Player Behavior and Team Strategy in Robocup Simulation 3D League

Georgios Methenitis

Technical University of Crete

Chania, August, 2012

Thesis Committee
Assistant Professor Michail G. Lagoudakis (ECE)
Assistant Professor Georgios Chalkiadakis (ECE)
Professor Minos Garofalakis (ECE)

Abstract

This thesis presents a complete team design for the RoboCup 3D Simulation League focusing on player behavior, team strategy, and team coordination. Our agents are designed in a way that enables them to act effectively both autonomously and as members of the team.

Outline of Topics

Robocup Competition

- RoboCup is an international robotics competition.

Robocup Competition

- RoboCup is an international robotics competition.
- Founded in 1997.

Robocup Competition

- RoboCup is an international robotics competition.
- Founded in 1997.
- The official goal of the project is stated as an ambitious endeavor: "By the year 2050, a team of fully autonomous humanoid robot soccer players shall win the soccer game, complying with the official rule of the FIFA, against the winner of the most recent World Cup".

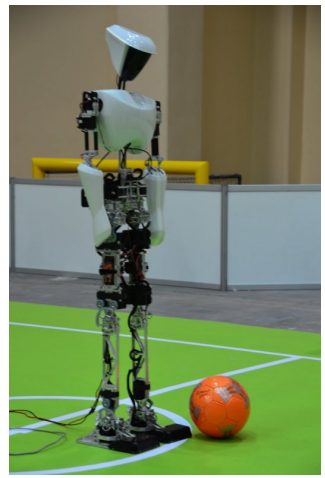
Soccer

Standard Platform League

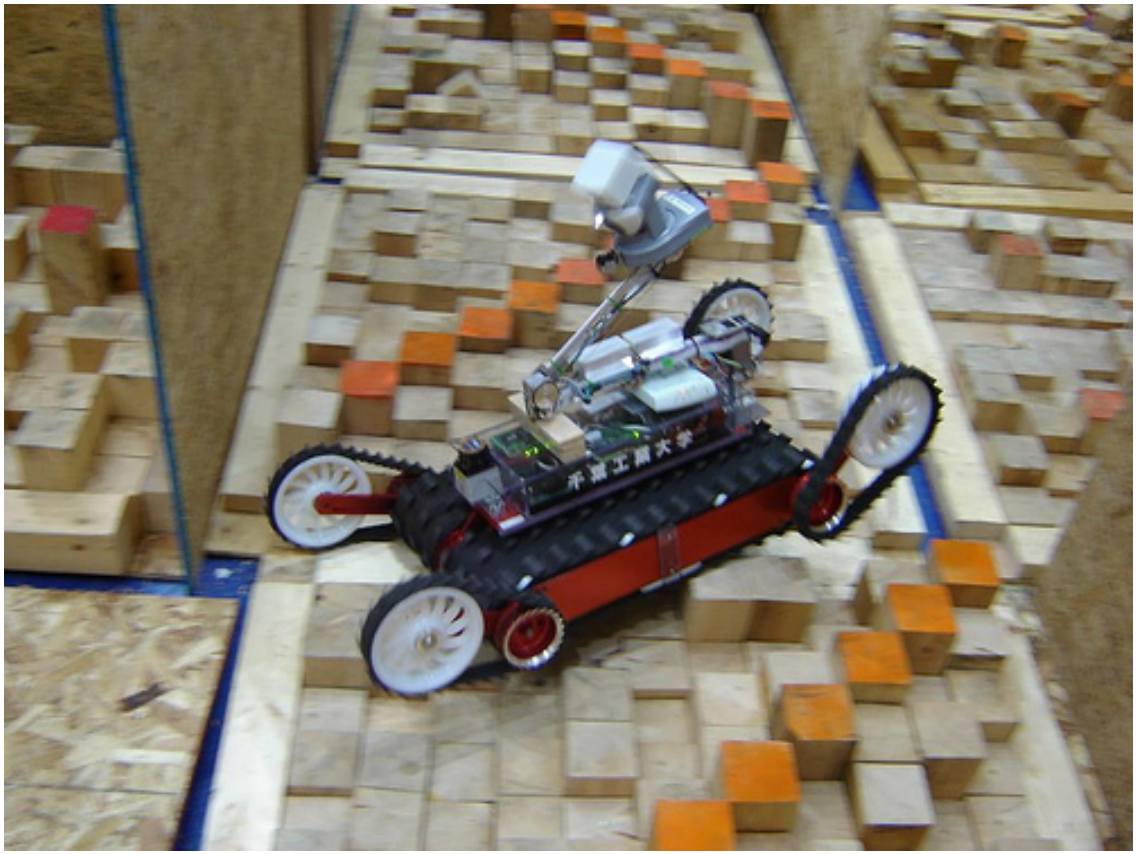
- Same robot platform

Soccer

Humanoid League



Rescue



@Home

Simulation League

- One of the oldest leagues in RoboCup Soccer.

Simulation League

- One of the oldest leagues in RoboCup Soccer.
- Independently moving software players (agents).

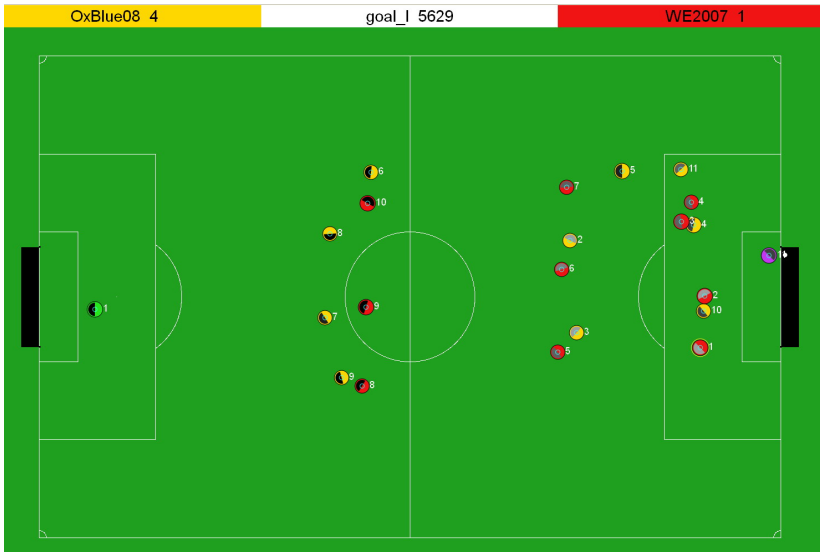
Simulation League

- One of the oldest leagues in RoboCup Soccer.
- Independently moving software players (agents).
- virtual field inside a computer.

Simulation League

- One of the oldest leagues in RoboCup Soccer.
- Independently moving software players (agents).
- virtual field inside a computer.
- 2D League, 3D League.

Simulation League 2D vs 3D



3D Simulation Soccer

- At its beginning, spherical agent.

3D Simulation Soccer

- At its beginning, spherical agent.
- In 2006, Fujitsu HOAP-2 robot.

3D Simulation Soccer

- At its beginning, spherical agent.
- In 2006, Fujitsu HOAP-2 robot.
- In 2008, Nao robot model.

3D Simulation Soccer

- At its beginning, spherical agent.
- In 2006, Fujitsu HOAP-2 robot.
- In 2008, Nao robot model.
- **SimSpark** is used as the official Robocup 3D simulator.

SimSpark

- **SimSpark** is a generic physics simulator system.

SimSpark

- **SimSpark** is a generic physics simulator system.
- Multiple agents in three-dimensional environment.

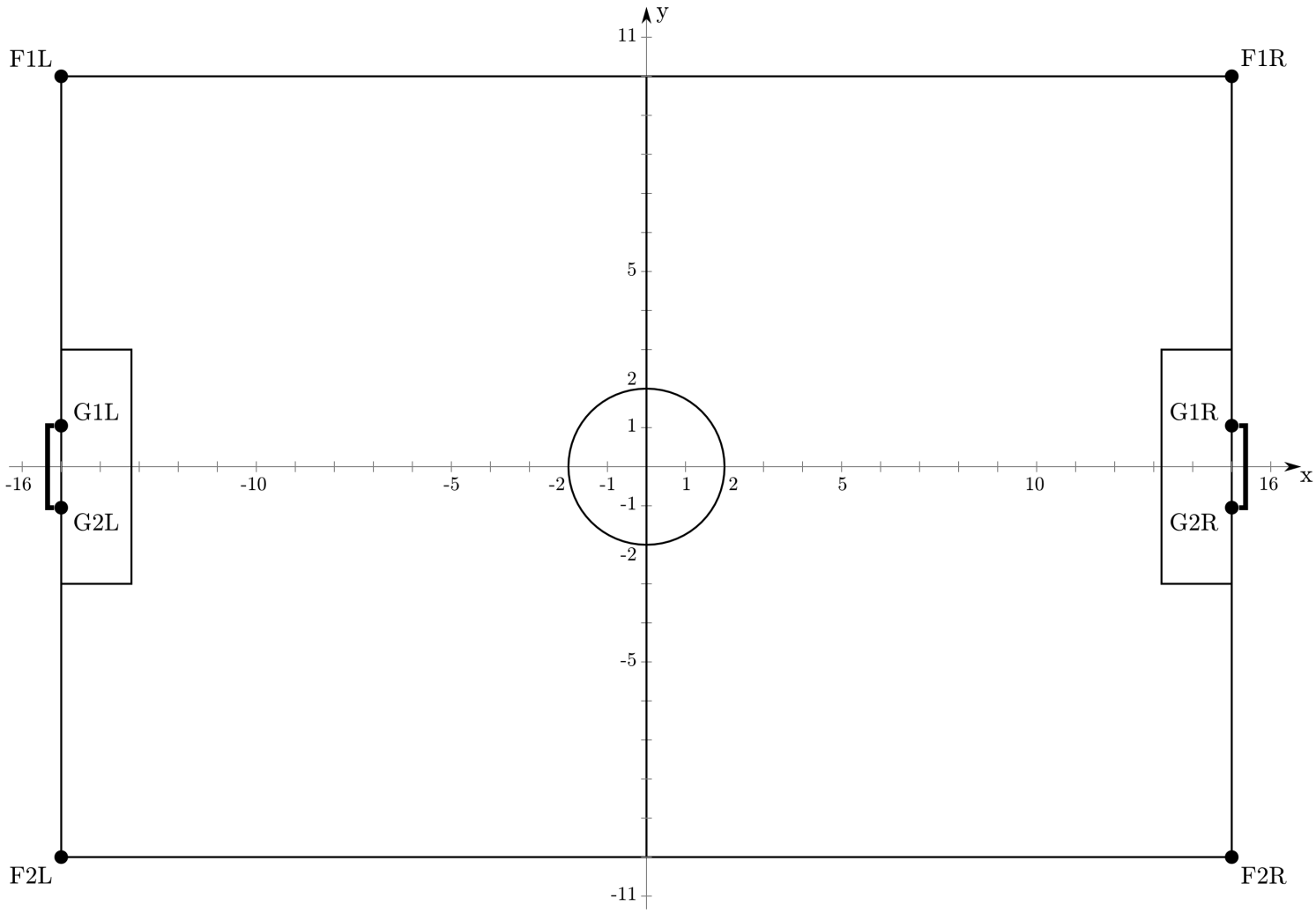
SimSpark

- **SimSpark** is a generic physics simulator system.
- Multiple agents in three-dimensional environment.
- *Rcssserver3d* is the official competition environment for the RoboCup 3D Simulation League.

Server's Versions

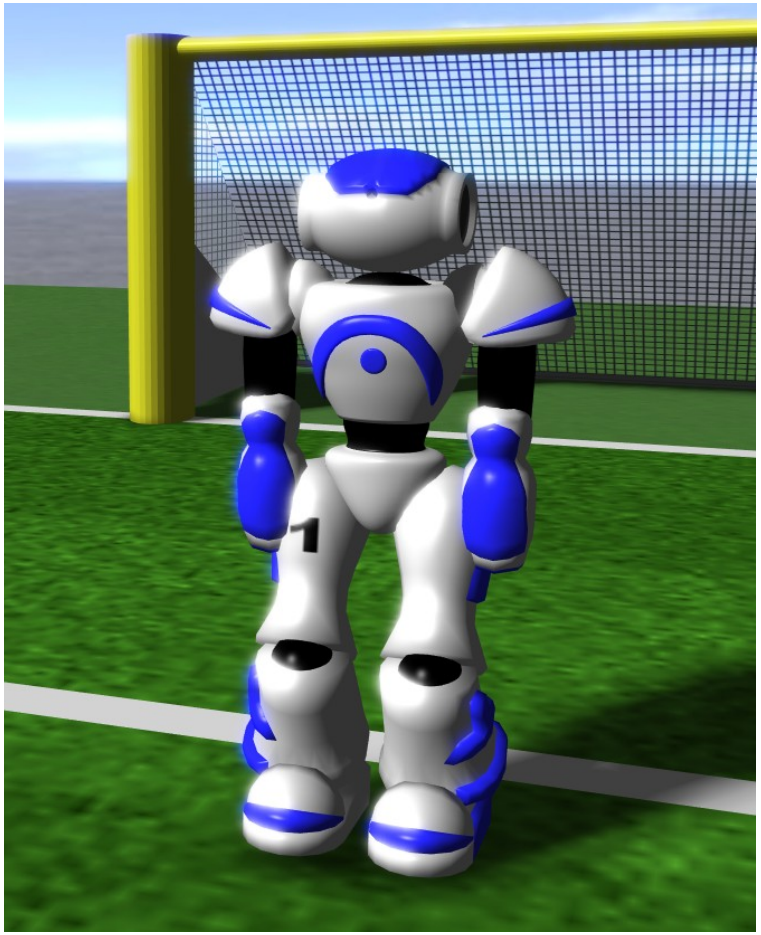
- Version 0.6.5
  - Players 9
  - Length 21m
  - Width 14m
- Version 0.6.6
  - Players 11
  - Length 30m
  - Width 20m

Soccer Field



Robot Model

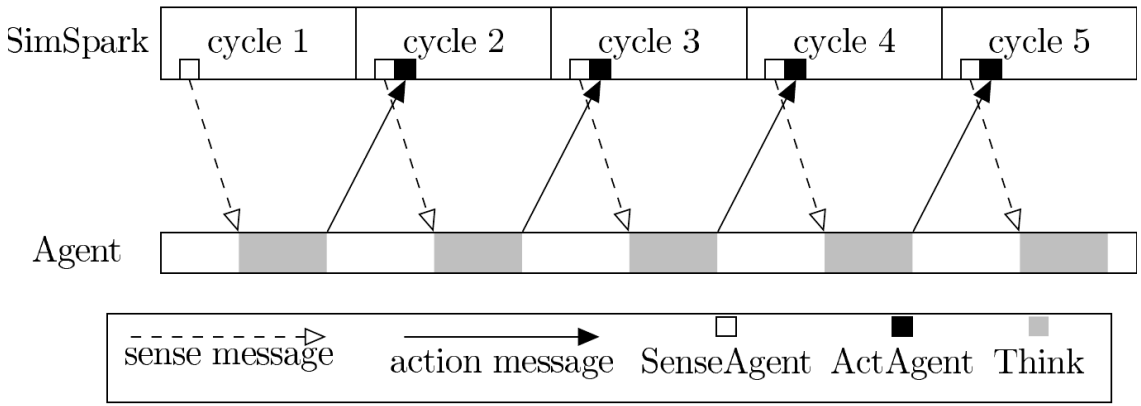
The Nao humanoid robot manufactured by Aldebaran Robotics. Its height is about 57cm and its weight is around 4.5kg. The simulated model comes with 22 degrees of freedom.



Server

The SimSpark server hosts the process that manages and advances the simulation.

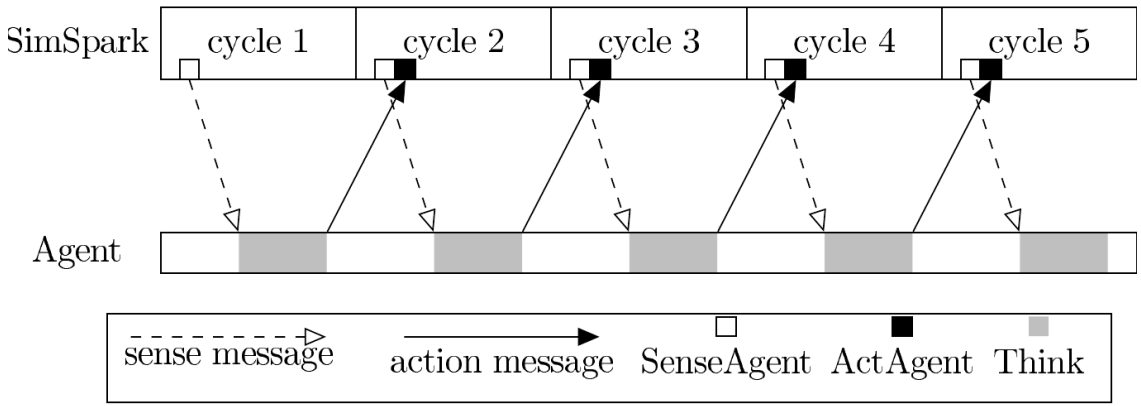
- Simulation Cycle, 20ms



Server

The SimSpark server hosts the process that manages and advances the simulation.

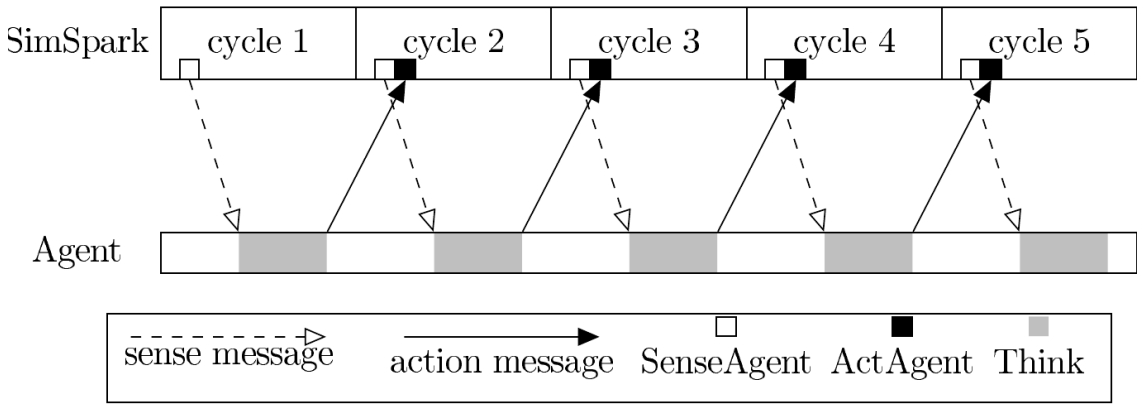
- Simulation Cycle, 20ms
- Sense



Server

The SimSpark server hosts the process that manages and advances the simulation.

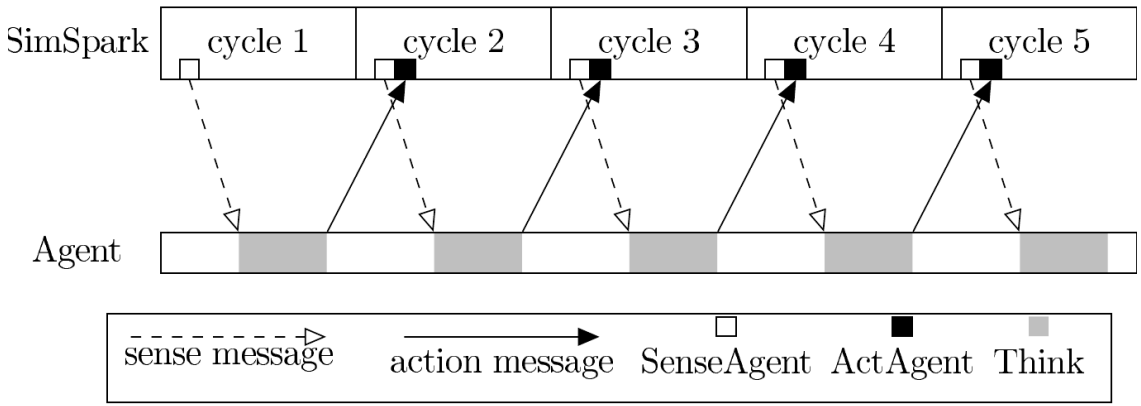
- Simulation Cycle, 20ms
- Sense
- Think



Server

The SimSpark server hosts the process that manages and advances the simulation.

- Simulation Cycle, 20ms
- Sense
- Think
- Act



Agent Perceptors

Perceptors are the senses of an agent, allowing awareness of the agent's model state and the environment.

- HingeJoint Perceptor

Agent Perceptors

Perceptors are the senses of an agent, allowing awareness of the agent's model state and the environment.

- HingeJoint Perceptor
- ForceResistance Perceptor

Agent Perceptors

Perceptors are the senses of an agent, allowing awareness of the agent's model state and the environment.

- HingeJoint Perceptor
- ForceResistance Perceptor
- GyroRate Perceptor

Agent Perceptors

Perceptors are the senses of an agent, allowing awareness of the agent's model state and the environment.

- HingeJoint Perceptor
- ForceResistance Perceptor
- GyroRate Perceptor
- Accelerometer Perceptor

Agent Perceptors

Perceptors are the senses of an agent, allowing awareness of the agent's model state and the environment.

- HingeJoint Perceptor
- ForceResistance Perceptor
- GyroRate Perceptor
- Accelerometer Perceptor
- Vision Perceptor

Agent Perceptors

Perceptors are the senses of an agent, allowing awareness of the agent's model state and the environment.

- HingeJoint Perceptor
- ForceResistance Perceptor
- GyroRate Perceptor
- Accelerometer Perceptor
- Vision Perceptor
- Hear Perceptor

Agent Perceptors

Perceptors are the senses of an agent, allowing awareness of the agent's model state and the environment.

- HingeJoint Perceptor
- ForceResistance Perceptor
- GyroRate Perceptor
- Accelerometer Perceptor
- Vision Perceptor
- Hear Perceptor
- GameState Perceptor

Agent Effectors

Effectors allow agents to perform actions within the simulation. Agents control them by sending messages to the server and the server changes the game state accordingly.

- Create Effector

Agent Effectors

Effectors allow agents to perform actions within the simulation. Agents control them by sending messages to the server and the server changes the game state accordingly.

- Create Effector
- HingeJoint Effector

Agent Effectors

Effectors allow agents to perform actions within the simulation. Agents control them by sending messages to the server and the server changes the game state accordingly.

- Create Effector
- HingeJoint Effector
- Synchronize Effector

Agent Effectors

Effectors allow agents to perform actions within the simulation. Agents control them by sending messages to the server and the server changes the game state accordingly.

- Create Effector
- HingeJoint Effector
- Synchronize Effector
- Init Effector

Agent Effectors

Effectors allow agents to perform actions within the simulation. Agents control them by sending messages to the server and the server changes the game state accordingly.

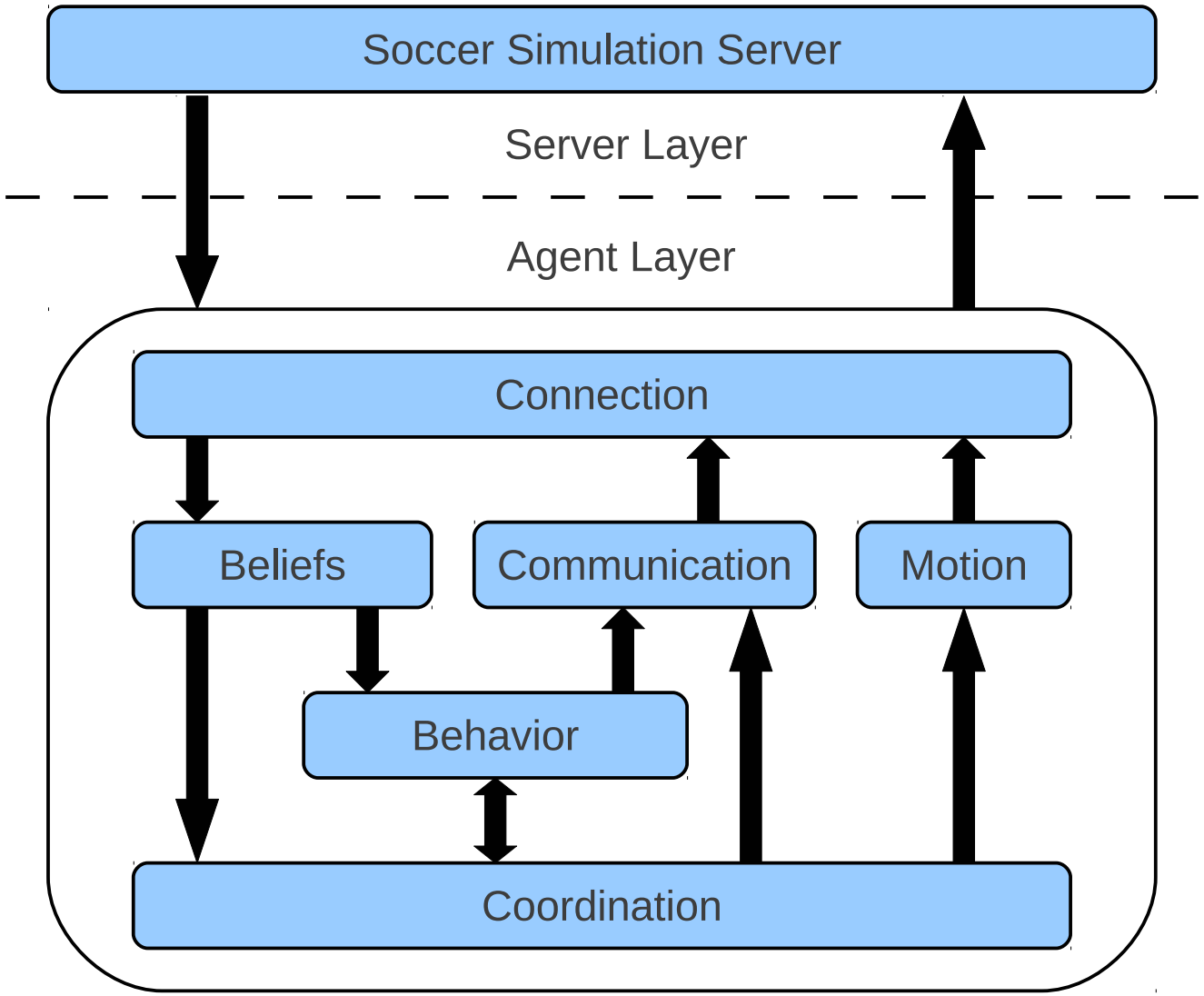
- Create Effector
- HingeJoint Effector
- Synchronize Effector
- Init Effector
- Beam Effector

Agent Effectors

Effectors allow agents to perform actions within the simulation. Agents control them by sending messages to the server and the server changes the game state accordingly.

- Create Effector
- HingeJoint Effector
- Synchronize Effector
- Init Effector
- Beam Effector
- Say Effector

Agents Architecture



Connection

- Agent connected to the server at all times during a simulated game.

Connection

- Agent connected to the server at all times during a simulated game.
- Sense messages from the server every 20ms.

Connection

- Agent connected to the server at all times during a simulated game.
- Sense messages from the server every 20ms.
- Action messages, at the end of their think cycles.

Perception

- Different from real robot soccer games.

Perception

- Different from real robot soccer games.
- No raw data.

Perception

- Different from real robot soccer games.
- No raw data.
- Perceptor Messages.

Perception

- Different from real robot soccer games.
- No raw data.
- Perceptor Messages.
- Agents update their beliefs and store sensors' data parsing these messages.

Perception Message Example

```
(time (now 46.20))(GS (t 0.00) (pm BeforeKickOff))(GYR (n torso)
(rt 0.00 0.00 0.00))(ACC (n torso) (a 0.00 -0.00 9.81))(HJ (n hj
1)(ax 0.00))(HJ (n hj2) (ax 0.01))(See (G2R (pol 14.83 -11.81 1.
08))(G1R (pol 14.54 -3.66 1.12)) (F1R (pol 15.36 19.12 -1.91))(F
2R (pol 17.07 -31.86 -1.83)) (B (pol 4.51 -26.40 -6.15)) (P (tea
m AST_3D)(id 8)(rlowerarm (pol 0.18 -35.78 -21.65)) (llowerarm (
pol 0.19 34.94-21.49)))(L (pol 8.01 -60.03 -3.87) (pol 6.42 51.1
90 -39.13 -5.17))(L (pol 5.91 -39.06 -5.11) (pol 6.28-29.26 -4.8
8)) (L (pol 6.28 29.34 -4.95)(pol 6.16 -19.05 -5.00)))(HJ(n raj1
) (ax -0.01))(HJ (n raj2) (ax -0.00))(HJ (n raj3)(ax -0.00))(HJ(
n raj4) (ax 0.00))(HJ (n laj1) (ax 0.01))(HJ (n laj2) (ax 0.00)) ...
```

Self-Localization

- Executed every three cycles (60ms).

Self-Localization

- Executed every three cycles (60ms).
- Localization uses the eight visible landmarks into the field.

Self-Localization

- Executed every three cycles (60ms).
- Localization uses the eight visible landmarks into the field.
  - G1R, G2R, G1L, G2L

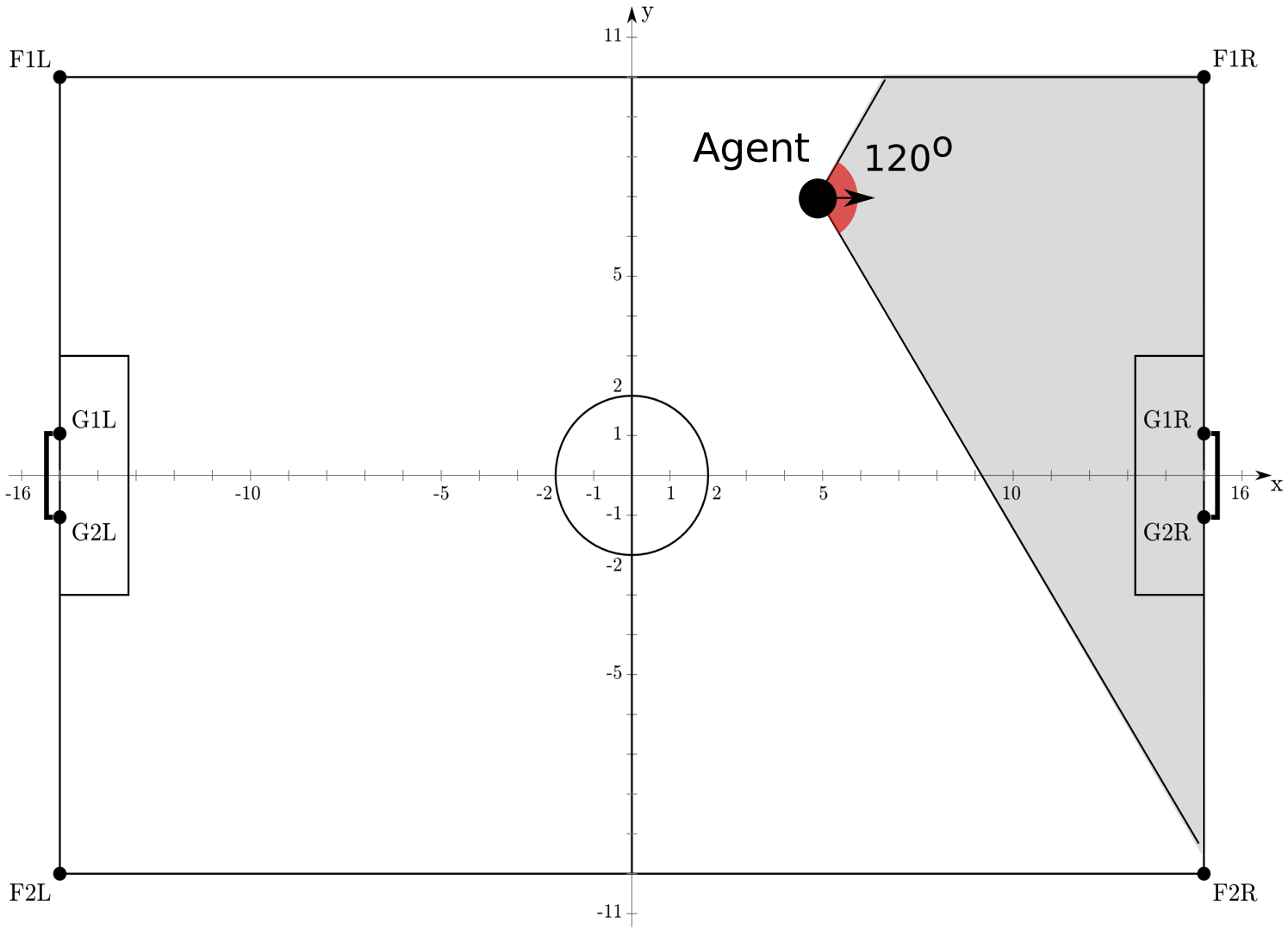
Self-Localization

- Executed every three cycles (60ms).
- Localization uses the eight visible landmarks into the field.
  - G1R, G2R, G1L, G2L
  - F1R, F2R, F1L, F2L

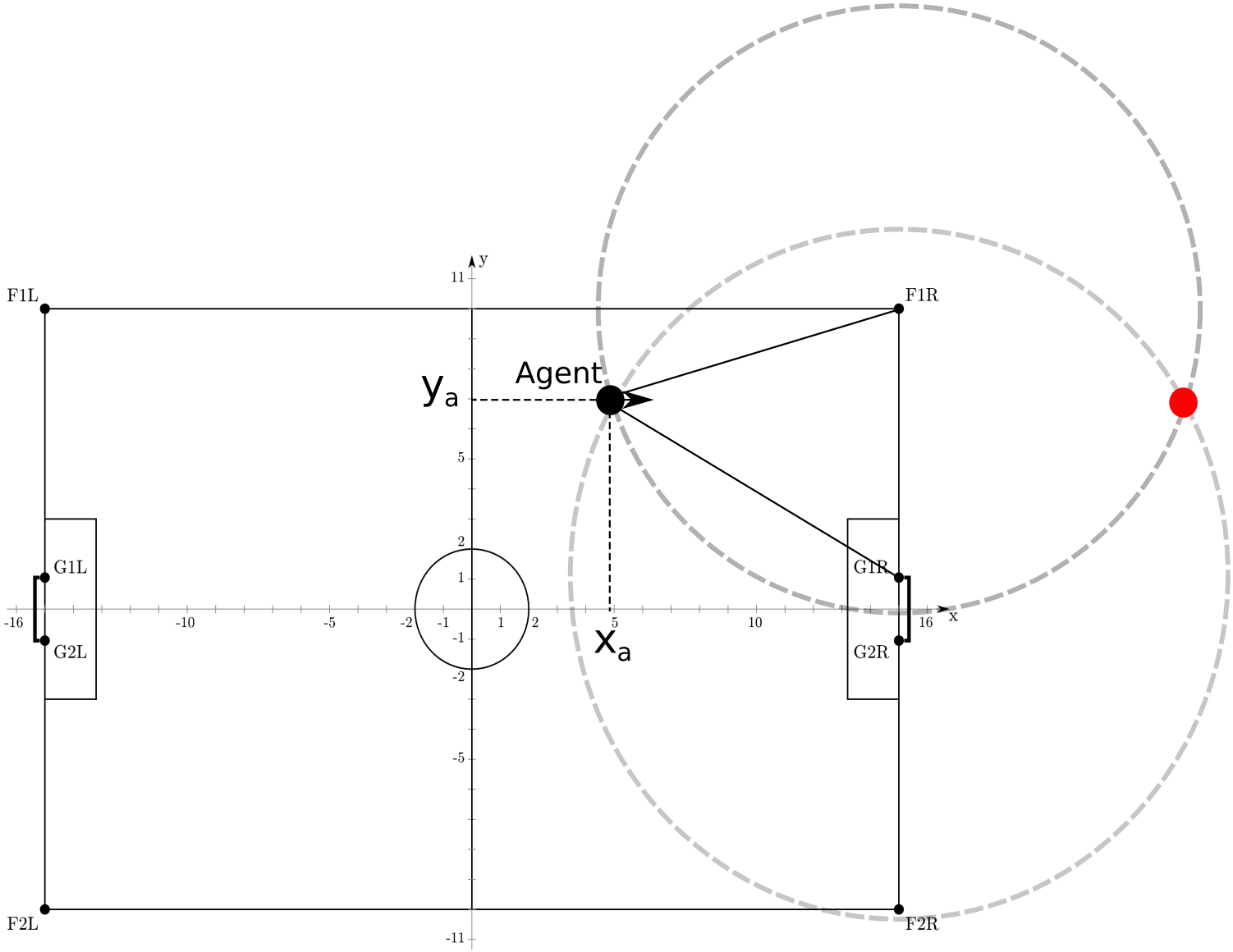
Self-Localization

- Executed every three cycles (60ms).
- Localization uses the eight visible landmarks into the field.
  - G1R, G2R, G1L, G2L
  - F1R, F2R, F1L, F2L
- Limited field of view (120 Degrees).

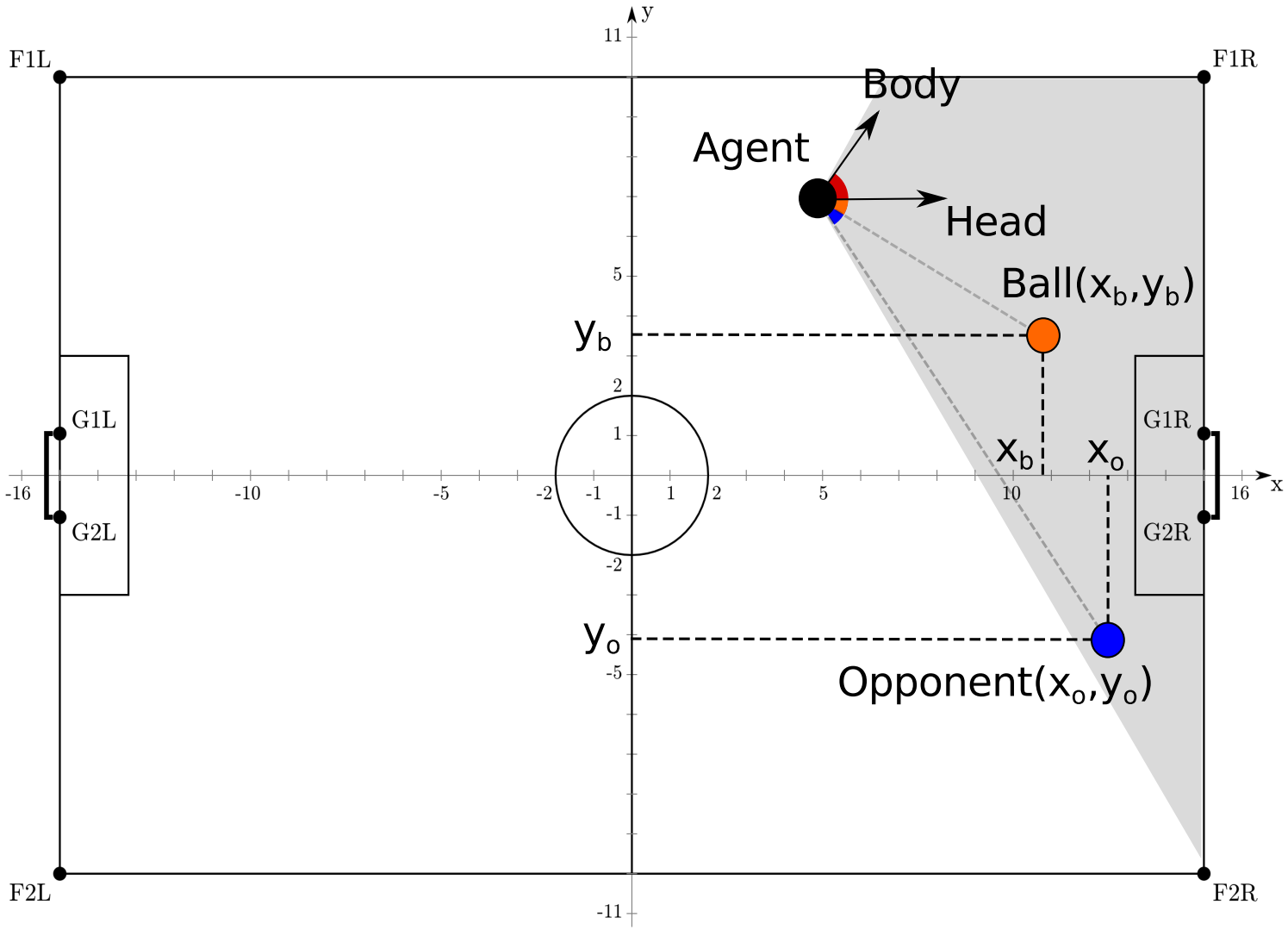
Simulated Nao's Restricted Field of View



Self-Localization Technique



Localization Results



Localization Filtering

Why a localization filtering is needed?

- Absence of a more sophisticated probabilistic localization scheme.

Filtering is applicable for ball and self localization.

Localization Filtering

Why a localization filtering is needed?

- Absence of a more sophisticated probabilistic localization scheme.
- Temporary absences of observations from self-localization.

Filtering is applicable for ball and self localization.

Localization Filtering

Why a localization filtering is needed?

- Absence of a more sophisticated probabilistic localization scheme.
- Temporary absences of observations from self-localization.
- Noisy observations.

Filtering is applicable for ball and self localization.

Localization Filtering

Why a localization filtering is needed?

- Absence of a more sophisticated probabilistic localization scheme.
- Temporary absences of observations from self-localization.
- Noisy observations.
- Filter incoming observations.

Filtering is applicable for ball and self localization.

Localization Filtering

Why a localization filtering is needed?

- Absence of a more sophisticated probabilistic localization scheme.
- Temporary absences of observations from self-localization.
- Noisy observations.
- Filter incoming observations.
- Filter incoming observations.

Filtering is applicable for ball and self localization.

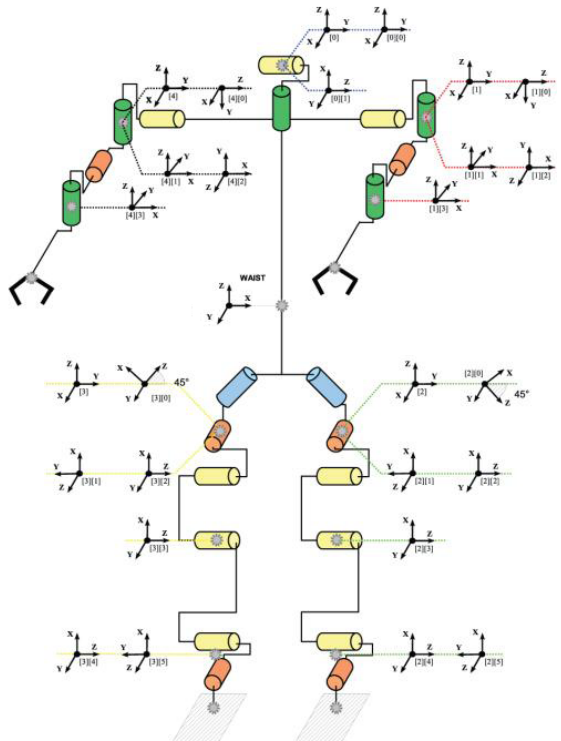
Localization Filtering Algorithm

Algorithm 1 Localization Filtering

1: **Input:** *LastEstimate*
2: **Output:** *FilteredLocation*
3: *Queue*: a FIFO queue storing the *MaxSize* (default=10) most recent estimates
4:
5: **if** $size(Queue) = 0$ **then**
6: $\quad Queue.Add(LastEstimate)$
7: **else if** $LastEstimate \not\approx AverageLocation(Queue)$ **then**
8: $\quad Queue.Remove()$
9: **else**
10: $\quad$ **if** $size(Queue) = MaxSize$ **then**
11: $\quad\quad Queue.Remove()$
12: $\quad$ **end if**
13: $\quad Queue.Add(LastEstimate)$
14: **end if**
15: **return** $AverageLocation(Queue)$

Nao's Anatomy

The simulated Nao robot comes with 22 degrees of freedom, corresponding to 22 hinge joints.



Motion and Movement

- In robotics, a complex motion is commonly defined as a sequence of timed joint poses.

Motion and Movement

- In robotics, a complex motion is commonly defined as a sequence of timed joint poses.
- A pose is a set of values for every joint in the robot's body or in a specific kinematic chain at a given time.

$$Pose(t) = \{J_1(t), J_2(t), \ldots, J_n(t)\}$$

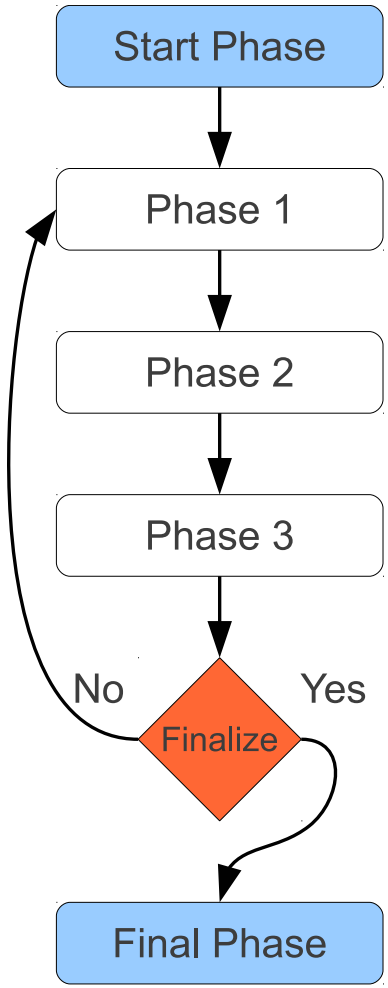
XML-Based Motion Files

```
<phase name="Start" next="Phase1">
   <effectors>
      Joint Values
   </effectors>
   <duration>duration</duration>
</phase>

<phase name="Phase1" next="Phase2">
   <effectors>
      Joint Values
   </effectors>
   <duration>duration</duration>
</phase>

<phase name="Phase2" next="Phase1">
   <effectors>
      Joint Values
   </effectors>
   <duration>duration</duration>
   <finalize>Final</finalize>
</phase>

<phase name="Final">
   <effectors>
      Joint Values
   </effectors>
   <duration>duration</duration>
</phase>
```



Motion and Movement

- To generate motions.

Motion and Movement

- To generate motions.
- Velocity computation is required.

Motion and Movement

- To generate motions.
- Velocity computation is required.
- This velocity is computed as follows:

$$JointVelocity = \frac{DesiredJointValue - CurrentJointValue}{PhaseDuration}$$

Text-Based Motion Files

```
#WEBOTS_MOTION,V1.0
LHipYawPitch,LHipRoll,LHipPitch,LKneePitch,LAnklePitch,...
00:00:000,Pose1,0,-0.012,-0.525,1.05,-0.525,0.012,0,...
00:00:040,Pose2,0,-0.011,-0.525,1.05,-0.525,0.011,0,...
00:00:080,Pose3,0,-0.009,-0.525,1.05,-0.525,0.009,0,...
00:00:120,Pose4,0,-0.007,-0.525,1.05,-0.525,0.007,0,...
00:00:160,Pose5,0,-0.004,-0.525,1.05,-0.525,0.004,0,...
00:00:200,Pose6,0,0.001,-0.525,1.051,-0.525,-0.001,0,...
00:00:240,Pose7,0,0.006,-0.525,1.05,-0.525,-0.006,0,...
00:00:280,Pose8,0,0.012,-0.525,1.05,-0.525,-0.012,0,...
00:00:320,Pose9,0,0.024,-0.525,1.05,-0.525,-0.024,0,...
```

Text-Based Motion Controller

Parameters that can be modified:

- Duration, time between poses in simulation cycles.

Text-Based Motion Controller

Parameters that can be modified:

- Duration, time between poses in simulation cycles.
- PoseStep, step for advancing from pose to pose.

Text-Based Motion Controller

Parameters that can be modified:

- Duration, time between poses in simulation cycles.
- PoseStep, step for advancing from pose to pose.
- The desired velocity of each joint i is computed by:

$$JointVelocity_i = \frac{DesiredJointValue_i - CurrentJointValue_i}{Duration \times CycleDuration}$$

Dynamic Motion Elements

- Walk Leaning

Dynamic Motion Elements

- Walk Leaning
- Walk Slowdown

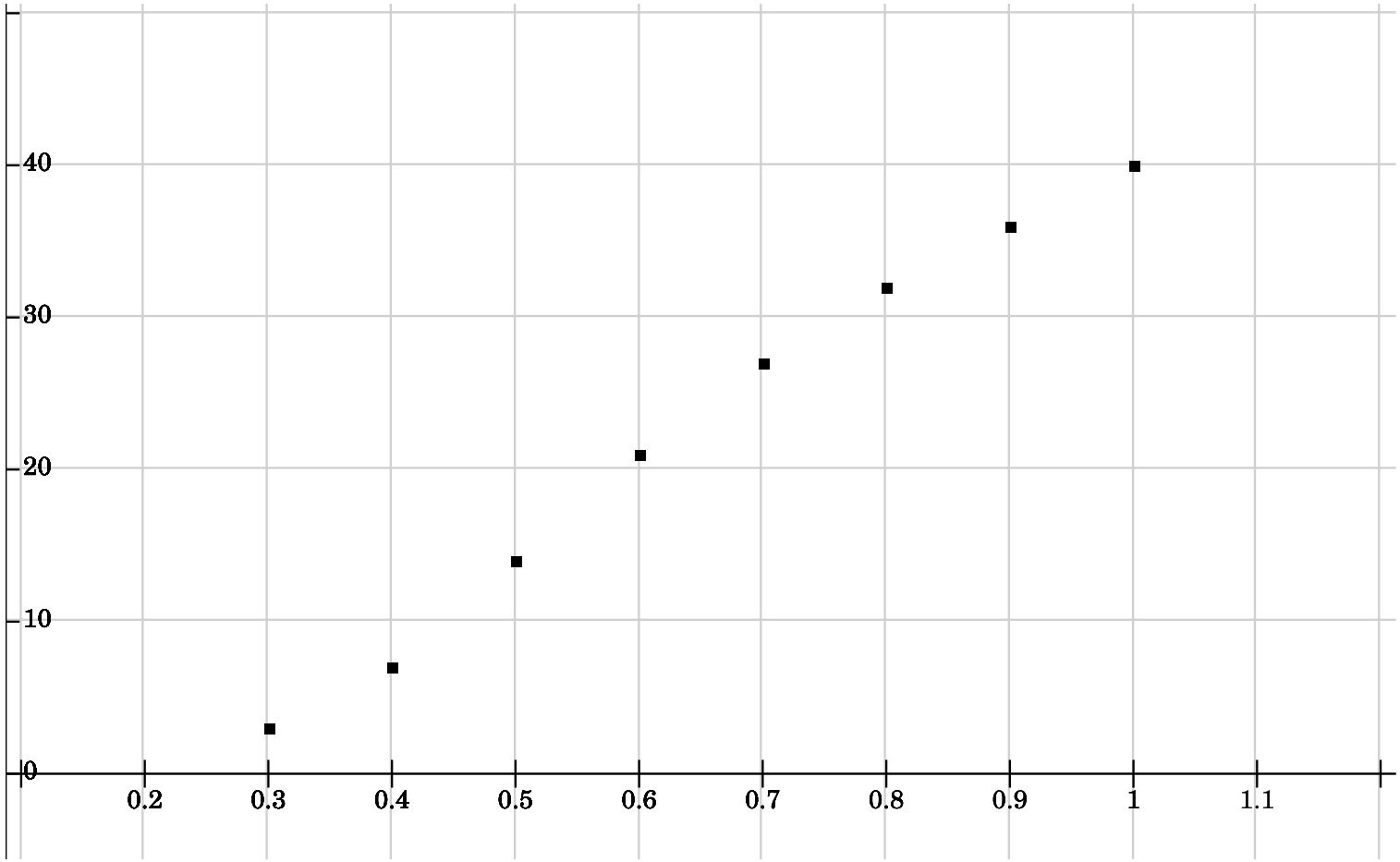
Dynamic Motion Elements

- Walk Leaning
- Walk Slowdown
- Dynamic Turn

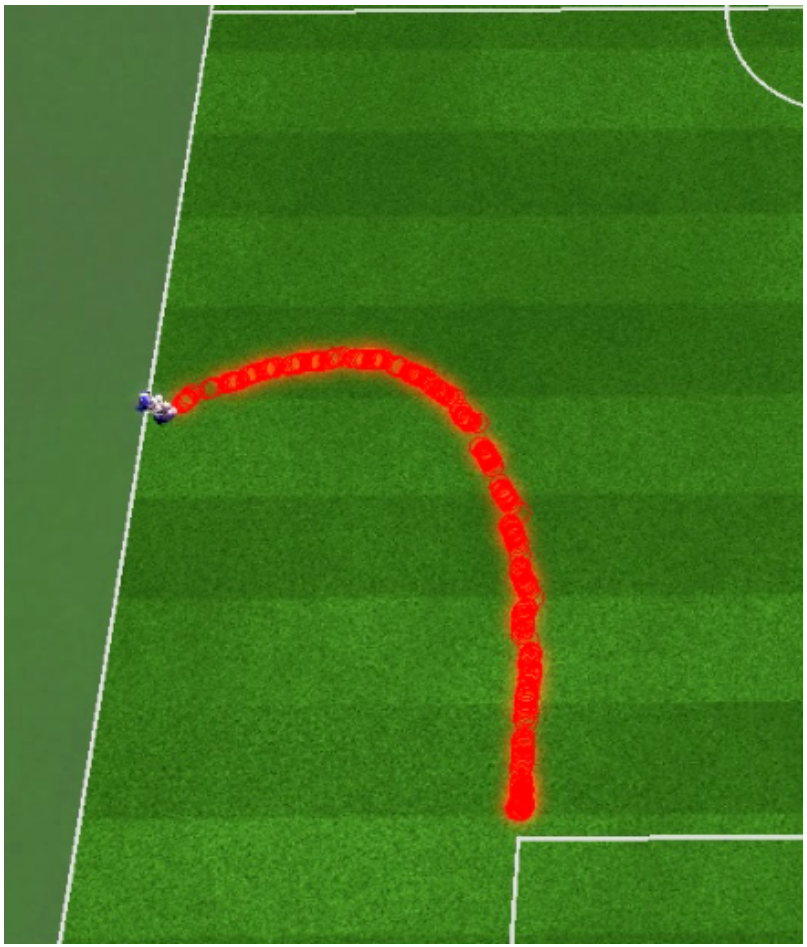
Dynamic Turn Example

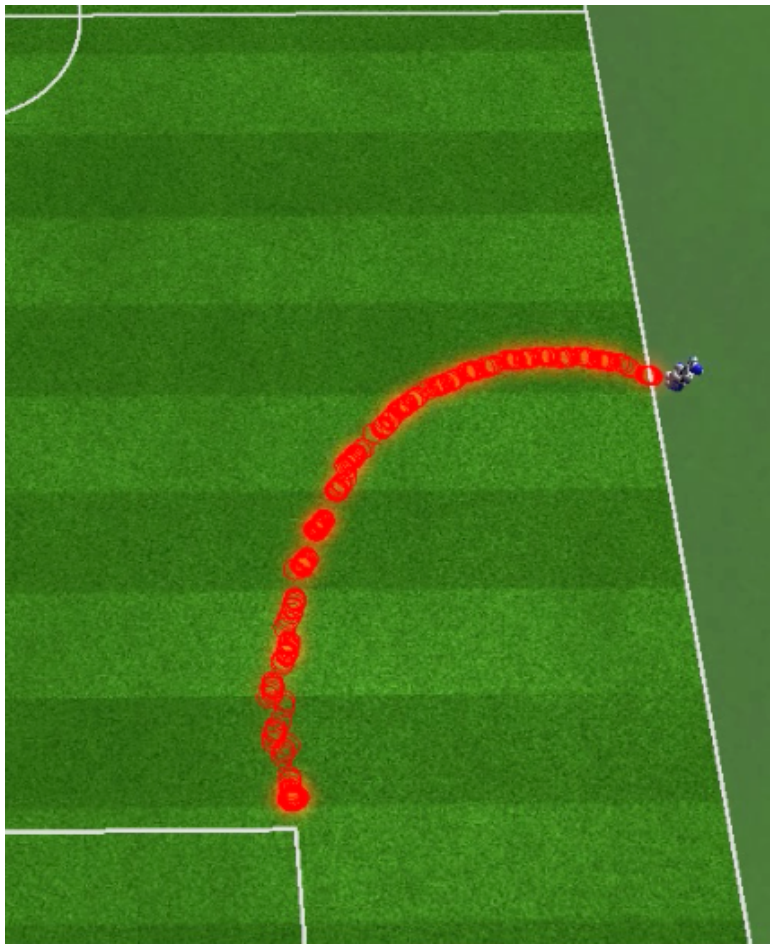
X-Axis Gain factor

Y-Axis Agent turn



Walk Leaning





Actions

Actions are split into groups in terms of their complexity:

- Basic Actions

Actions

Actions are split into groups in terms of their complexity:

- Basic Actions
- Complex Actions

Simple Actions

- Look Straight

Simple Actions

- Look Straight
- Scan

Simple Actions

- Look Straight
- Scan
- Pan Head

Simple Actions

- Look Straight
- Scan
- Pan Head
- Track Object

Simple Actions

- Look Straight
- Scan
- Pan Head
- Track Object
- Track Moving Object

Simple Actions

- Look Straight
- Scan
- Pan Head
- Track Object
- Track Moving Object
- Find Opponent'As Goals

Simple Actions

- Look Straight
- Scan
- Pan Head
- Track Object
- Track Moving Object
- Find Opponent'As Goals
- Look For Ball

Simple Actions

- Look Straight
- Scan
- Pan Head
- Track Object
- Track Moving Object
- Find Opponent'As Goals
- Look For Ball
- Turn To Ball

Simple Actions

- Look Straight
- Scan
- Pan Head
- Track Object
- Track Moving Object
- Find Opponent'As Goals
- Look For Ball
- Turn To Ball
- Turn To Localize

Simple Actions

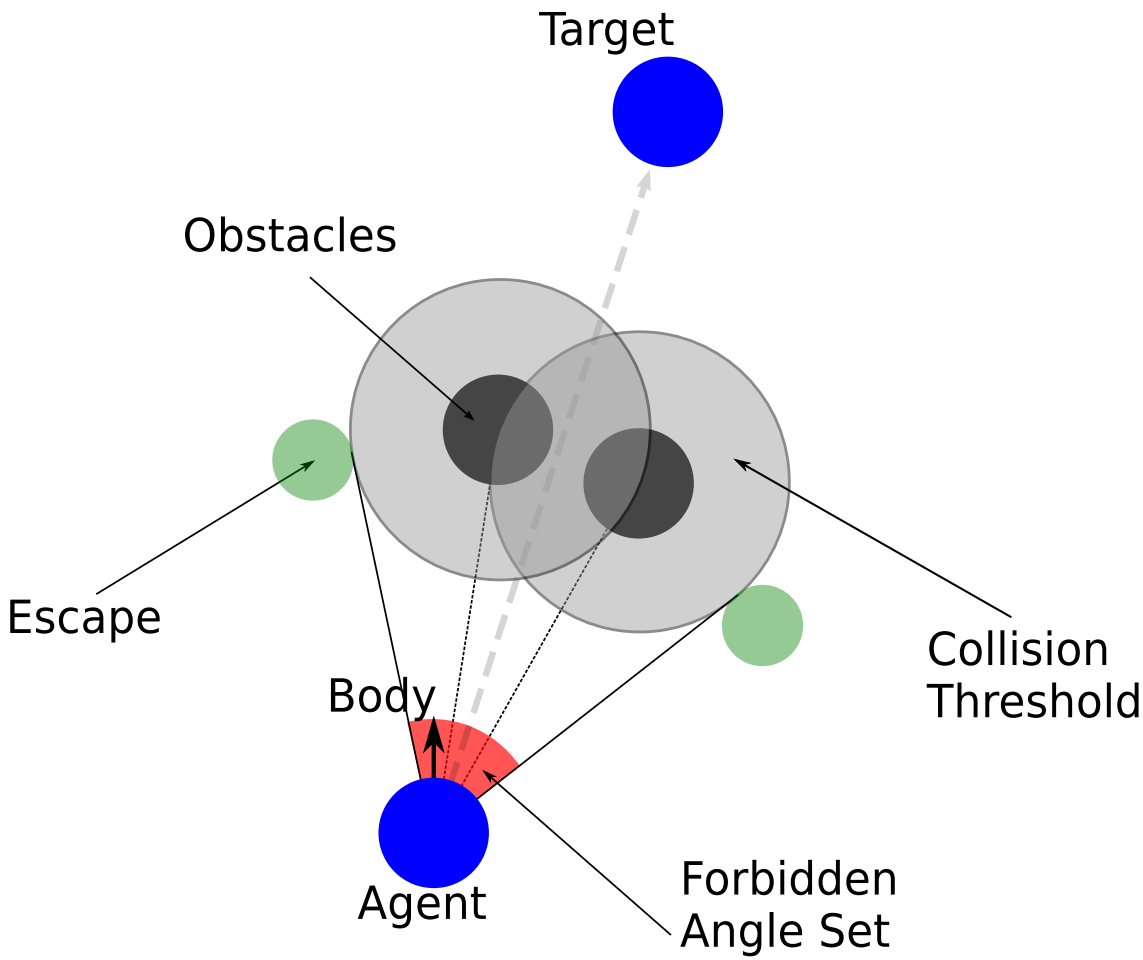
- Look Straight
- Scan
- Pan Head
- Track Object
- Track Moving Object
- Find Opponent'As Goals
- Look For Ball
- Turn To Ball
- Turn To Localize
- Stand Up

Simple Actions

- Look Straight
- Scan
- Pan Head
- Track Object
- Track Moving Object
- Find Opponent'As Goals
- Look For Ball
- Turn To Ball
- Turn To Localize
- Stand Up
- Prepare for Kick

Complex Actions

- Avoid Obstacles

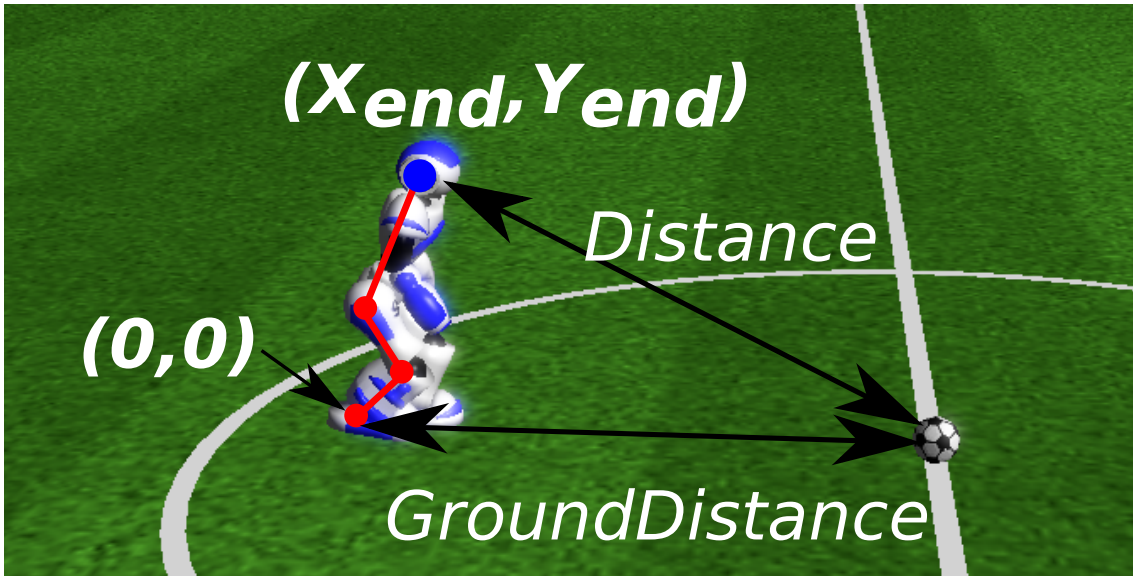


Complex Actions

- On Ball Action

Complex Actions

- On Ball Action
- Walk to Ball



Complex Actions

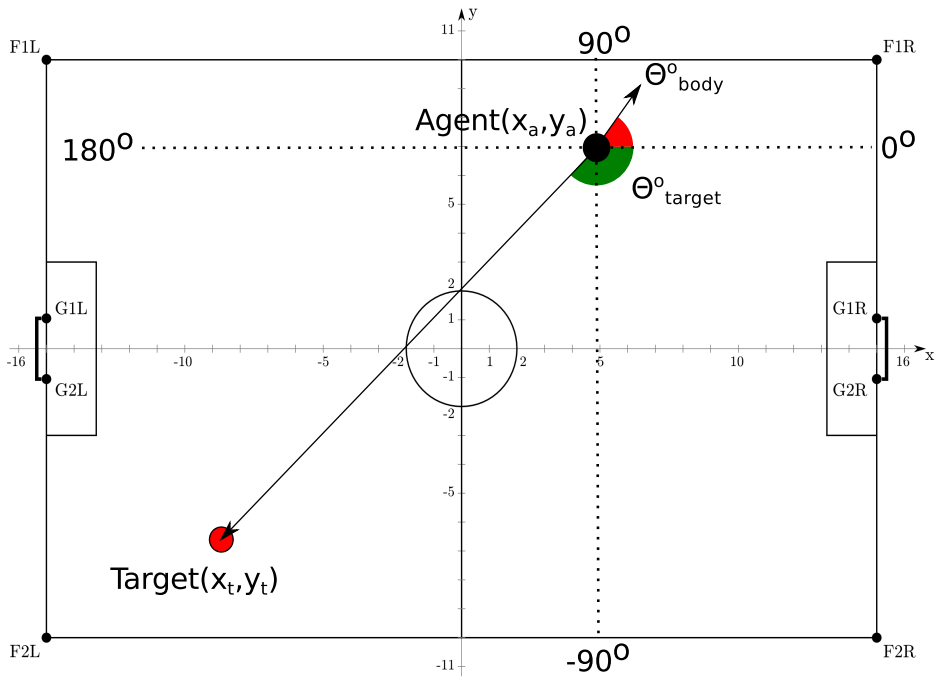
- Walk To Direction

Complex Actions

- Walk To Direction
- Dribble Ball To Direction

Complex Actions

- Walk To Direction
- Dribble Ball To Direction
- Walk to Coordinate



Communication Restrictions

- Maximum distance of 50 meters.

Communication Restrictions

- Maximum distance of 50 meters.
- Maximum length of 20 ASCII characters.

Communication Restrictions

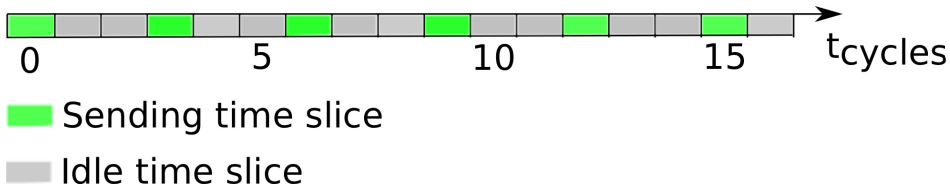
- Maximum distance of 50 meters.
- Maximum length of 20 ASCII characters.
- Only one message can be heard at any given time.

Communication Restrictions

- Maximum distance of 50 meters.
- Maximum length of 20 ASCII characters.
- Only one message can be heard at any given time.
- Messages from the same team can be heard only every other cycle.

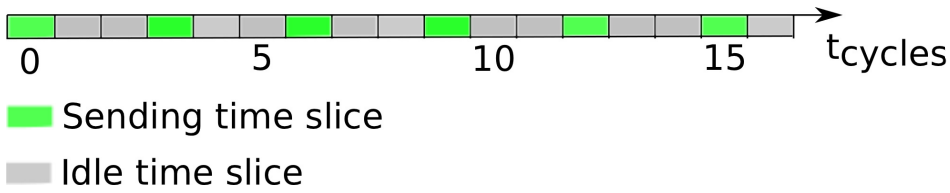
Communication Protocol

- Every time slice of the protocol has an associated integer label which indicates the uniform number of the player able to send its message at that slice.



Communication Protocol

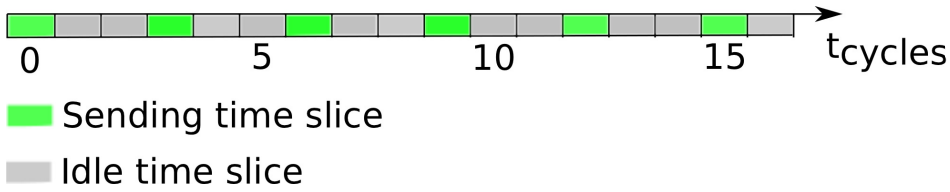
- Every time slice of the protocol has an associated integer label which indicates the uniform number of the player able to send its message at that slice.



- This label starts at 1 and grows by 1 every time a player sends a message.

Communication Protocol

- Every time slice of the protocol has an associated integer label which indicates the uniform number of the player able to send its message at that slice.



- This label starts at 1 and grows by 1 every time a player sends a message.
- Every player can receive reliably the messages from all teammates every 540ms (27 cycles) for a team of 9 players or every 660ms (33 cycles) for a team of 11 players.

Team Coordination

- Dynamic determination of individual behaviors for each agent.

Team Coordination

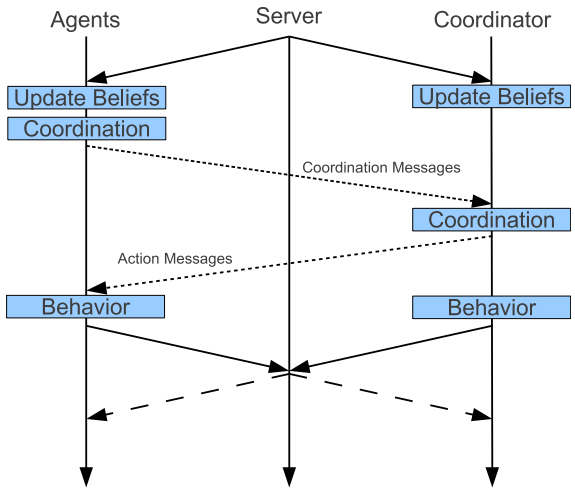
- Dynamic determination of individual behaviors for each agent.
- Executed only by one agent, the coordinator.

Team Coordination

- Dynamic determination of individual behaviors for each agent.
- Executed only by one agent, the coordinator.
- All players communicate their beliefs to the coordinator.

Team Coordination

- Dynamic determination of individual behaviors for each agent.
- Executed only by one agent, the coordinator.
- All players communicate their beliefs to the coordinator.
- Coordinator sends back to the players the computed actions.



Coordination Phases

- Update Coordination Beliefs

Coordination Phases

- Update Coordination Beliefs
- Determine Coordination Subsets

Coordination Phases

- Update Coordination Beliefs
- Determine Coordination Subsets
  - *Goalkeeper*: one player, the goalkeeper

Coordination Phases

- Update Coordination Beliefs
- Determine Coordination Subsets
  - *Goalkeeper*: one player, the goalkeeper
  - *Inactive*: players fallen on the ground or players with lost self-location

Coordination Phases

- Update Coordination Beliefs
- Determine Coordination Subsets
  - *Goalkeeper*: one player, the goalkeeper
  - *Inactive*: players fallen on the ground or players with lost self-location
  - *Active*: three players, the ones closest to the ball

Coordination Phases

- Update Coordination Beliefs
- Determine Coordination Subsets
  - *Goalkeeper*: one player, the goalkeeper
  - *Inactive*: players fallen on the ground or players with lost self-location
  - *Active*: three players, the ones closest to the ball
  - *Support*: all remaining players

Coordination Phases

- Update Coordination Beliefs
- Determine Coordination Subsets
  - *Goalkeeper*: one player, the goalkeeper
  - *Inactive*: players fallen on the ground or players with lost self-location
  - *Active*: three players, the ones closest to the ball
  - *Support*: all remaining players
- Determine Active Positions

Coordination Phases

- Update Coordination Beliefs
- Determine Coordination Subsets
  - *Goalkeeper*: one player, the goalkeeper
  - *Inactive*: players fallen on the ground or players with lost self-location
  - *Active*: three players, the ones closest to the ball
  - *Support*: all remaining players
- Determine Active Positions
- Coordinate Active Players

Coordination Phases

- Update Coordination Beliefs
- Determine Coordination Subsets
  - *Goalkeeper*: one player, the goalkeeper
  - *Inactive*: players fallen on the ground or players with lost self-location
  - *Active*: three players, the ones closest to the ball
  - *Support*: all remaining players
- Determine Active Positions
- Coordinate Active Players
- Generate Team Formation

Coordination Phases

- Update Coordination Beliefs
- Determine Coordination Subsets
  - *Goalkeeper*: one player, the goalkeeper
  - *Inactive*: players fallen on the ground or players with lost self-location
  - *Active*: three players, the ones closest to the ball
  - *Support*: all remaining players
- Determine Active Positions
- Coordinate Active Players
- Generate Team Formation
- Assign Team Roles

Coordination Phases

- Update Coordination Beliefs
- Determine Coordination Subsets
  - *Goalkeeper*: one player, the goalkeeper
  - *Inactive*: players fallen on the ground or players with lost self-location
  - *Active*: three players, the ones closest to the ball
  - *Support*: all remaining players
- Determine Active Positions
- Coordinate Active Players
- Generate Team Formation
- Assign Team Roles
- Determine Support Positions

Coordination Phases

- Update Coordination Beliefs
- Determine Coordination Subsets
  - *Goalkeeper*: one player, the goalkeeper
  - *Inactive*: players fallen on the ground or players with lost self-location
  - *Active*: three players, the ones closest to the ball
  - *Support*: all remaining players
- Determine Active Positions
- Coordinate Active Players
- Generate Team Formation
- Assign Team Roles
- Determine Support Positions
- Coordinate Support Players

Coordination Modes

Coordination is not a static procedure and may change dynamically during different game states. There are three modes of team coordination:

- Active Mode

Coordination Modes

Coordination is not a static procedure and may change dynamically during different game states. There are three modes of team coordination:

- Active Mode
- Support Mode

Coordination Modes

Coordination is not a static procedure and may change dynamically during different game states. There are three modes of team coordination:

- Active Mode
- Support Mode
- Wait Mode

Message Types and Communication

There are several types of messages, each one of them having different functionality and serving a specific purpose.

- Init Message

Message Types and Communication

There are several types of messages, each one of them having different functionality and serving a specific purpose.

- Init Message
- Start Message

Message Types and Communication

There are several types of messages, each one of them having different functionality and serving a specific purpose.

- Init Message
- Start Message
- Coordination Message

Message Types and Communication

There are several types of messages, each one of them having different functionality and serving a specific purpose.

- Init Message
- Start Message
- Coordination Message
  - Type C, position, ball position.

Message Types and Communication

There are several types of messages, each one of them having different functionality and serving a specific purpose.

- Init Message
- Start Message
- Coordination Message
  - Type C, position, ball position.
  - Type L, position.

Message Types and Communication

There are several types of messages, each one of them having different functionality and serving a specific purpose.

- Init Message
- Start Message
- Coordination Message
  - Type C, position, ball position.
  - Type L, position.
  - Type B, ball position in relation to body.

Message Types and Communication

There are several types of messages, each one of them having different functionality and serving a specific purpose.

- Init Message
- Start Message
- Coordination Message
  - Type C, position, ball position.
  - Type L, position.
  - Type B, ball position in relation to body.
  - Type X, empty messages.

Message Types and Communication

There are several types of messages, each one of them having different functionality and serving a specific purpose.

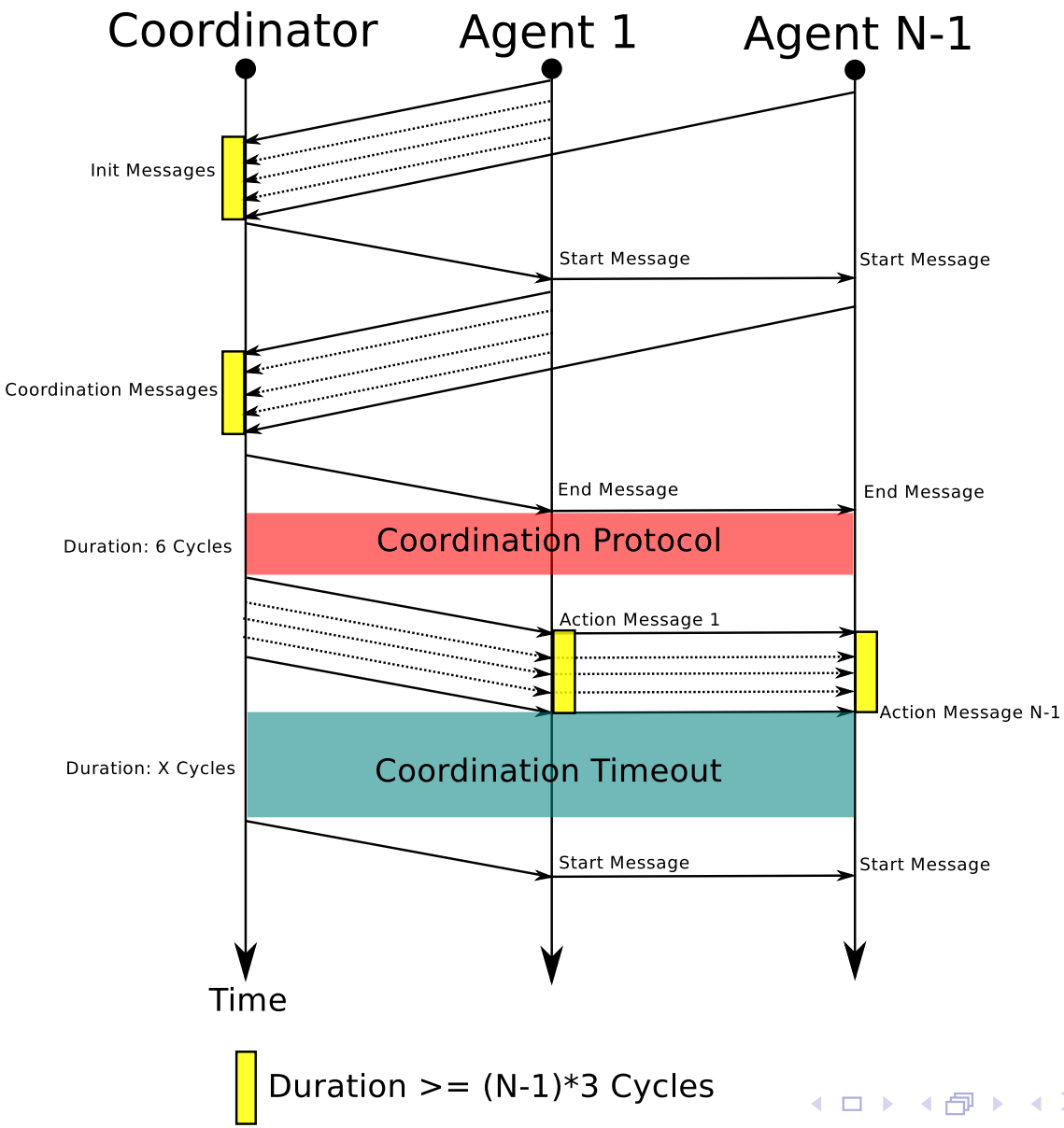
- Init Message
- Start Message
- Coordination Message
  - Type C, position, ball position.
  - Type L, position.
  - Type B, ball position in relation to body.
  - Type X, empty messages.
- End Message

Message Types and Communication

There are several types of messages, each one of them having different functionality and serving a specific purpose.

- Init Message
- Start Message
- Coordination Message
  - Type C, position, ball position.
  - Type L, position.
  - Type B, ball position in relation to body.
  - Type X, empty messages.
- End Message
- Action Message

Messaging Protocol



Coordination Beliefs

Coordination need to have a good knowledge about the world state.

- Global ball position

Coordination Beliefs

Coordination need to have a good knowledge about the world state.

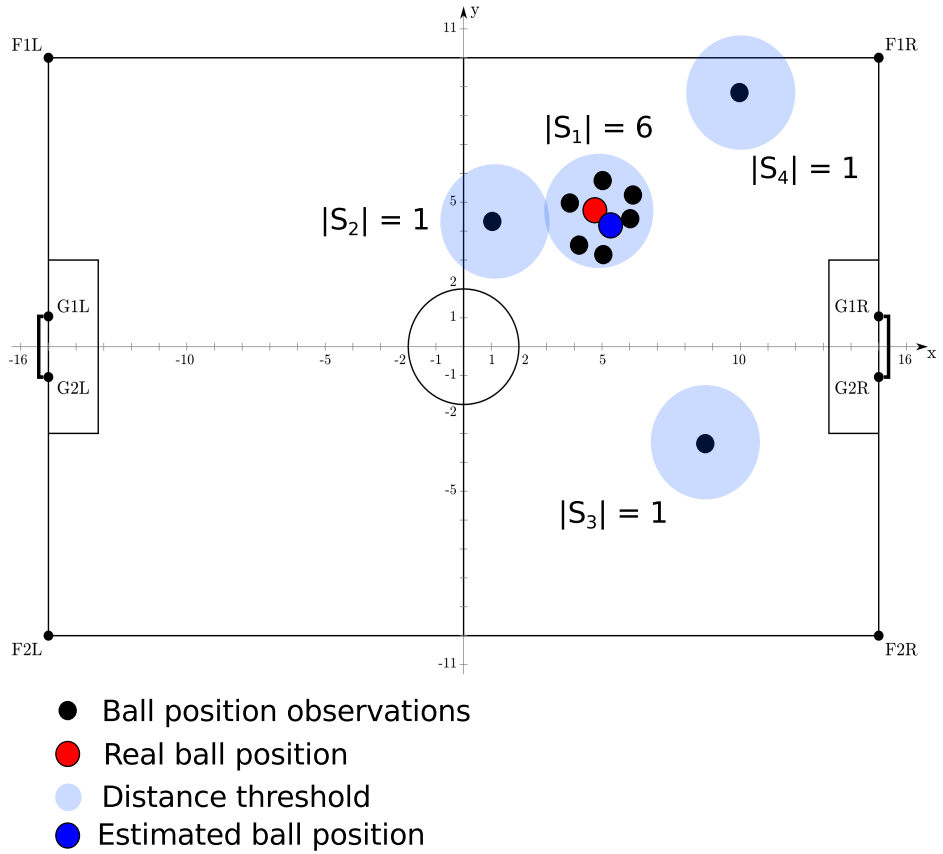
- Global ball position
- Agents' distances from ball

Coordination Beliefs

Coordination need to have a good knowledge about the world state.

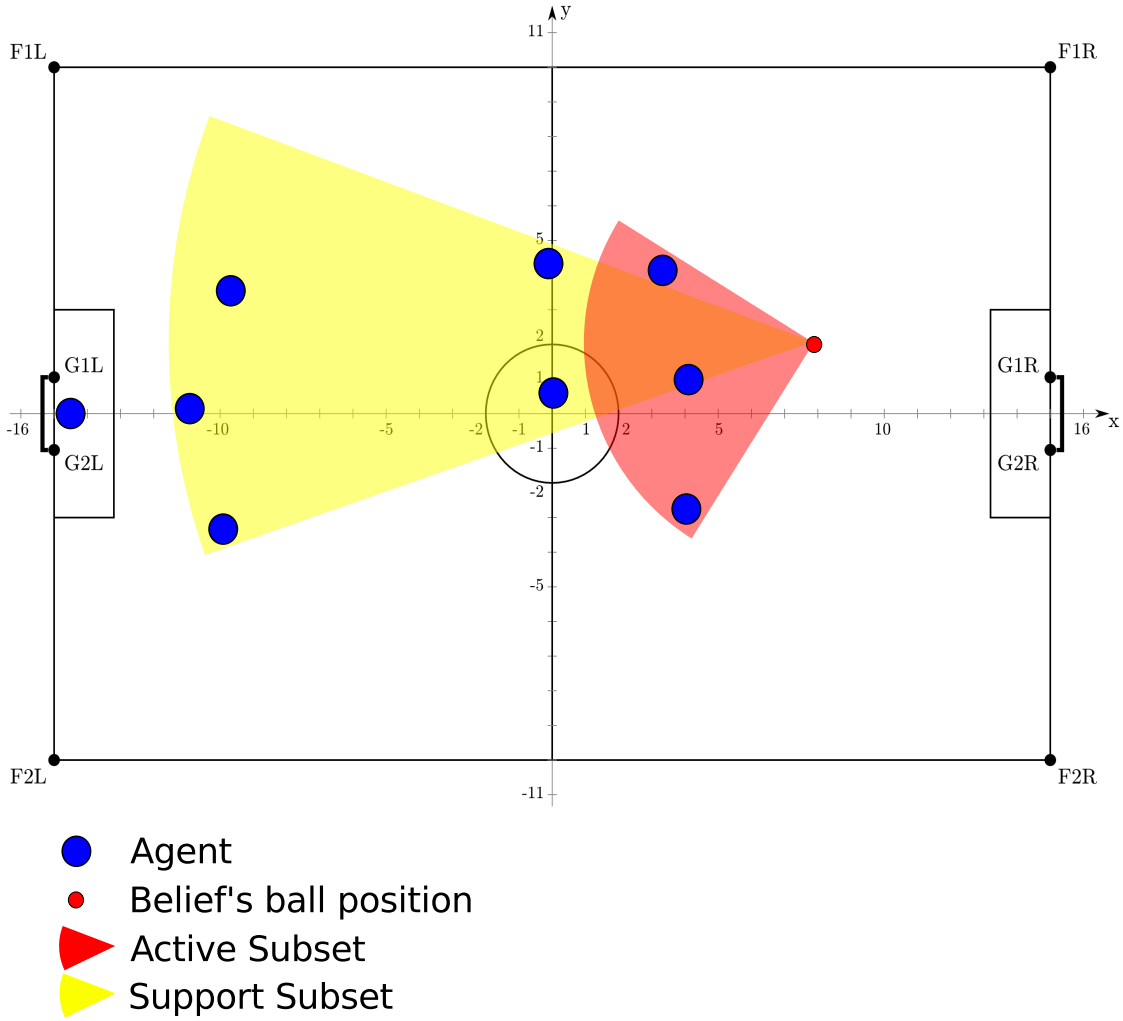
- Global ball position
- Agents' distances from ball
- Agents' positions

Estimated Ball Position

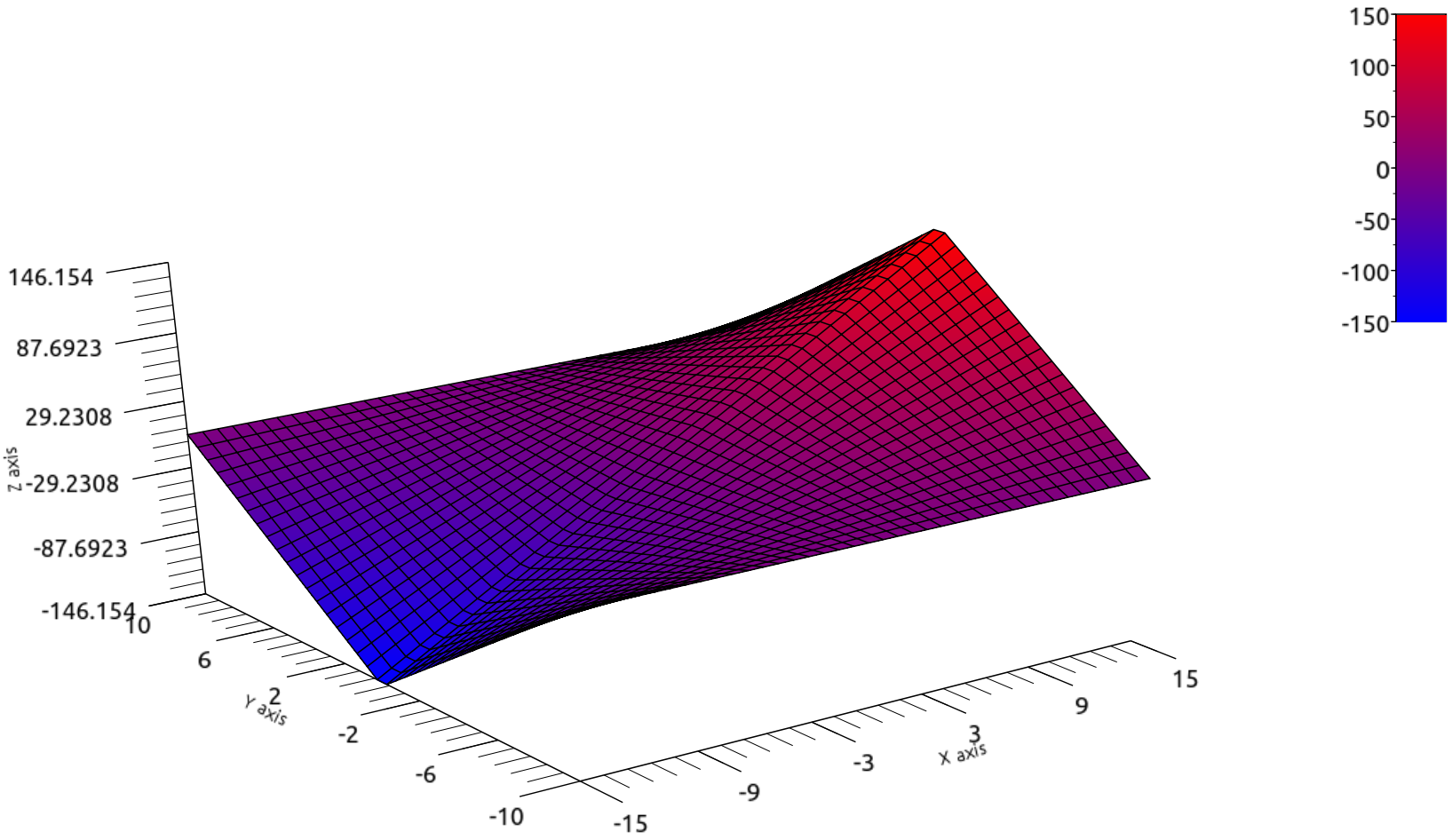


$$\textbf{GlobalBallBelief} = \sum_{i=1}^{m} \frac{w(s_i)}{\sum_{k=1}^{m} w(s_k)} \sum_{o_{ij} \in s_i} \frac{o_{ij}}{|s_i|}$$

Coordination Splitter

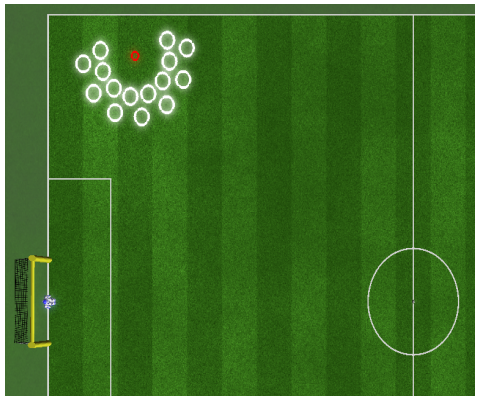


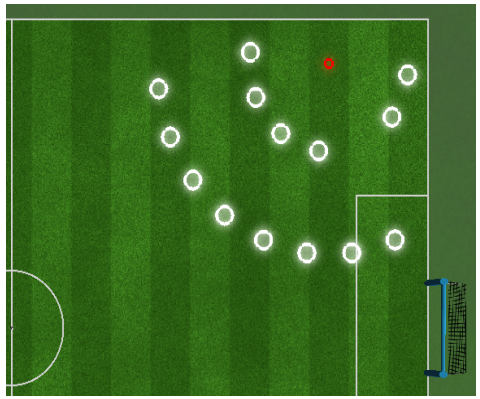
Soccer Field Utility Fuction



Determination of Active Positions

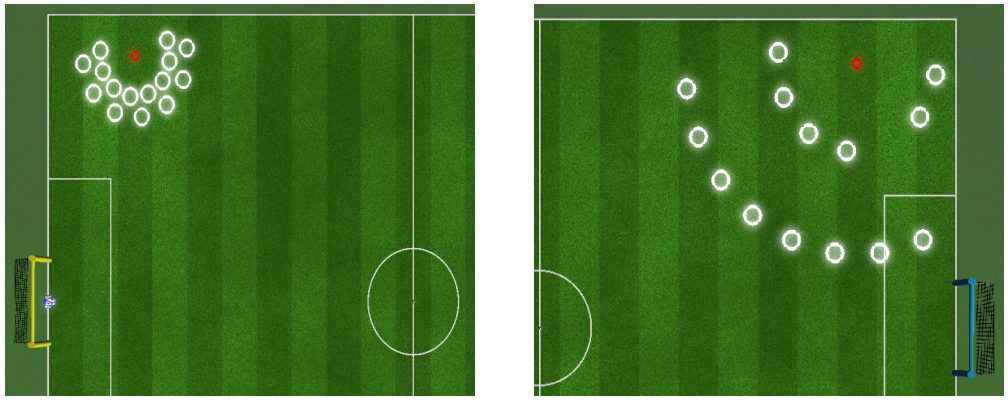
- Dynamic production of active positions.





Determination of Active Positions

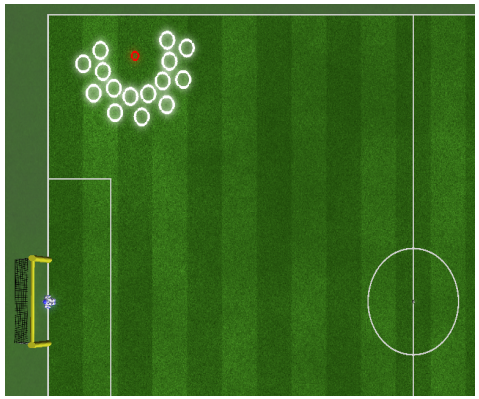
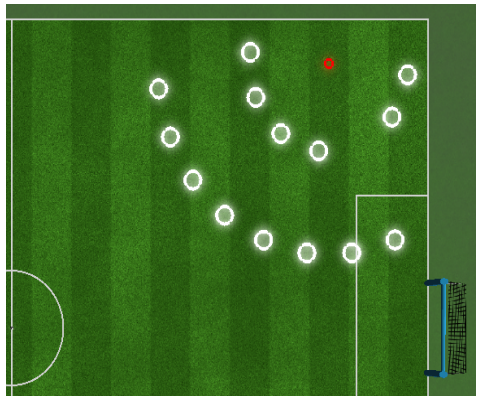
- Dynamic production of active positions.



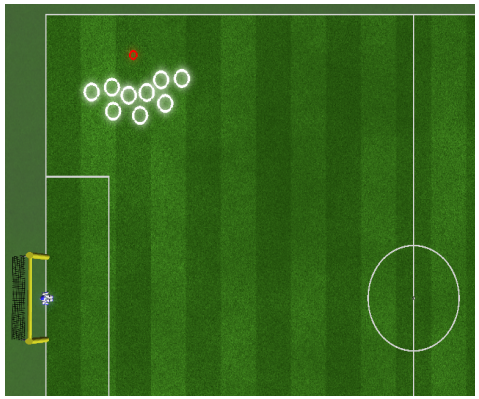
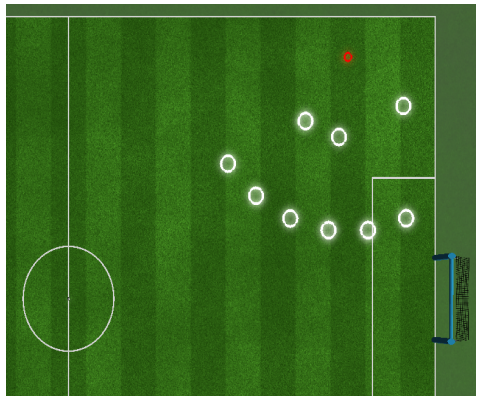
- Positions elimination making use of their utility field value.

Determination of Active Positions

- Dynamic production of active positions.

- Positions elimination making use of their utility field value.
- Max number of 9 positions.

Active Players Coordination

- On Ball player.

Active Players Coordination

- On Ball player.
- Agent closest to the ball.

Active Players Coordination

- On Ball player.
- Agent closest to the ball.
- Angle from goal is also taken into consideration.

Active Players Coordination

- On Ball player.
- Agent closest to the ball.
- Angle from goal is also taken into consideration.
- Set of ≤ 9 positions.

Active Players Coordination

- On Ball player.
- Agent closest to the ball.
- Angle from goal is also taken into consideration.
- Set of ≤ 9 positions.
- Set of ≤ 2 agents.

Active Players Coordination

- On Ball player.
- Agent closest to the ball.
- Angle from goal is also taken into consideration.
- Set of ≤ 9 positions.
- Set of ≤ 2 agents.
- Exhaustive algorithm.

Active Players Coordination

- On Ball player.
- Agent closest to the ball.
- Angle from goal is also taken into consideration.
- Set of ≤ 9 positions.
- Set of ≤ 2 agents.
- Exhaustive algorithm.
- Every combination of mappings is checked.

Active Players Coordination

- On Ball player.
- Agent closest to the ball.
- Angle from goal is also taken into consideration.
- Set of ≤ 9 positions.
- Set of ≤ 2 agents.
- Exhaustive algorithm.
- Every combination of mappings is checked.
- Cost function to determine which of them is optimal.

Active Coordination Evaluation Function

The evaluation function scores each possible mapping using the following features defined for each agent i:

1. **Distance** $C_{d,i}$
2. **Potential Collisions** $C_{c,i}$
3. **Field Utility** $C_{u,i}$
4. **Close Targets** $C_{t,i}$
5. **Horizontal Stretch** $C_{h,i}$

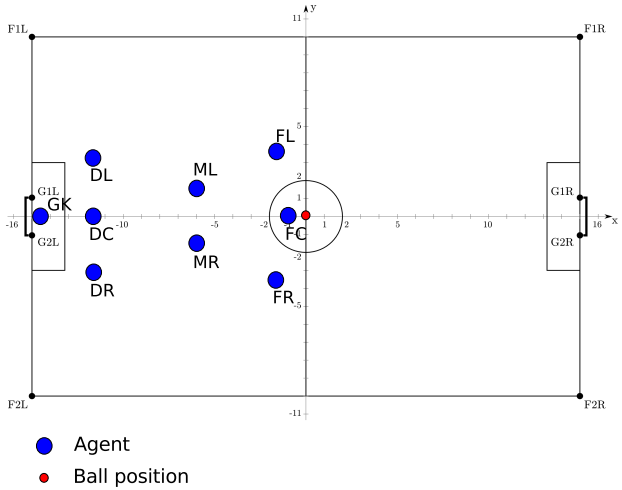
$$ActiveCost(ActiveMapping) = \sum_{i=1}^{3} w_d C_{d,i} + w_c C_{c,i} - w_u C_{u,i} - w_t C_{t,i} - w_h C_h$$

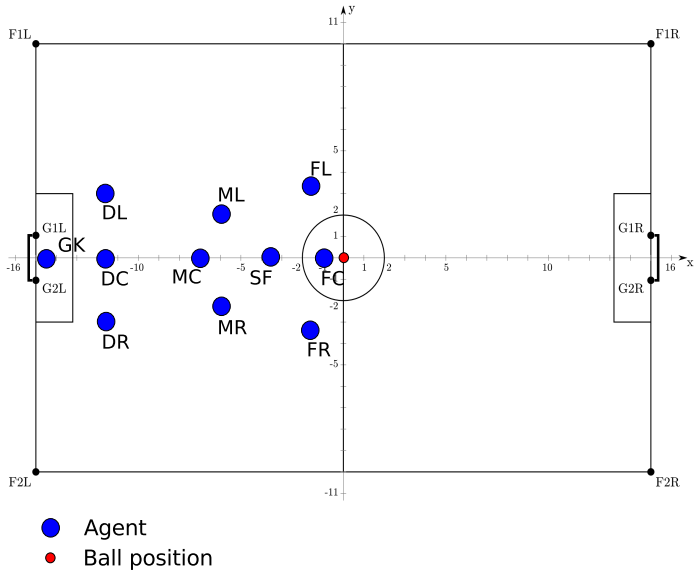
where $(w_d, w_c, w_u, w_t, w_h)$ are the weights of the features, currently set at $(1, 1, 1/7, 1, 1)$.

Generation of Team Formation

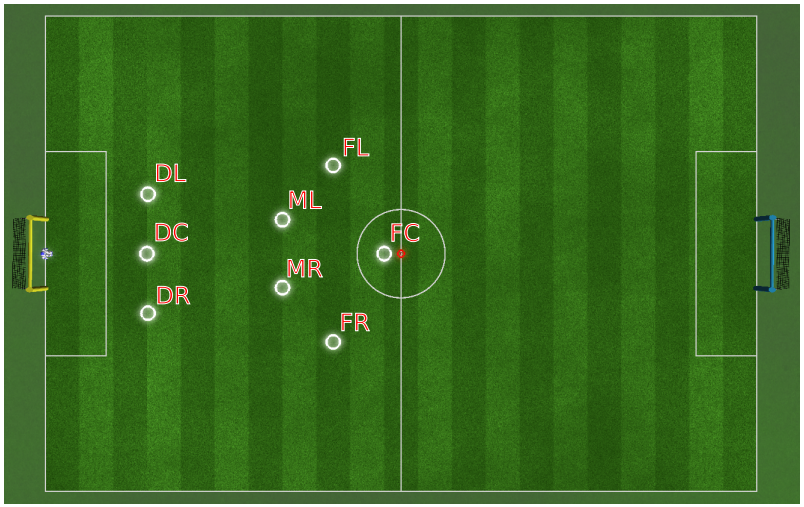
Serves to set up the role assignment function and the coordination of the support subset.

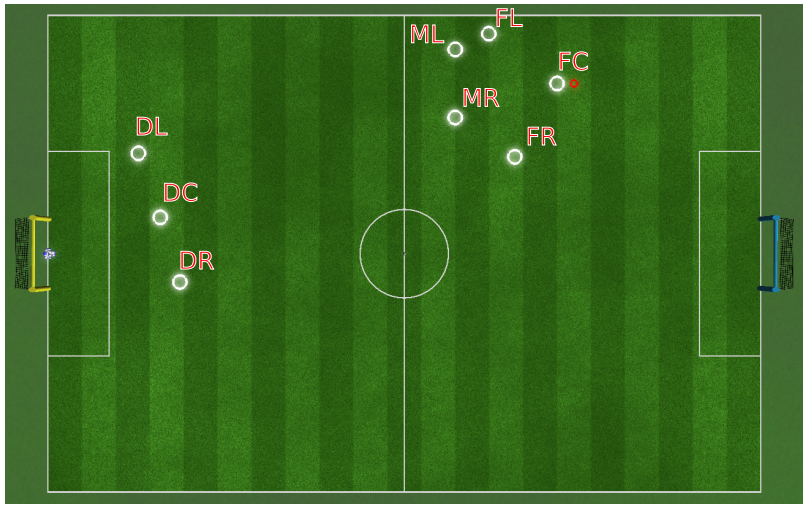
- 9 Players Version.
- 11 Players Version.

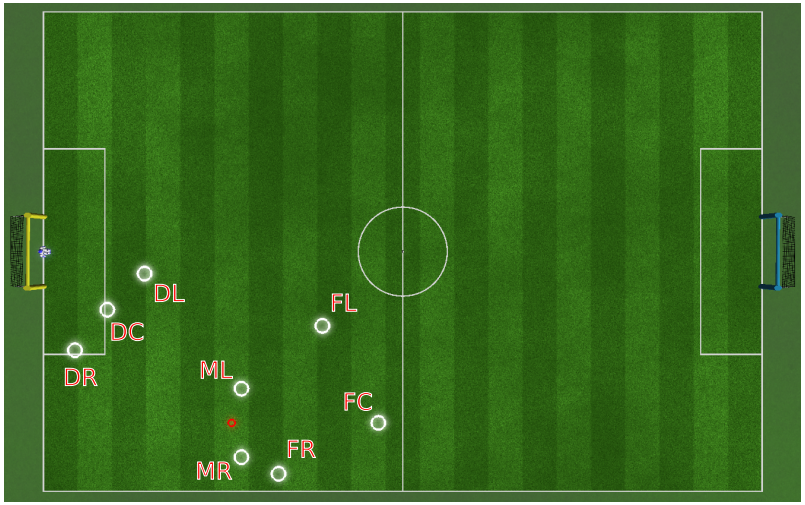




Team Formation Examples







Team Roles Assignment

- Assigns roles to all agents.

Team Roles Assignment

- Assigns roles to all agents.
- Roles close to ball, active players.

Team Roles Assignment

- Assigns roles to all agents.
- Roles close to ball, active players.
- Remaining roles will be assigned to support players.

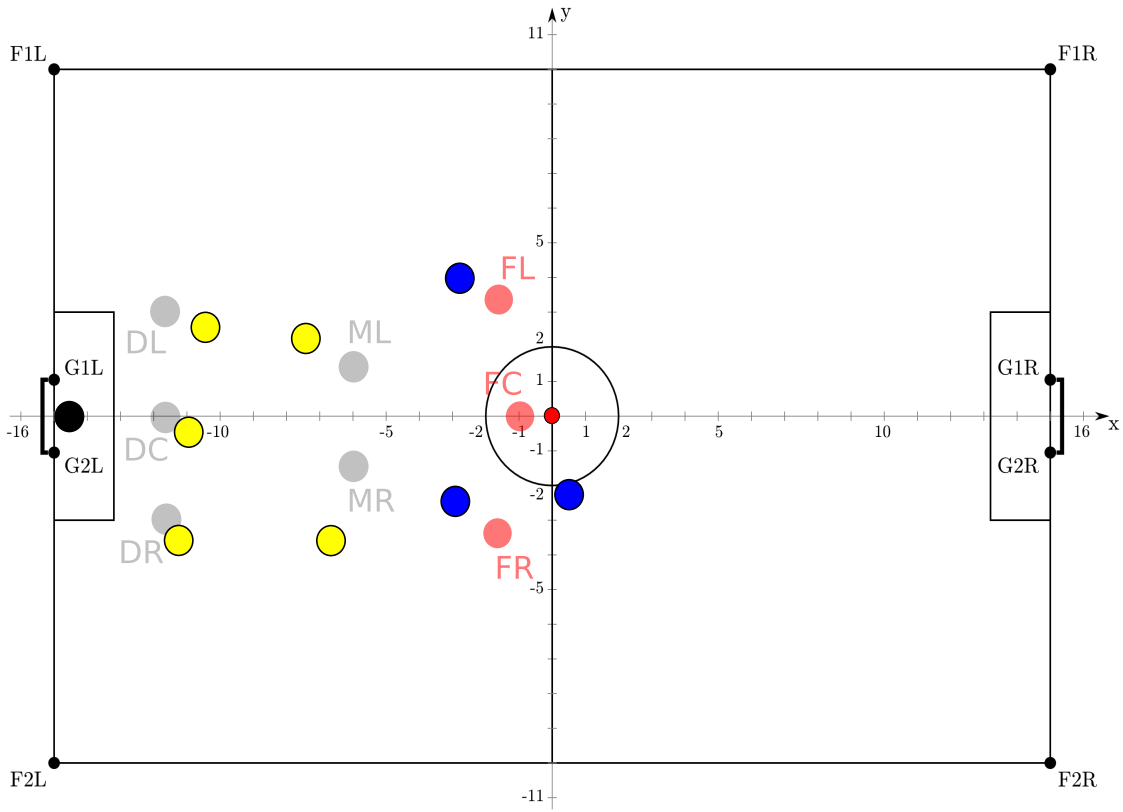
Team Roles Assignment

- Assigns roles to all agents.
- Roles close to ball, active players.
- Remaining roles will be assigned to support players.
- Inactive agents.

Team Roles Assignment

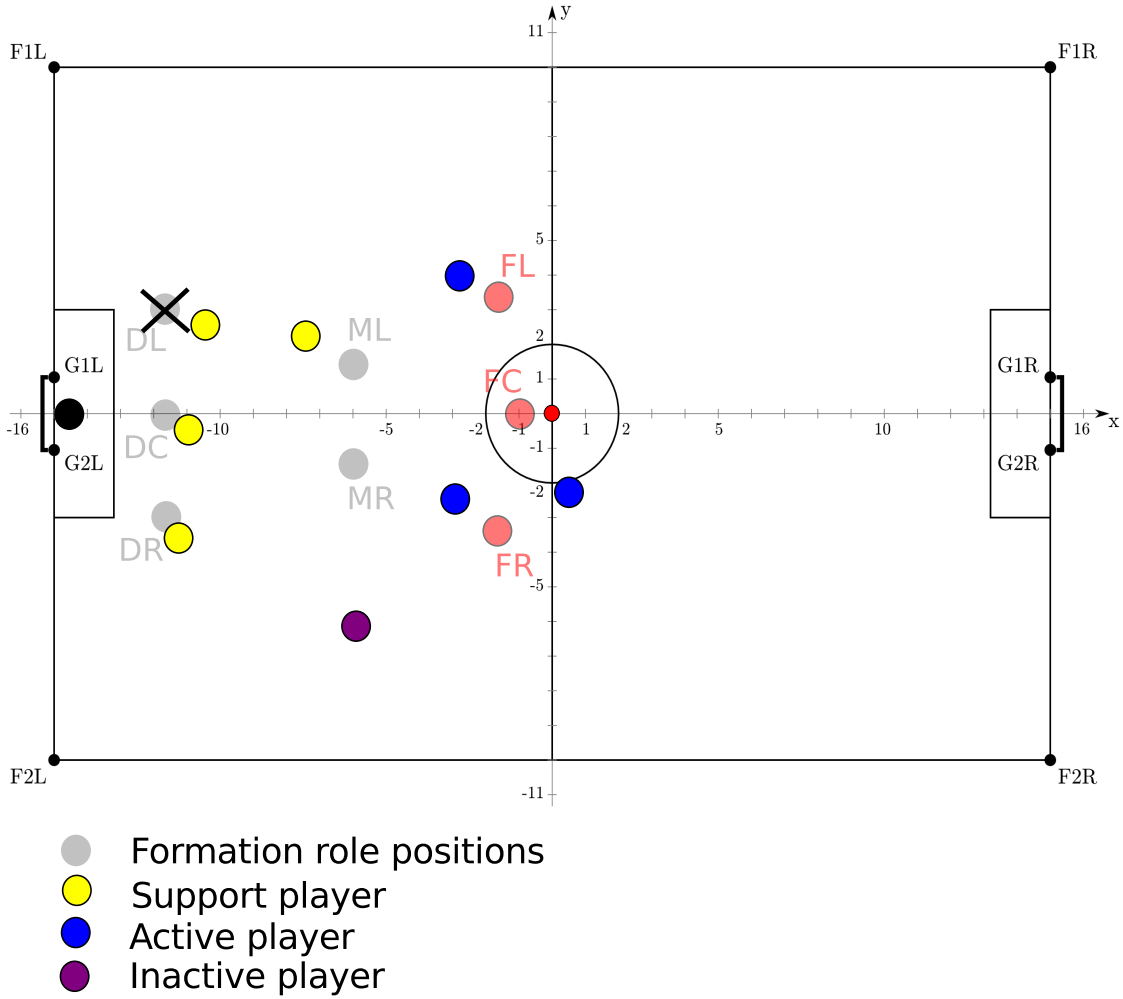
- Assigns roles to all agents.
- Roles close to ball, active players.
- Remaining roles will be assigned to support players.
- Inactive agents.
- Same number of roles will be discarded.

Team Roles Assignment Example 1



- Formation role positions
- Formation role positions near to ball
- Ball position
- Support player
- Active player

Team Roles Assignment Example 2



Formation role positions
Support player
Active player
Inactive player

Team Roles Assignment Example 1

- Same number of positions, agents.

Team Roles Assignment Example 1

- Same number of positions, agents.
- Computationally expensive problem.

Team Roles Assignment Example 1

- Same number of positions, agents.
- Computationally expensive problem.
- The exhaustive algorithm, in a worst case scenario, would have to examine between 40320 and 3628800 possible mappings.

Team Roles Assignment Example 1

- Same number of positions, agents.
- Computationally expensive problem.
- The exhaustive algorithm, in a worst case scenario, would have to examine between 40320 and 3628800 possible mappings.
- The exhaustive algorithm can work satisfactorily on support subsets of size up to 6 (720 possible mappings).

Dynamic Algorithm

- Given the difficulties of applying the exhaustive algorithm.

Dynamic Algorithm

- Given the difficulties of applying the exhaustive algorithm.
- A method proposed by the UT Austin Villa team.

Dynamic Algorithm

- Given the difficulties of applying the exhaustive algorithm.
- A method proposed by the UT Austin Villa team.
- Able to compute an approximately optimal solution.

$\{P_1\}$	$\{P_1, P_2\}$	$\{P_1, P_2, P_3\}$
$\{A_1 \leftarrow P_1\}$	$\{A_1 \leftarrow P_2\} \cup \arg\min(\{A_2\} \leftarrow \{P_1\})$	$\{A_1 \leftarrow P_3\} \cup \arg\min(\{A_2, A_3\} \leftarrow \{P_1, P_2\})$
$\{A_2 \leftarrow P_1\}$	$\{A_1 \leftarrow P_2\} \cup \arg\min(\{A_3\} \leftarrow \{P_1\})$	$\{A_2 \leftarrow P_3\} \cup \arg\min(\{A_1, A_3\} \leftarrow \{P_1, P_2\})$
$\{A_3 \leftarrow P_1\}$	$\{A_2 \leftarrow P_2\} \cup \arg\min(\{A_1\} \leftarrow \{P_1\})$	$\{A_3 \leftarrow P_3\} \cup \arg\min(\{A_1, A_2\} \leftarrow \{P_1, P_2\})$
	$\{A_2 \leftarrow P_2\} \cup \arg\min(\{A_3\} \leftarrow \{P_1\})$	
	$\{A_3 \leftarrow P_2\} \cup \arg\min(\{A_1\} \leftarrow \{P_1\})$	
	$\{A_3 \leftarrow P_2\} \cup \arg\min(\{A_2\} \leftarrow \{P_1\})$	

Dynamic Algorithm Complexity

- In kth iteration of the algorithm, each agent will be assigned to the P_k position.

Dynamic Algorithm Complexity

- In kth iteration of the algorithm, each agent will be assigned to the P_k position.
- The previous $k-1$ positions will be assigned to the other $n-1$ agents.

Dynamic Algorithm Complexity

- In kth iteration of the algorithm, each agent will be assigned to the P_k position.
- The previous $k-1$ positions will be assigned to the other $n-1$ agents.
- Result in a total of $\binom{n-1}{k-1}$ mappings to be evaluated in each iteration for each agent.

$$n\sum_{k=1}^{n}\binom{n-1}{k-1} = n\sum_{k=0}^{n-1}\binom{n-1}{k} = n2^{n-1}$$

Dynamic Algorithm Complexity

- In kth iteration of the algorithm, each agent will be assigned to the P_k position.
- The previous $k-1$ positions will be assigned to the other $n-1$ agents.
- Result in a total of $\binom{n-1}{k-1}$ mappings to be evaluated in each iteration for each agent.

$$n\sum_{k=1}^{n}\binom{n-1}{k-1} = n\sum_{k=0}^{n-1}\binom{n-1}{k} = n2^{n-1}$$

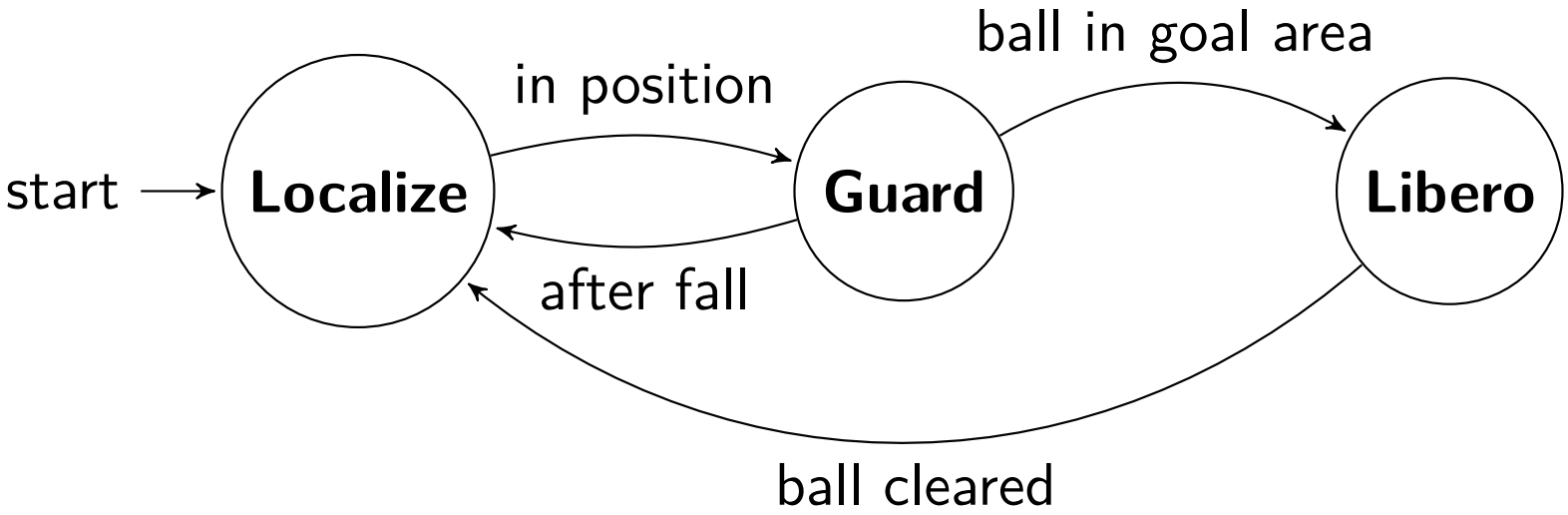
- This represents a reduction to 1024 and 5120 mappings for 8 and 10 agents/positions respectively compared to 40320 and 3628800 mappings of the exhaustive algorithm.

Dynamic Algorithm Complexity

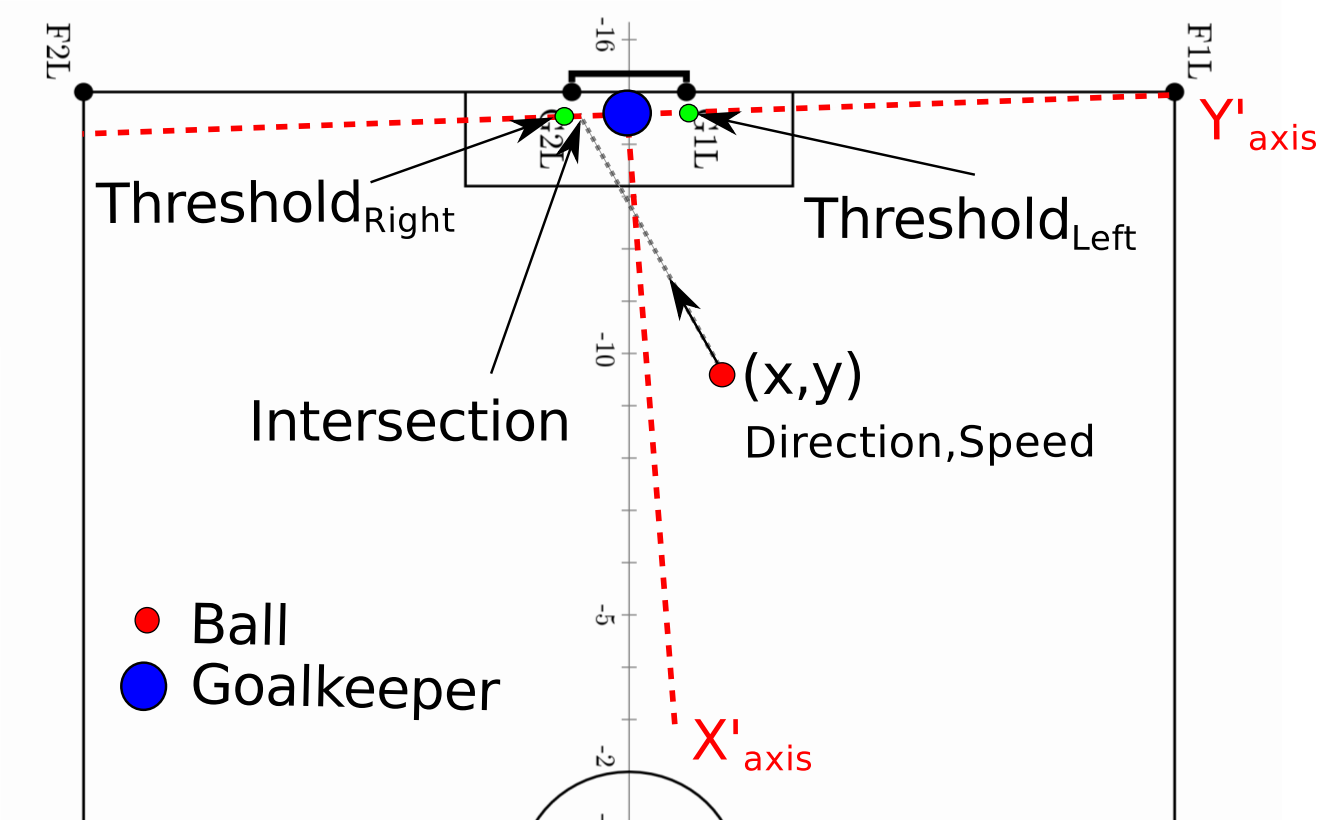
- Only agent in our team who “runs” his own behavior.

Dynamic Algorithm Complexity

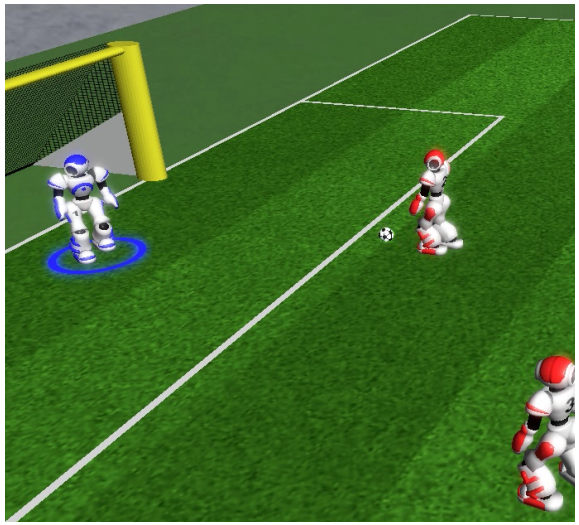
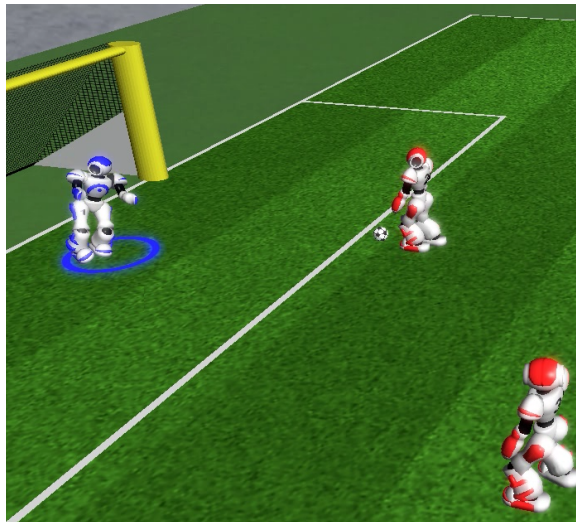
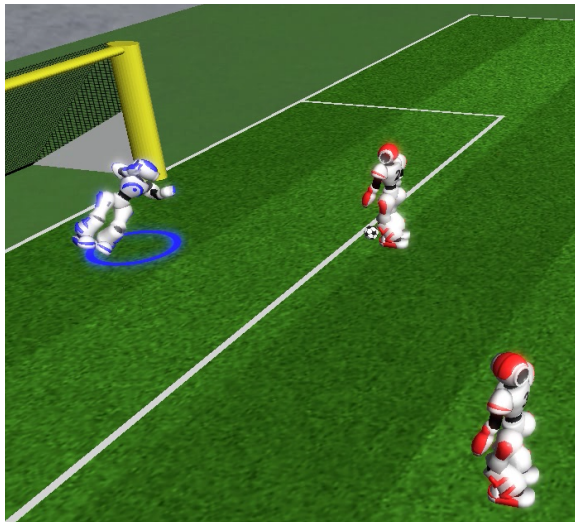
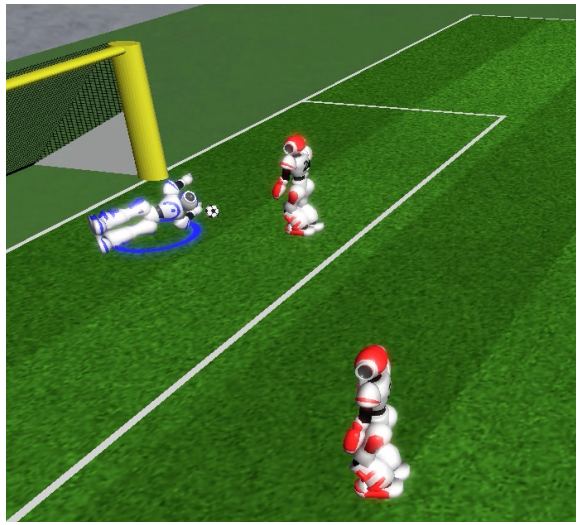
- Only agent in our team who "runs" his own behavior.
- His behavior depends on a finite state machine.



"Guard" State


F2L
F1L
-16
-10
-5
-2
G2L
G1L
Threshold Right
Threshold Left
Y' axis
(x,y)
Direction,Speed
Intersection
Ball
Goalkeeper
X' axis

Goalkeeper Fall Example

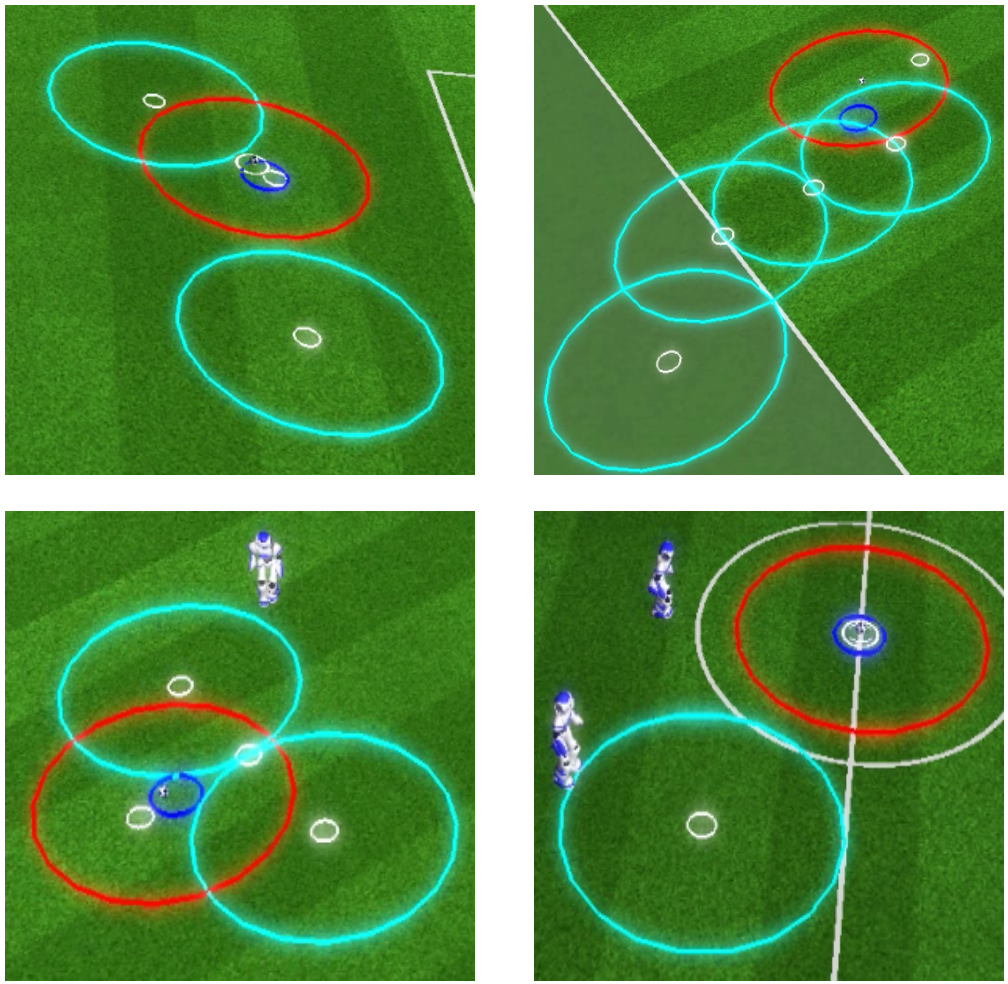
Motion Results and Improvements

Motion Version	Walk(m/s)	Turn(d/s)	Kick(m)	Strong Kick(m)
Webots (Text-Based)	0.11	21	3	-
FIIT (XML)	0.22	25	3 (4 Sec.)	4 (5 Sec.)
AST_3D	0.45	30	3 (2.5 Sec.)	5.5 (2.5 Sec.)

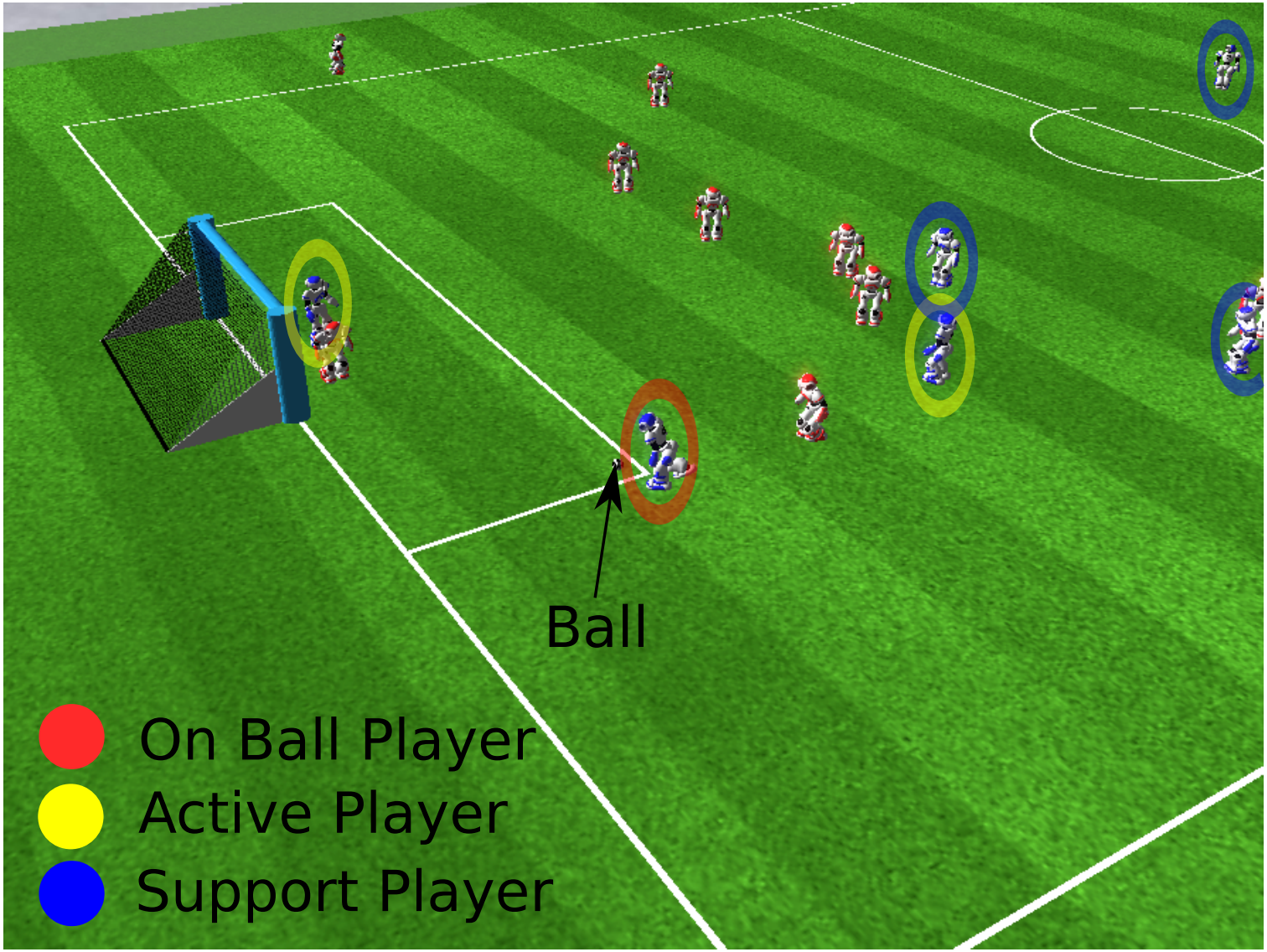
Communication Results

Communication Phase	Ideal (Cycles (Sec.))	During Match (Cycles (Sec.))
Init Messages	24 (0.48)	24 (0.48)
Coordination Messages	24 (0.48)	42.5 (0.85)
Action Messages	24 (0.48)	24 (0.48)

Coordination Beliefs (Global Ball Position)

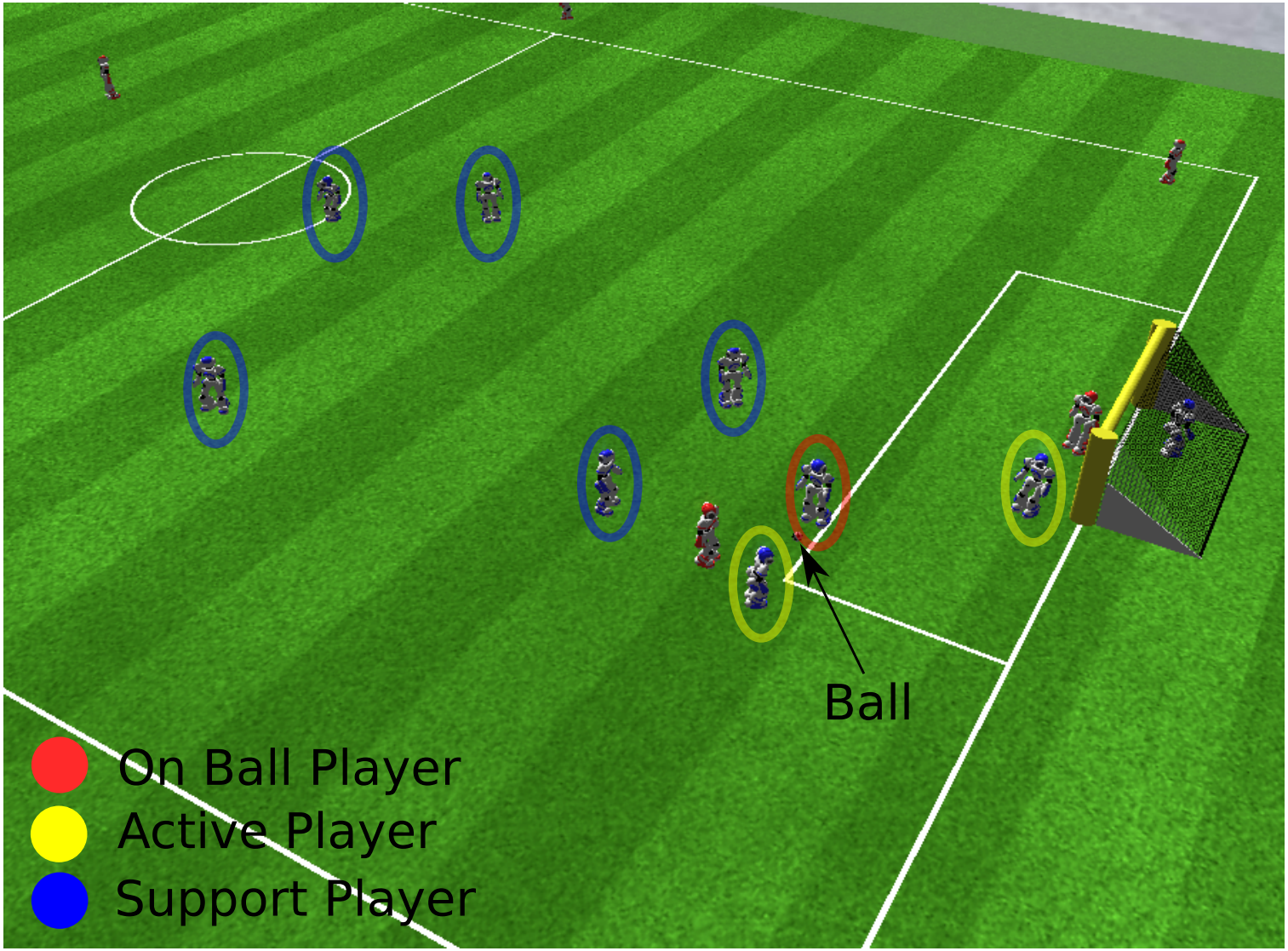


Offensive Positioning 1



Formation Consistency

Defensive Positioning 1



Defensive Positioning 2

Offensive Positioning 2

Goalkeeper Results

GoalKeeper Type	Goals Conceded
No Goalkeeper	~ 7
Goalkeeper with "Empty" Behavior	~ 7
Goalkeeper with "Full" Behavior	~ 3

Full-Games Results

Team	W	D	L	AGD	Games
MAK	2	0	0	+2.0	2
L3M-SIM	3	2	0	+0.6	5
FARZANEGAN	1	1	0	+0.5	2
NomoFC	1	2	0	+0.3	3
Rail	0	4	0	0.0	4
FUTK3D	0	5	0	0.0	5
OxBlue	0	0	2	-1.5	2
BeeStanbul	0	0	3	-4.0	3
UTAustinVilla	0	0	4	-5.2	4
Robocanes	0	0	1	-6.0	1

Full-Games Results

Team	P	W	D	L	F	A	Pts
AST3D	**8**	**2**	**6**	**0**	**2**	**0**	**12**
NomoFC	2	0	1	1	0	1	1
L3M-SIM	2	0	1	1	0	1	1
Rail	2	0	2	0	0	0	2
farzanegan	2	0	2	0	0	0	2

Future Work

- Probabilistic localization system

Future Work

- Probabilistic localization system
- Dynamic Omni-Directional locomotiom

Future Work

- Probabilistic localization system
- Dynamic Omni-Directional locomotiom
- Passing

Future Work

- Probabilistic localization system
- Dynamic Omni-Directional locomotiom
- Passing
- Participation in Robocup

Finally...

Ευχαριστώ πολύ!