strategy

## Limitation and Future Work

- Relax assumptions on context consensus
- Autonomously detect events and recognize contexts by learning world dynamics

(detecting routines Fentanes et al. @IROS-2016)



```
void ContextCoordinator::update(Role& role) {
    if (theRobotInfo.number == 5) plotUtilities();

    role.role = Role::RoleType::goalie;

    if (theUtilityShare.context == 0){
        switch( (int) theRobotInfo.number ){
        case 2:
            role.role = Role::RoleType::LASTNAME_losing_role2;
            break;
        case 3:
            role.role = Role::RoleType::LASTNAME_losing_role3;
            break;
        case 4:
            role.role = Role::RoleType::LASTNAME_losing_role4;
            break;
        case 5:
            role.role = Role::RoleType::LASTNAME_losing_role5;
            break;
        }
    } else if (theUtilityShare.context == 1){

        switch( (int) theRobotInfo.number ){
        case 2:
            role.role = Role::RoleType::LASTNAME_outnumbered_role2;
            break;
        case 3:
            role.role = Role::RoleType::LASTNAME_outnumbered_role3;
            break;
        case 4:
            role.role = Role::RoleType::LASTNAME_outnumbered_role4;
            break;
        case 5:
            role.role = Role::RoleType::LASTNAME_outnumbered_role5;
            break;
        }

    } else { //PLAYING CONTEXT

        switch( (int) theRobotInfo.number ){
        case 2:
            role.role = Role::RoleType::LASTNAME_role2;
            break;
        case 3:
            role.role = Role::RoleType::LASTNAME_role3;
            break;
        case 4:
            role.role = Role::RoleType::LASTNAME_role4;
            break;
        case 5:
            role.role = Role::RoleType::LASTNAME_role5;
            break;
        }

    }
}
```

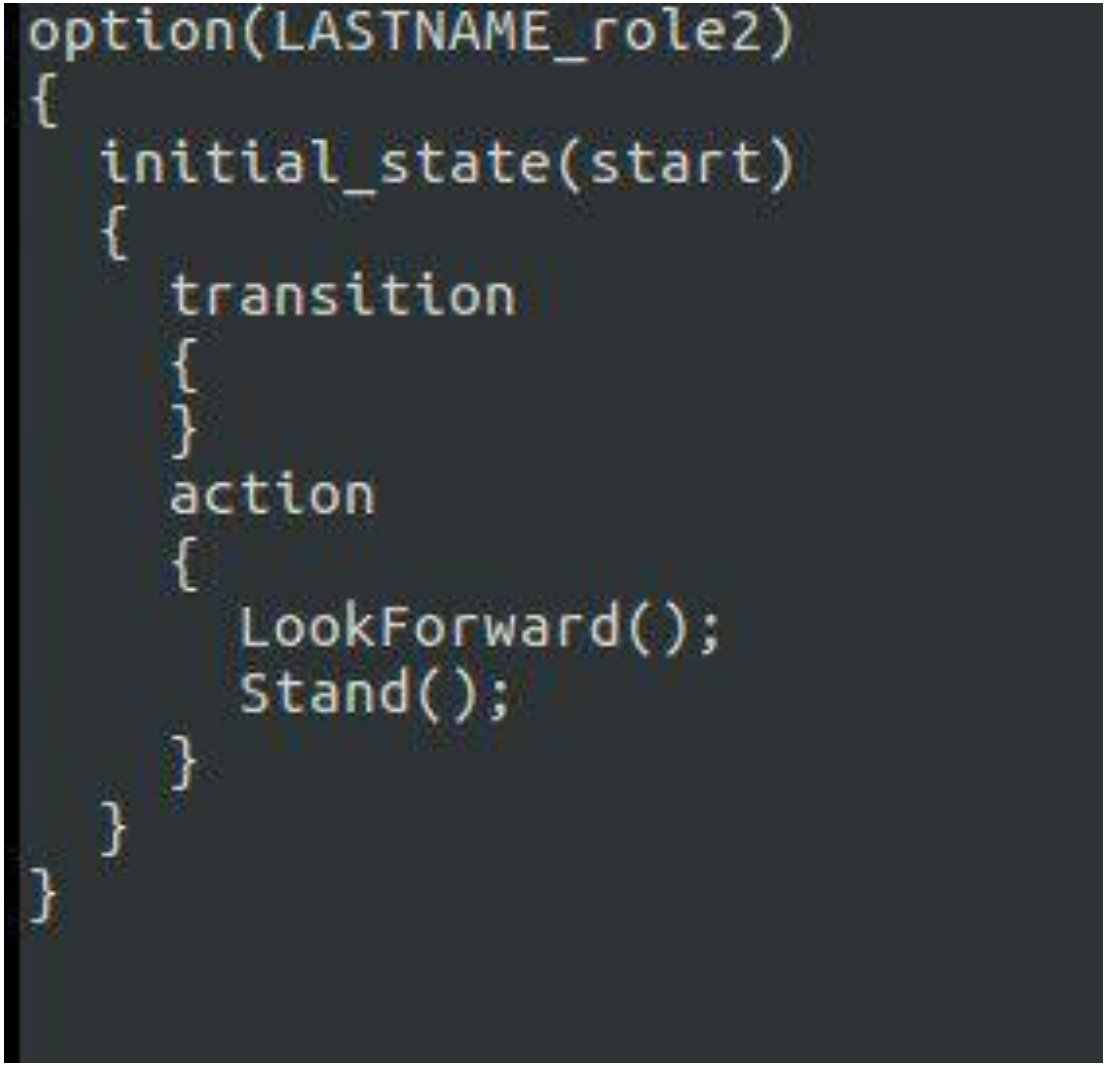
```cpp
#pragma once

#include <iostream>
#include <set>
#include "Tools/Module/Module.h"
#include "Tools/Math/Transformation.h"
#include "Representations/Modeling/RobotPose.h"
#include "Representations/Infrastructure/FrameInfo.h"
#include "Representations/Infrastructure/RobotInfo.h"
#include "Representations/Infrastructure/GameInfo.h"
#include "Representations/Infrastructure/TeamInfo.h"
#include "Representations/Sensing/FallDownState.h"
#include "Representations/Communication/TeamData.h"
#include "Representations/spqr_representations/ConfigurationParameters.h"
#include "Representations/spqr_representations/OurDefinitions.h"
#include "Representations/spqr_representations/UtilityShare.h"

#include "Platform/SystemCall.h"
#include "Platform/Time.h"
#include <mutex>

MODULE(UtilityShareProvider,
{,
 REQUIRES(GameInfo),
 REQUIRES(OpponentTeamInfo),
 REQUIRES(OwnTeamInfo),
 REQUIRES(RobotInfo),
 REQUIRES(RobotPose),
 REQUIRES(BallModel),
 REQUIRES(FrameInfo),
 REQUIRES(FallDownState),
 REQUIRES(TeamData),
 PROVIDES(UtilityShare),
});

class UtilityShareProvider : public UtilityShareProviderBase
{
private:
    bool weAreLosing();
    bool outnumberedByOpponents();
public:
    void update(UtilityShare& us);
    UtilityShareProvider();
};
```

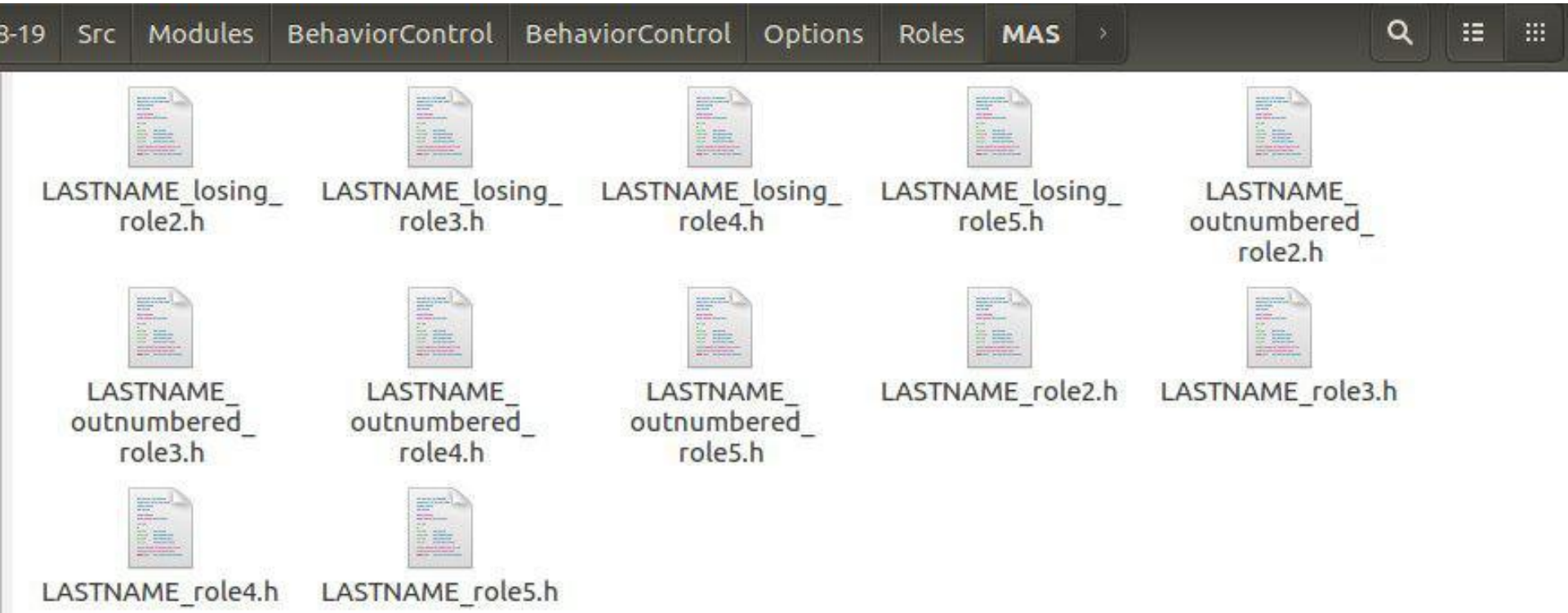
```cpp
#include "UtilityShareProvider.h"

#include <unistd.h>
#include <iostream>


UtilityShareProvider::UtilityShareProvider(){
    SPQR::ConfigurationParameters();
}


bool UtilityShareProvider::weAreLosing(){
    if ( static_cast<int>(theOwnTeamInfo.score) < static_cast<int>(theOpponentTeamInfo.score))
        return true;
    return false;
}

bool UtilityShareProvider::outnumberedByOpponents(){
    uint obstacle_counter = 0;
    for (auto const& obstacle : theTeamPlayersModel.obstacles) {
        if (obstacle.isOpponent() && obstacle.center.x() < 0.f) {
            obstacle_counter++;
        }
    }
    if (obstacle_counter > 2)
        return true;
    return false;
}

void UtilityShareProvider::update(UtilityShare& us) {

    us.role2_utility = 1.1;
    us.role3_utility = 1.2;
    us.role4_utility = 1.3;
    us.role5_utility = 1.4;
    us.context = weAreLosing() ?
                            0 : outnumberedByOpponents() ?
                                        1 : 2;
}

MAKE_MODULE(UtilityShareProvider, modeling)
```

```
option(PlayingState)
{
    initial_state(demo)
    {
        action
        {
            //  PLAYING CONTEXT
            if(theRole.role == Role::RoleType::goalie)
                Goalie();
            else if(theRole.role== Role::RoleType::LASTNAME_role2)
                LASTNAME_role2();
            else if(theRole.role== Role::RoleType::LASTNAME_role3)
                LASTNAME_role3();
            else if(theRole.role == Role::RoleType::LASTNAME_role4) {
                LASTNAME_role4();
            } else if(theRole.role == Role::RoleType::LASTNAME_role5)
                Striker();

            //  LOOSING CONTEXT
            else if(theRole.role == Role::RoleType::LASTNAME_losing_role2)
                LASTNAME_losing_role2;
            else if(theRole.role == Role::RoleType::LASTNAME_losing_role3)
                LASTNAME_losing_role3;
            else if(theRole.role == Role::RoleType::LASTNAME_losing_role4)
                LASTNAME_losing_role4;
            else if(theRole.role == Role::RoleType::LASTNAME_losing_role5)
                LASTNAME_losing_role5;


            //  OUT NUMBERED BY OPPONENTS CONTEXT
            else if(theRole.role == Role::RoleType::LASTNAME_outnumbered_role2)
                LASTNAME_outnumbered_role2;
            else if(theRole.role == Role::RoleType::LASTNAME_outnumbered_role3)
                LASTNAME_outnumbered_role3;
            else if(theRole.role == Role::RoleType::LASTNAME_outnumbered_role4)
                LASTNAME_outnumbered_role4;
            else if(theRole.role == Role::RoleType::LASTNAME_outnumbered_role5)
                LASTNAME_outnumbered_role5;
        }
    }
}
```

```
 */
STREAMABLE(Role,
{
  /** The different roles */
  ENUM(RoleType,    //SPQR Roles
  {,
   undefined,
   goalie,
   striker,
   defender,
   supporter,
   jolly,
   LASTNAME_role2,
   LASTNAME_role3,
   LASTNAME_role4,
   LASTNAME_role5,
   LASTNAME_losing_role2,
   LASTNAME_losing_role3,
   LASTNAME_losing_role4,
   LASTNAME_losing_role5,
   LASTNAME_outnumbered_role2,
   LASTNAME_outnumbered_role3,
   LASTNAME_outnumbered_role4,
   LASTNAME_outnumbered_role5,
   searcher_1,
   searcher_2,
   searcher_3,
   searcher_4,
   penaltyStriker,
   penaltyKeeper,
   planStriker,
   planJolly,
   none,

  });
```

```
//MAS PROJECT
#include "Options/Roles/MAS/LASTNAME_role2.h"
#include "Options/Roles/MAS/LASTNAME_role3.h"
#include "Options/Roles/MAS/LASTNAME_role4.h"
#include "Options/Roles/MAS/LASTNAME_role5.h"
#include "Options/Roles/MAS/LASTNAME_losing_role2.h"
#include "Options/Roles/MAS/LASTNAME_losing_role3.h"
#include "Options/Roles/MAS/LASTNAME_losing_role4.h"
#include "Options/Roles/MAS/LASTNAME_losing_role5.h"
#include "Options/Roles/MAS/LASTNAME_outnumbered_role2.h"
#include "Options/Roles/MAS/LASTNAME_outnumbered_role3.h"
#include "Options/Roles/MAS/LASTNAME_outnumbered_role4.h"
#include "Options/Roles/MAS/LASTNAME_outnumbered_role5.h"
```


option(LASTNAME_role2)
{
initial_state(start)
{
transition
{
}
action
{
LookForward();
Stand();
}
}
}


8-19
Src
Modules
BehaviorControl
BehaviorControl
Options
Roles
MAS
LASTNAME_losing_role2.h
LASTNAME_losing_role3.h
LASTNAME_losing_role4.h
LASTNAME_losing_role5.h
LASTNAME_outnumbered_role2.h
LASTNAME_outnumbered_role3.h
LASTNAME_outnumbered_role4.h
LASTNAME_outnumbered_role5.h
LASTNAME_role2.h
LASTNAME_role3.h
LASTNAME_role4.h
LASTNAME_role5.h