CAMBADA's New Agent Architecture and Behaviours

Ricardo Dias ricardodias@ua.pt

Bernardo Cunha mbc@det.ua.pt

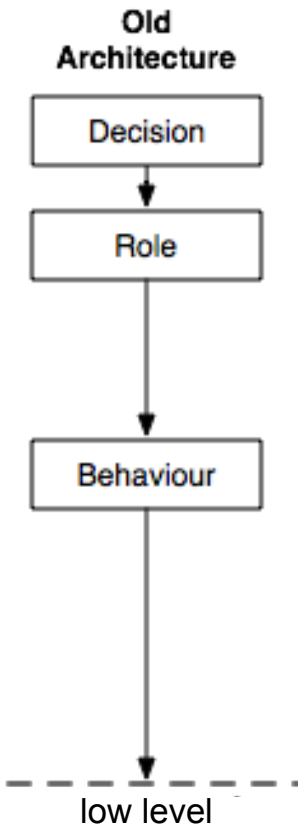
RoboCup MSL Workshop 2014 - Eindhoven, The Netherlands

- Introduction
- New Agent Architecture
- Developed Behaviors
- Heightmap Integration
- Results
- Conclusions

- **Introduction**
- New Agent Architecture
- Developed Behaviors
- Heightmap Integration
- Results
- Conclusions

Introduction

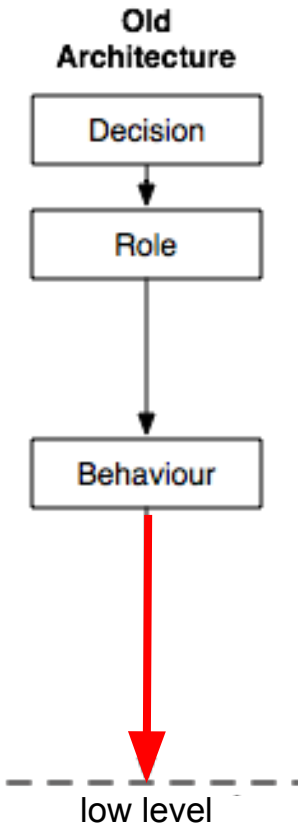
Execution Flow



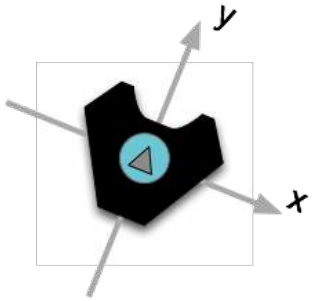
- `Behaviours`
  - Execute a specific task
  - Ex.: `Move`, `Dribble`, `Kick`, `Pass`, ...
- `Roles`
  - Play a role in the game
  - Ex.: `Striker`, `Midfielder`, `Receiver`, ...
  - They instantiate behaviors (FSM)
- `Decision`
  - Instantiate a `Role` each cycle

Introduction

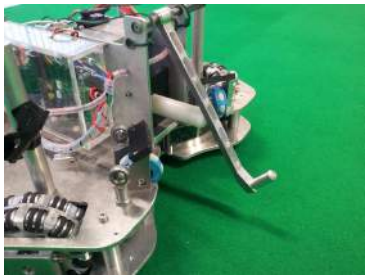
Execution Flow



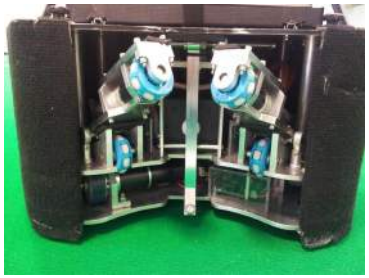
Decision Layer Outputs



`velX, velY, velA`



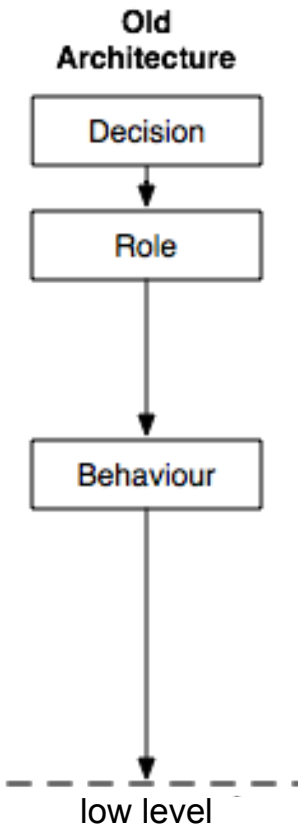
`kickPower`



`grabberState`

Introduction

Execution Flow



2 Major Problems

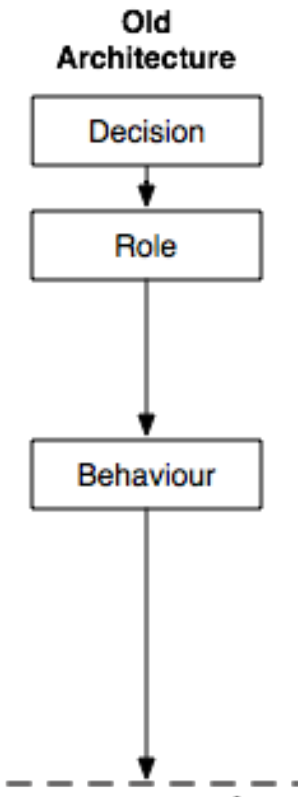
History Loss
(`Roles` and `Behaviors`)

High Code Complexity
(`Roles`)

- Introduction
- **New Agent Architecture**
- Developed Behaviors
- Heightmap Integration
- Results
- Conclusions

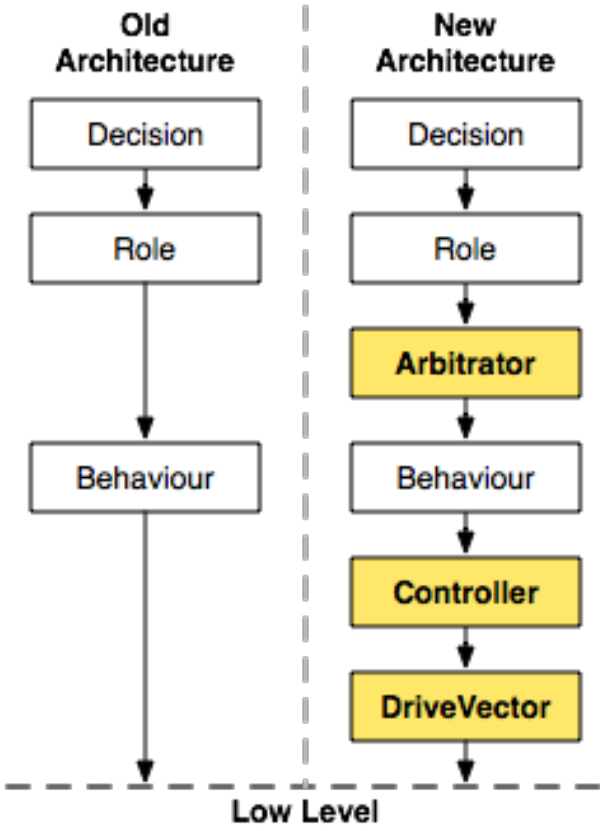
New Agent Architecture

Execution Flow



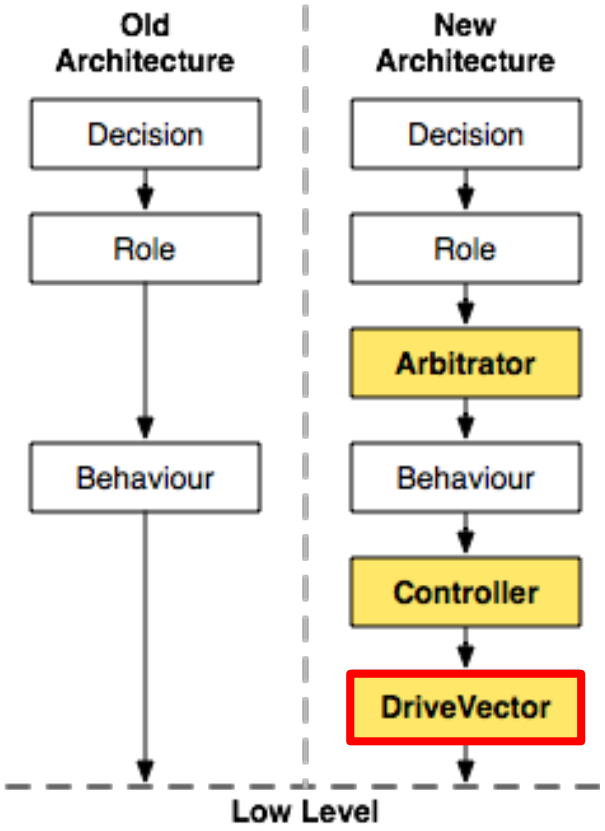
New Agent Architecture

Execution Flow

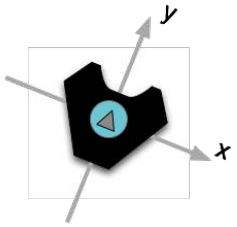


New Agent Architecture

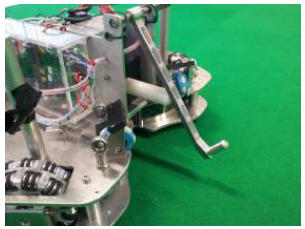
Execution Flow



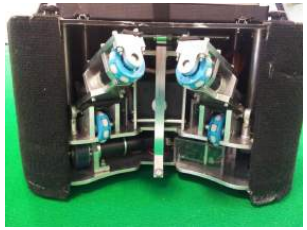
Decision Layer Outputs



`velX,velY,velA`



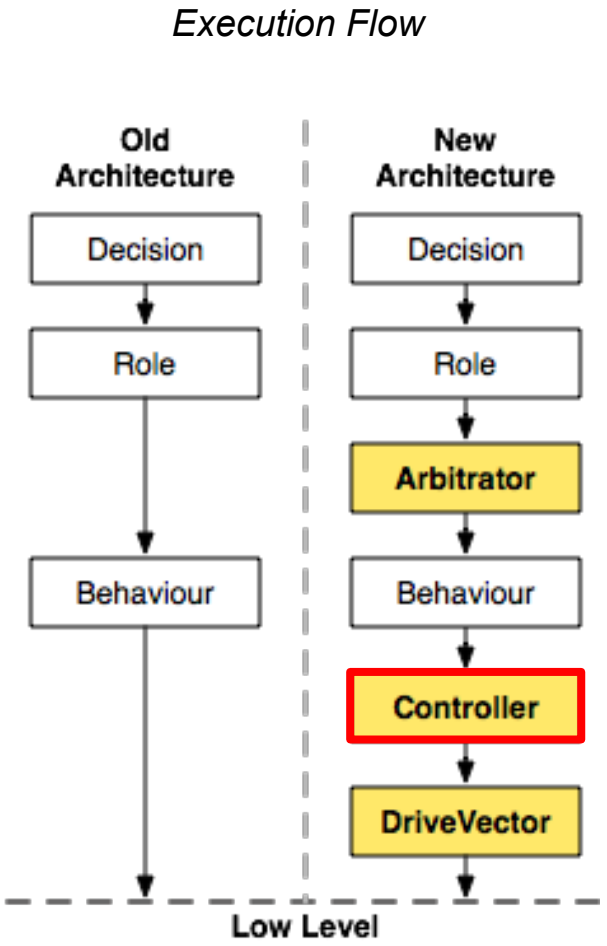
`kickPower`



`grabberState`

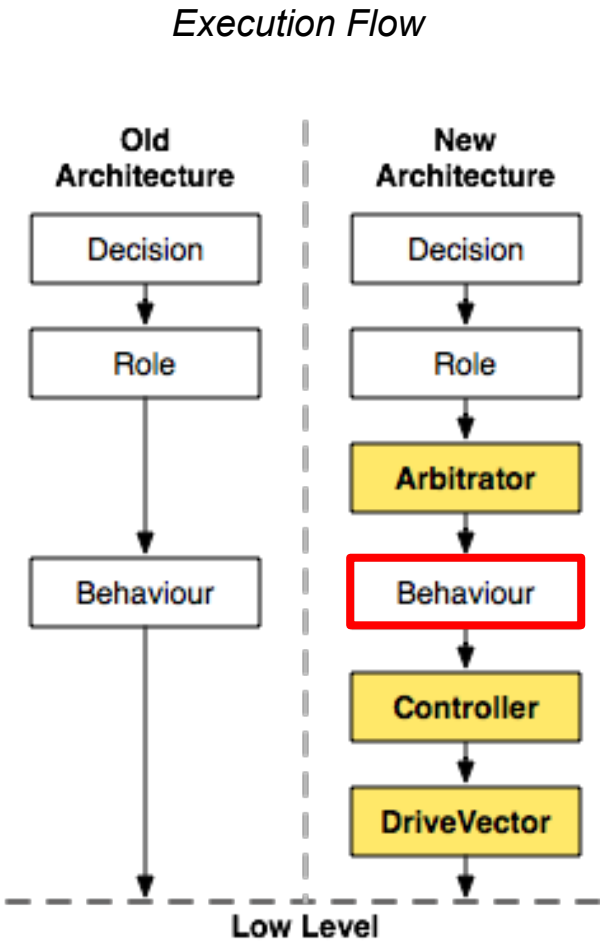
```
allOff()
motorsOff()
pass()
kick()
...
```

New Agent Architecture



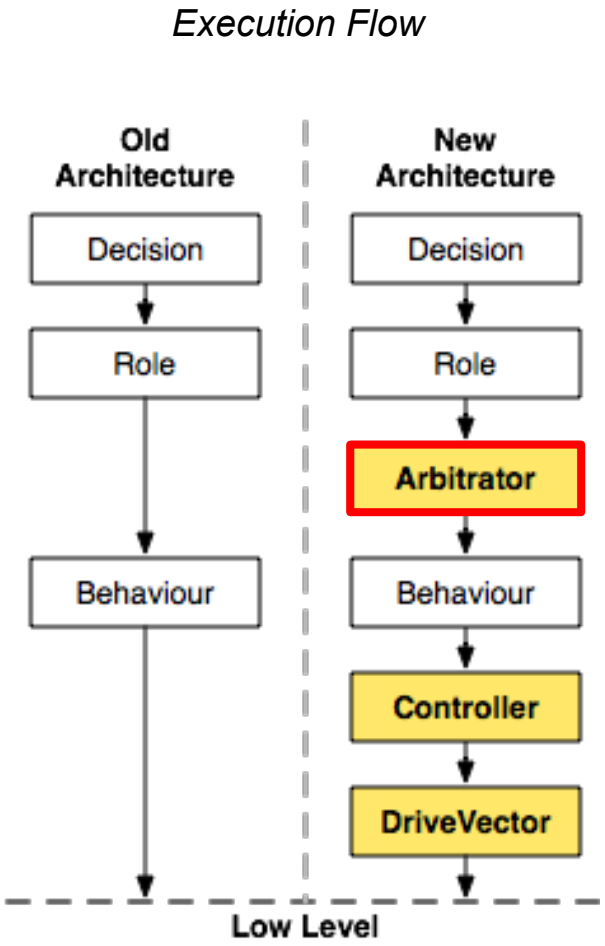
- **`Controller`**
  - Input: target (pos, ori)
  - PID
  - Calculates X, Y, Ang velocities
    - `cMove`
    - `cArc`
    - `cRotateAroundTheBall`
    - `cRotate`

New Agent Architecture



- **`Behaviour`**
  - Followed the ideas from the **Brainstormers Tribots** team
  - Interface was extended:
    - `InvocationCondition`
    - `CommitmentCondition`

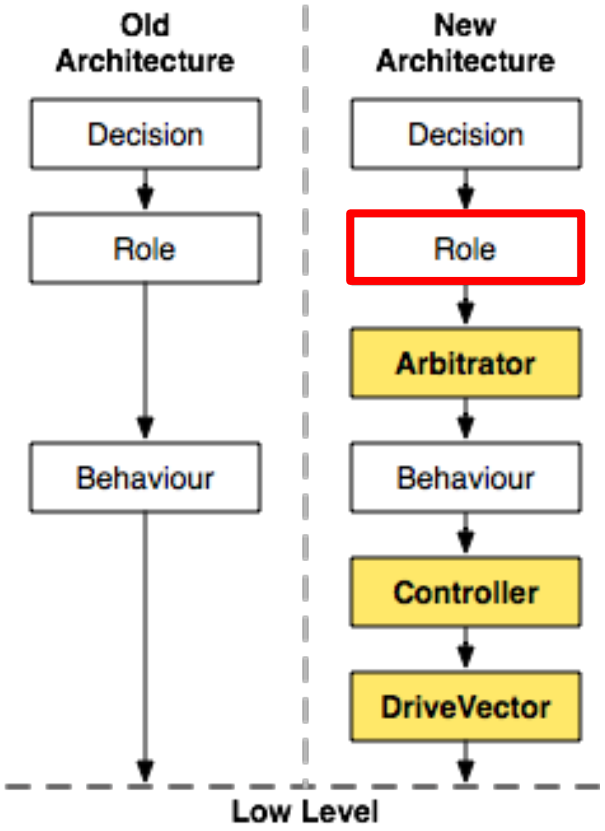
New Agent Architecture



- **`Arbitrator`**
  - Input: list of possible `Behaviours`
  - Typically **Priority Arbitration**

New Agent Architecture

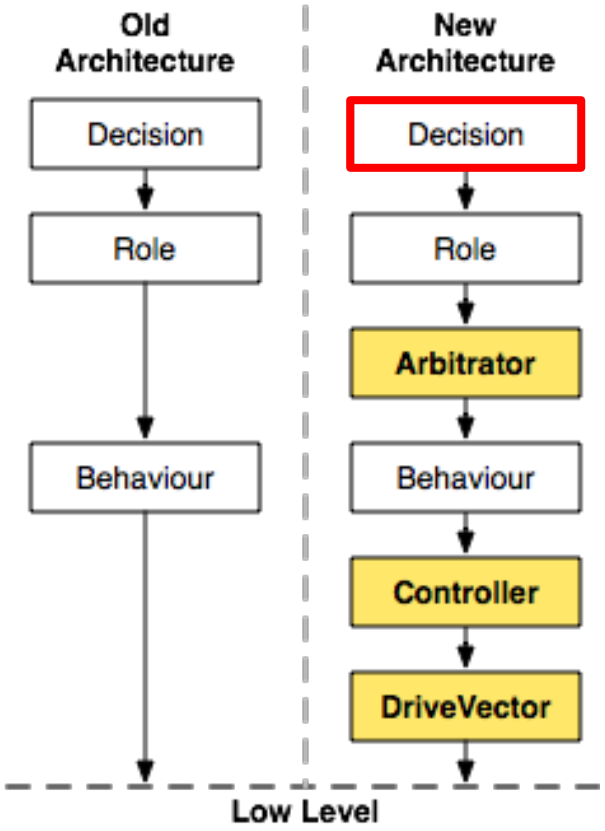
Execution Flow



- **`Role`**
  - Contains an `Arbitrator`
  - `Behaviours` are added in the `Role` constructor

New Agent Architecture

Execution Flow

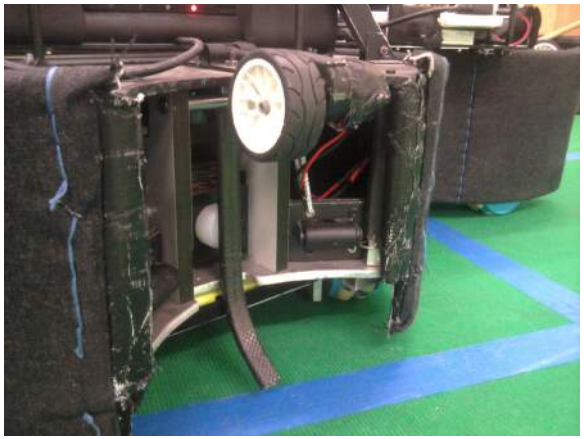


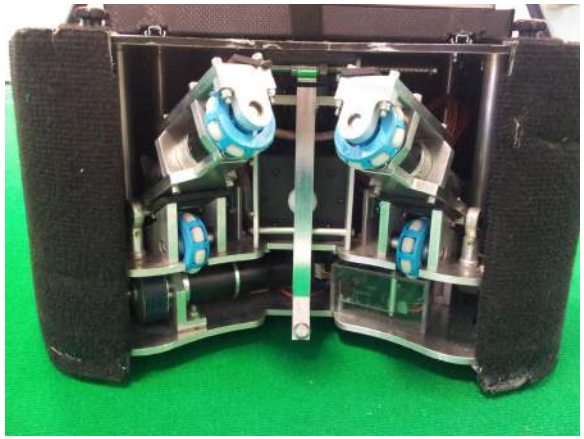
- **`Decision`**
  - `Roles` instantiated in the `Decision` constructor
  - The `Roles` are selected instead of instantiated

- Introduction
- New Agent Architecture
- **Developed Behaviors**
- Heightmap Integration
- Results
- Conclusions

Ball Passes

- Real coordination challenge
- New algorithm takes advantage of:
  - The new platform grabbing system

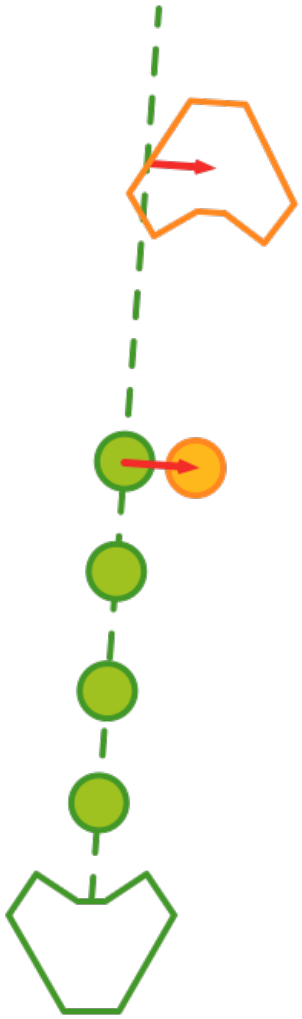




  - Robot-robot communications

Ball Passes

- `CoordinationFlag`
  - *LineClear*, *NotClear*
  - *TryPass*, *BallPassed*
- `CoordinationVec`
  - 2D Position in the world
  - Forward Passes
- `PassLine`



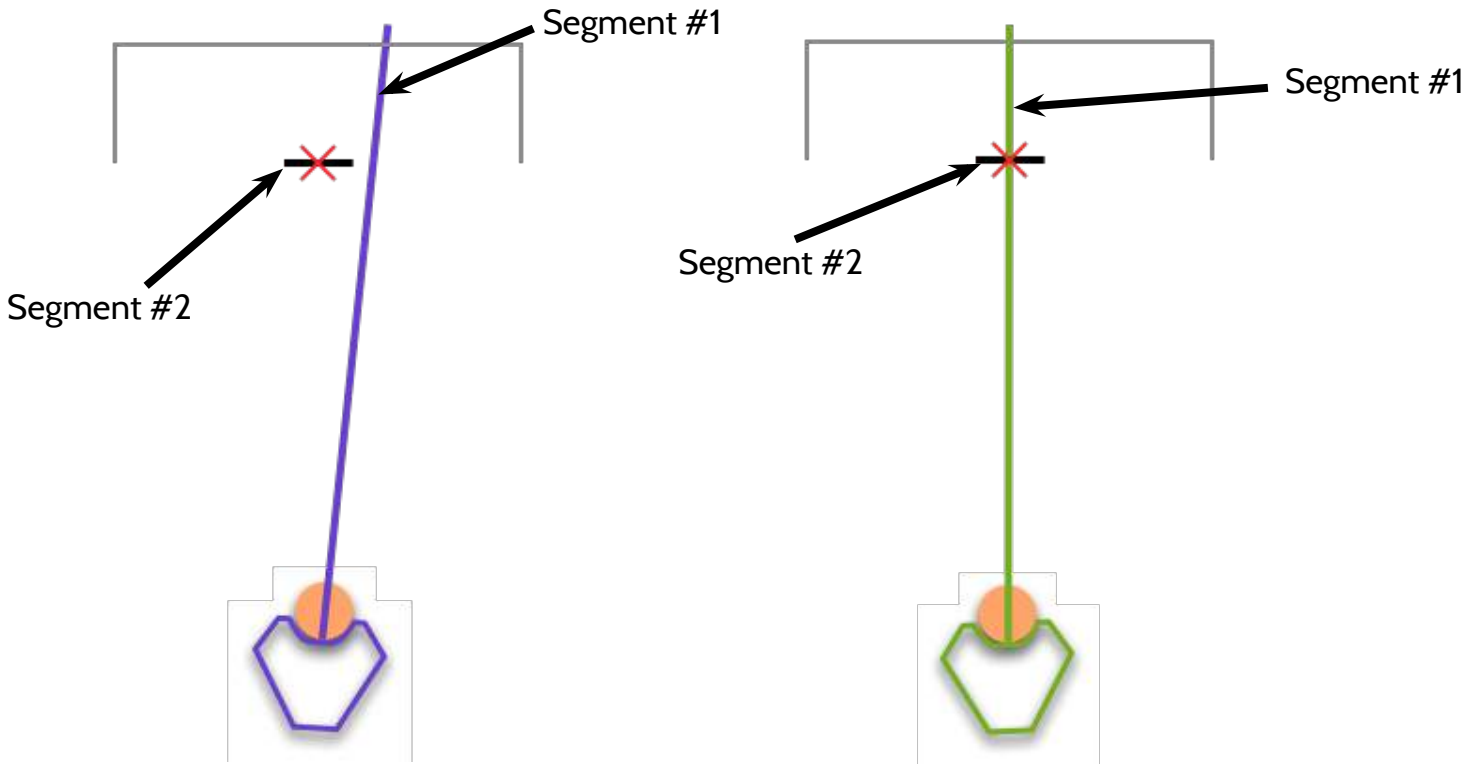
Ball Passes

Aim and Kick

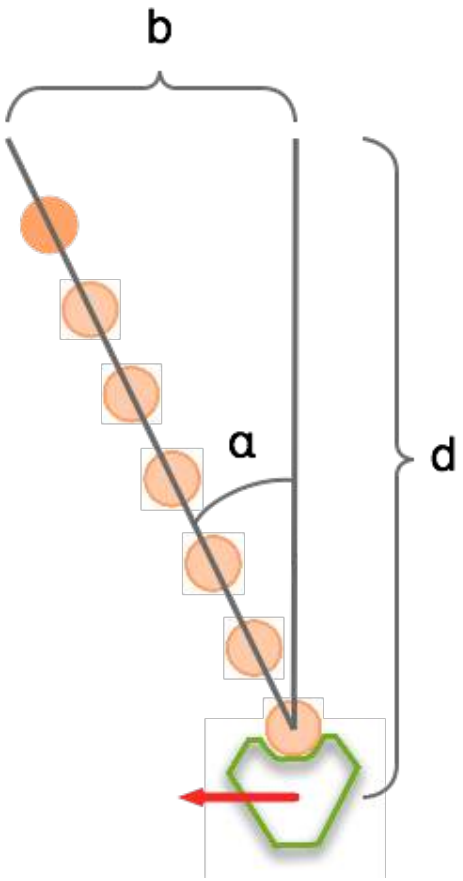
Previous alignment-check algorithm

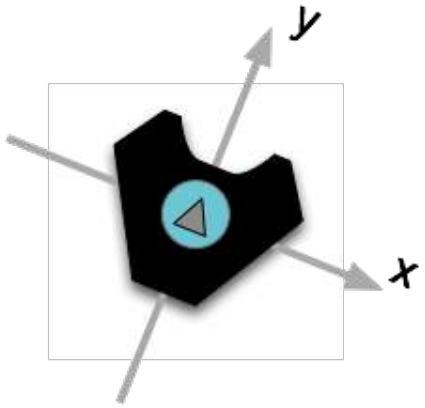
Aim and Kick

New alignment-check algorithm



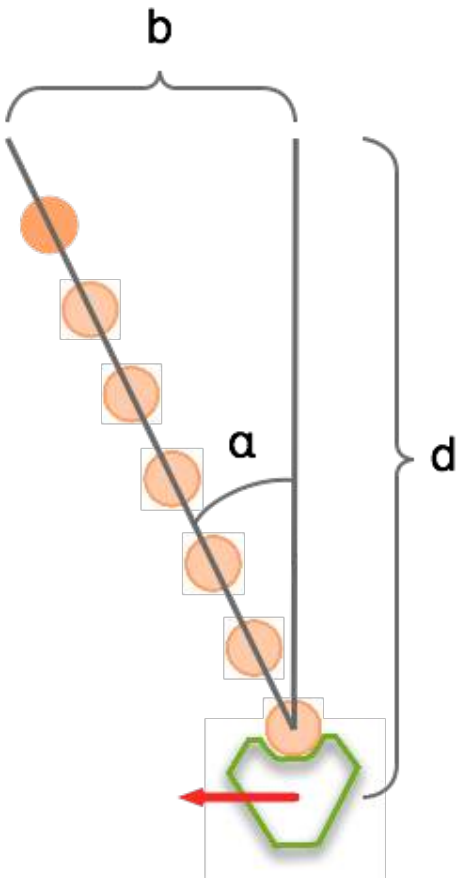
Aim and Kick

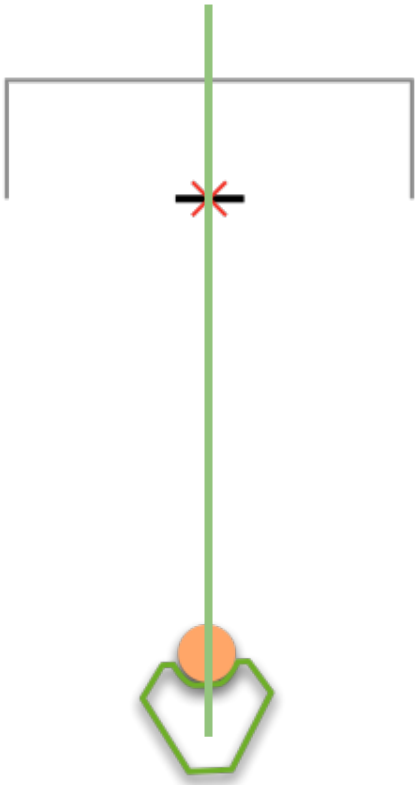




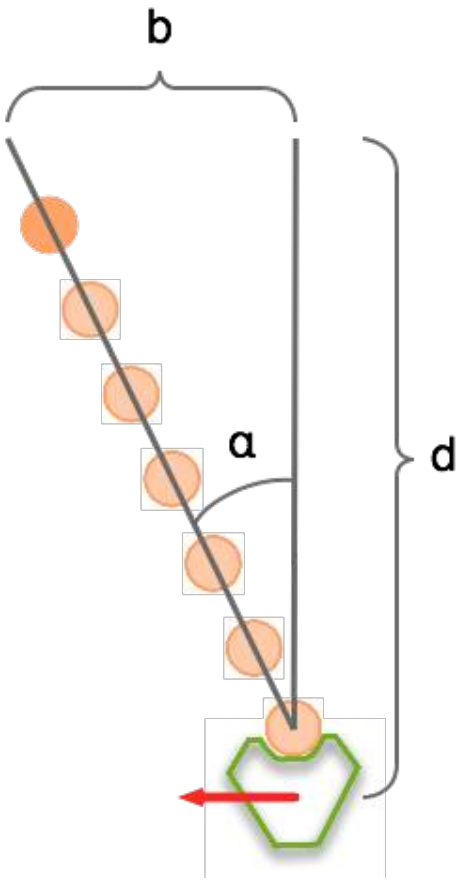
Vel_x [m/s]	d [m]	b [m]	α [°]
0.5	6.0	0.5	4.76
1.0	6.0	1.0	9.46
2.0	6.0	1.8	16.70

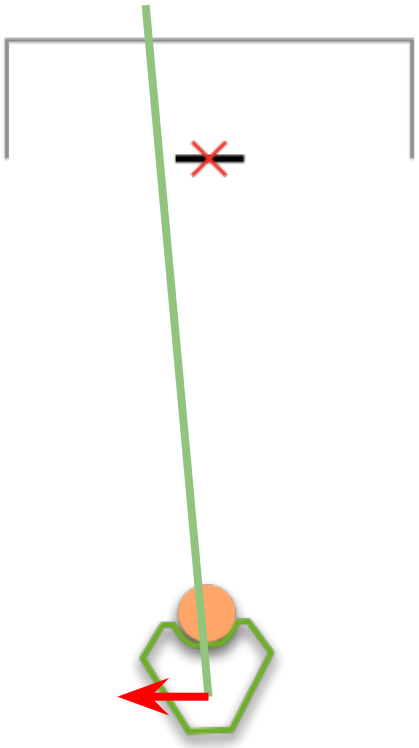
Aim and Kick



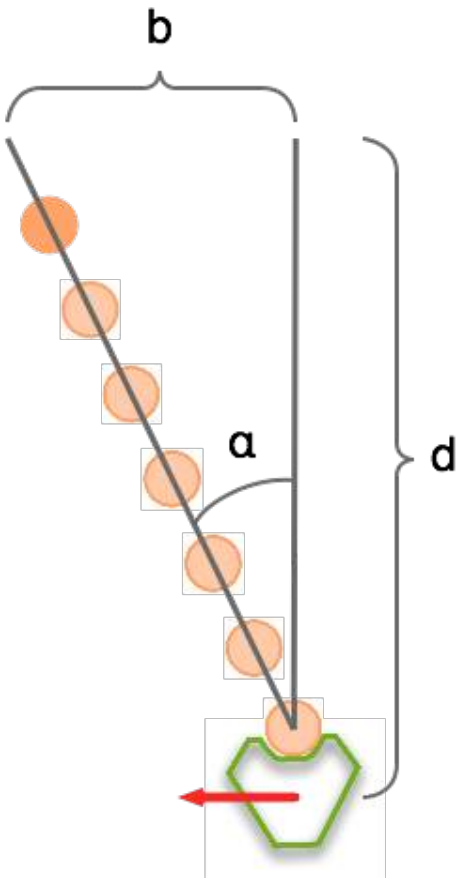


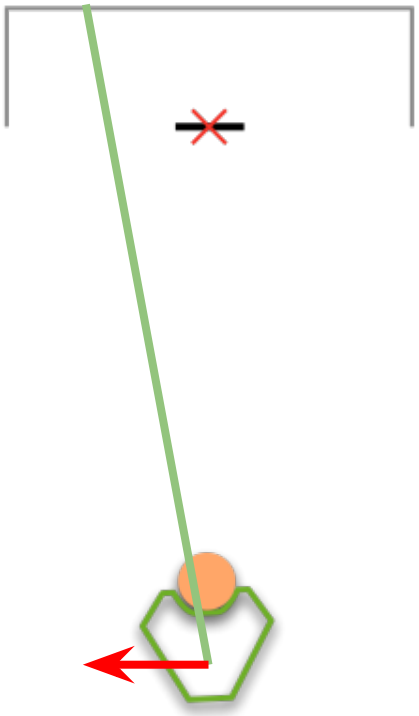
Aim and Kick



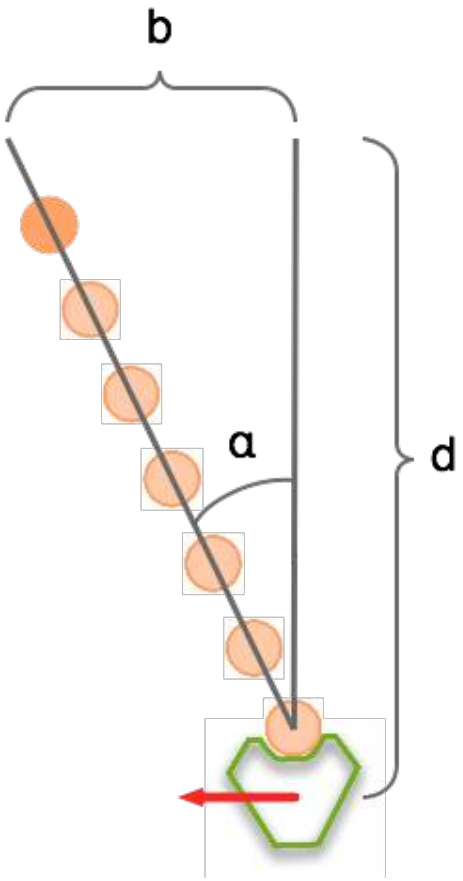


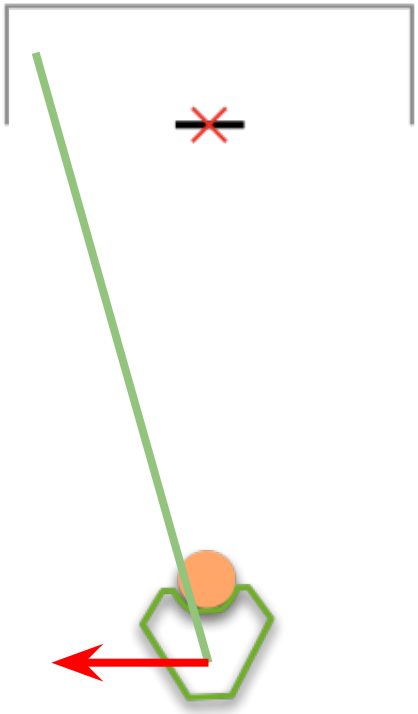
Aim and Kick



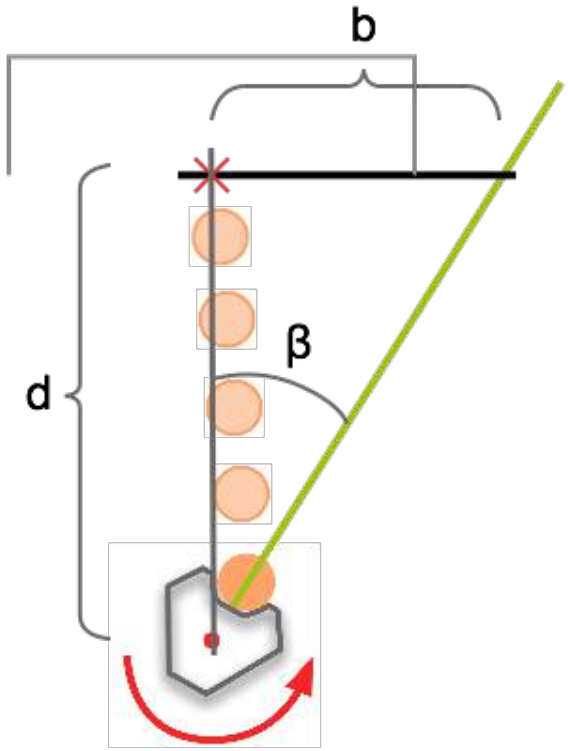


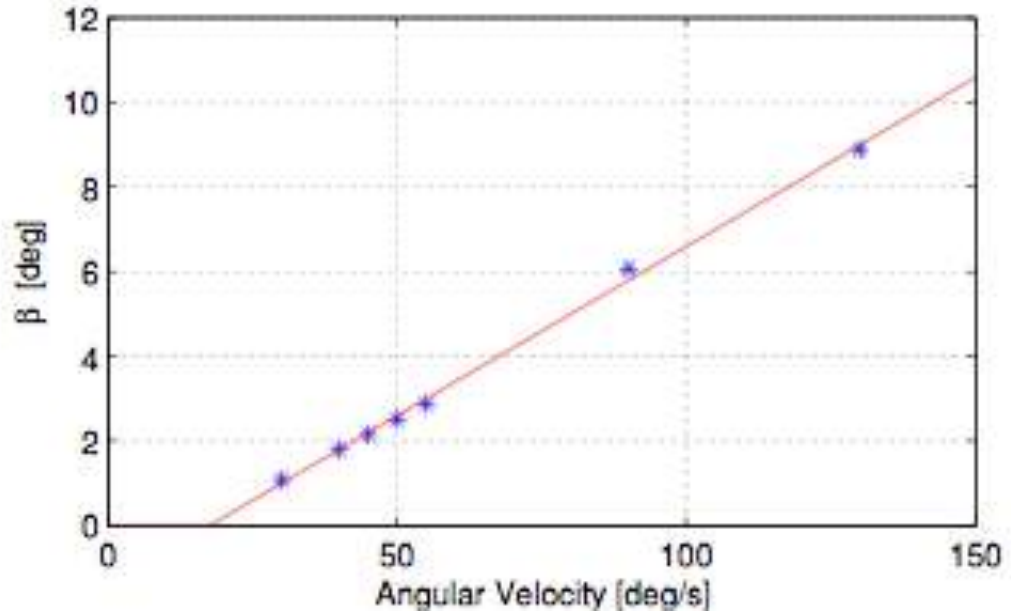
Aim and Kick



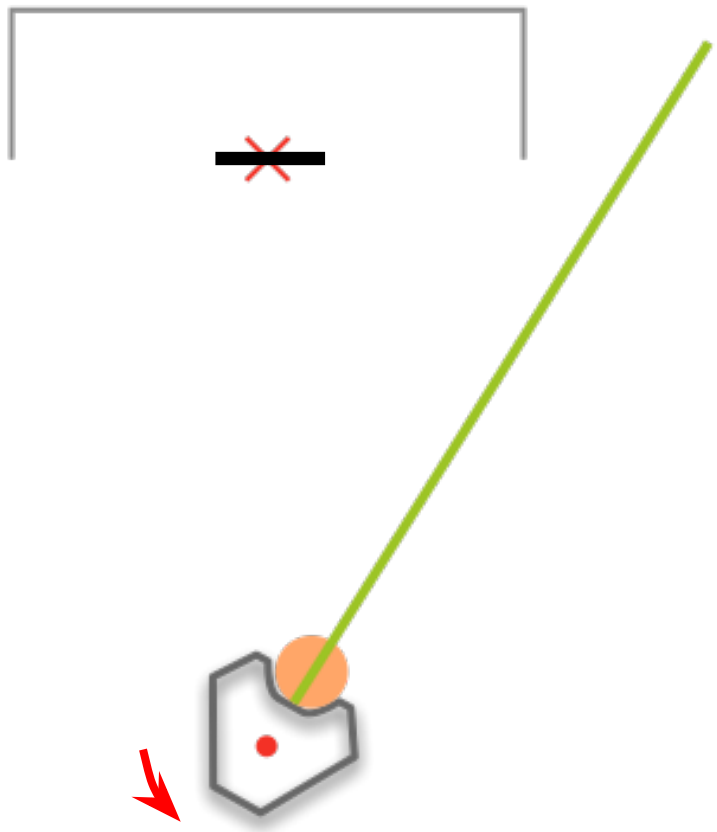


Aim and Kick

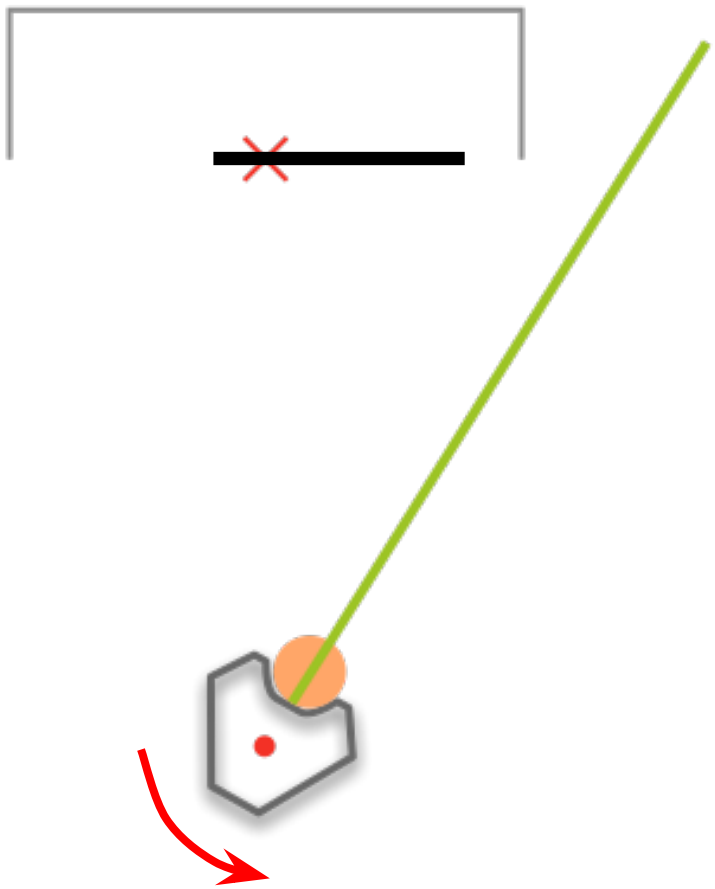




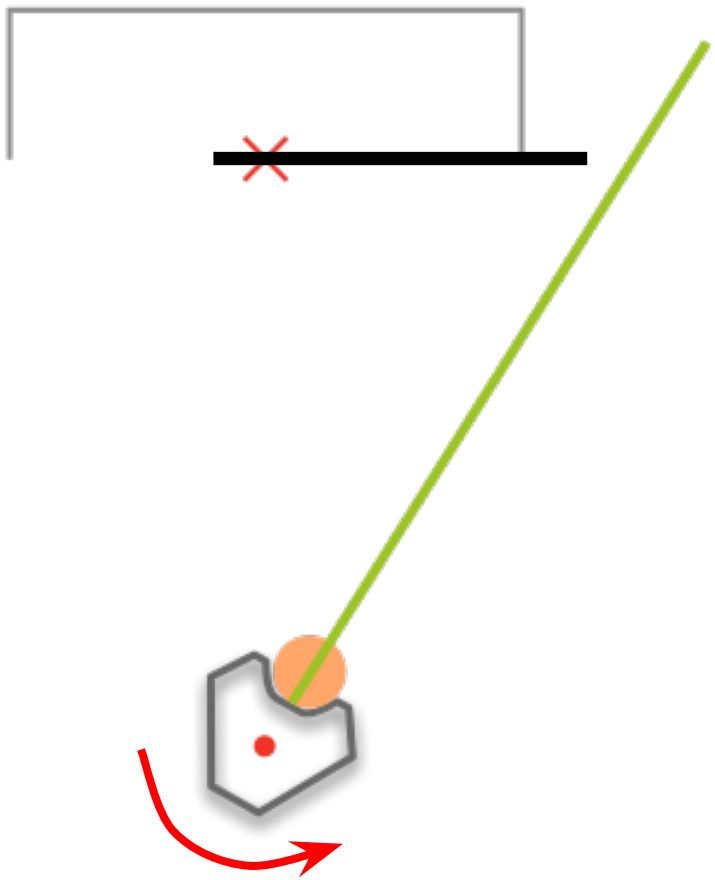
Aim and Kick



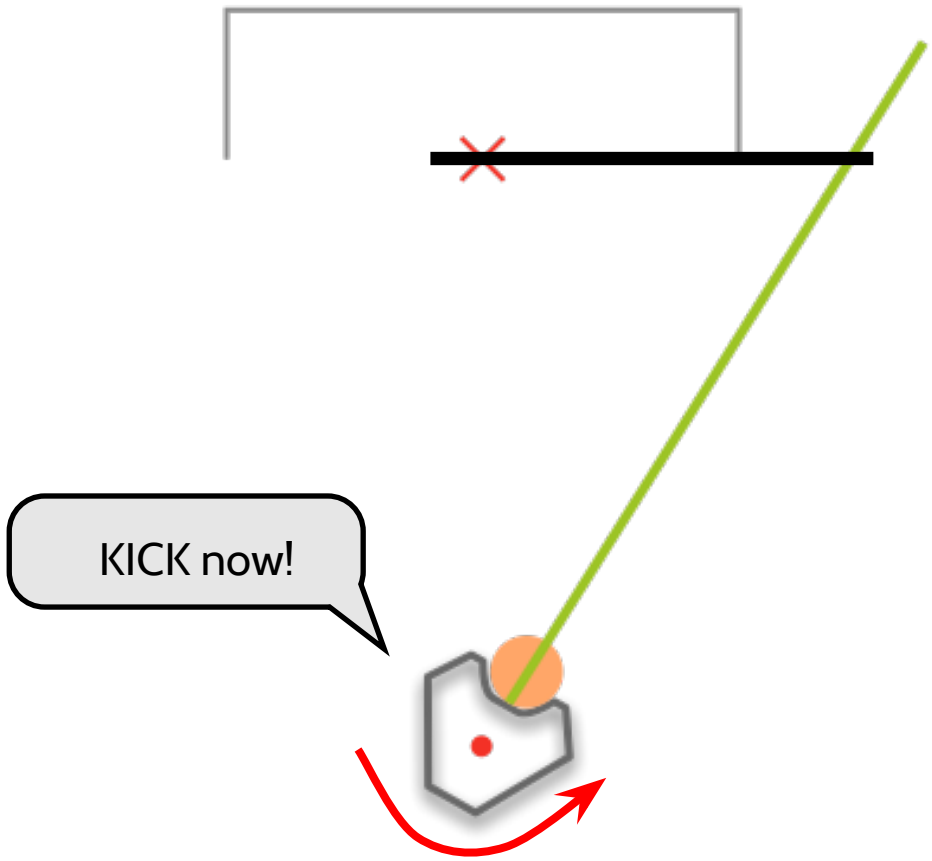
Aim and Kick



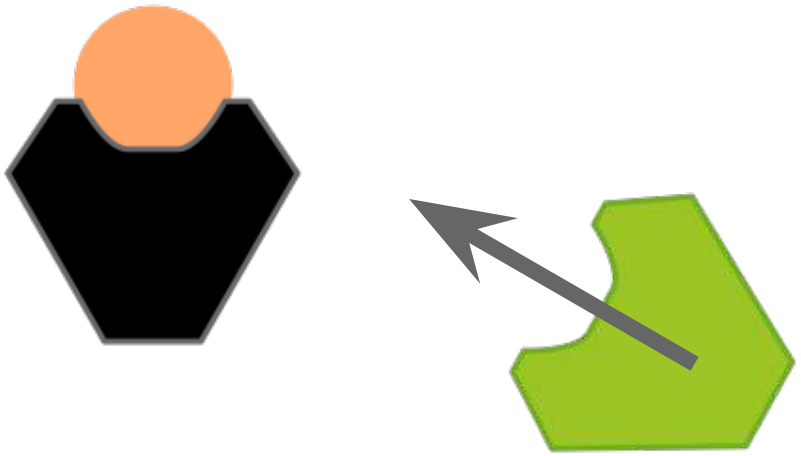
Aim and Kick



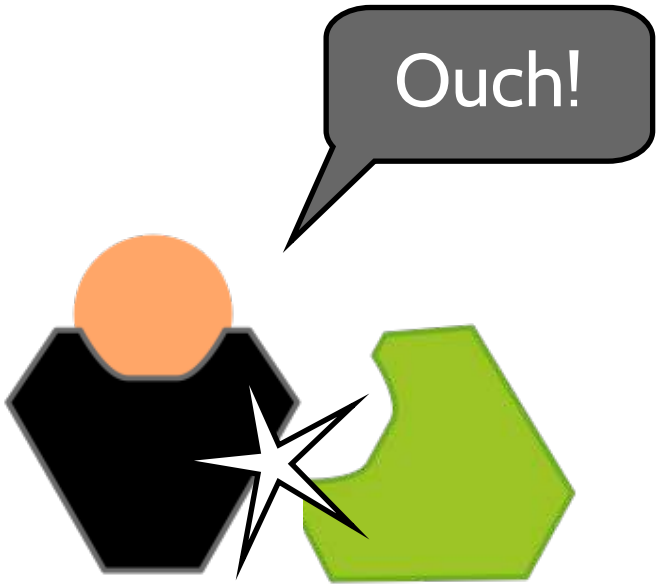
Aim and Kick



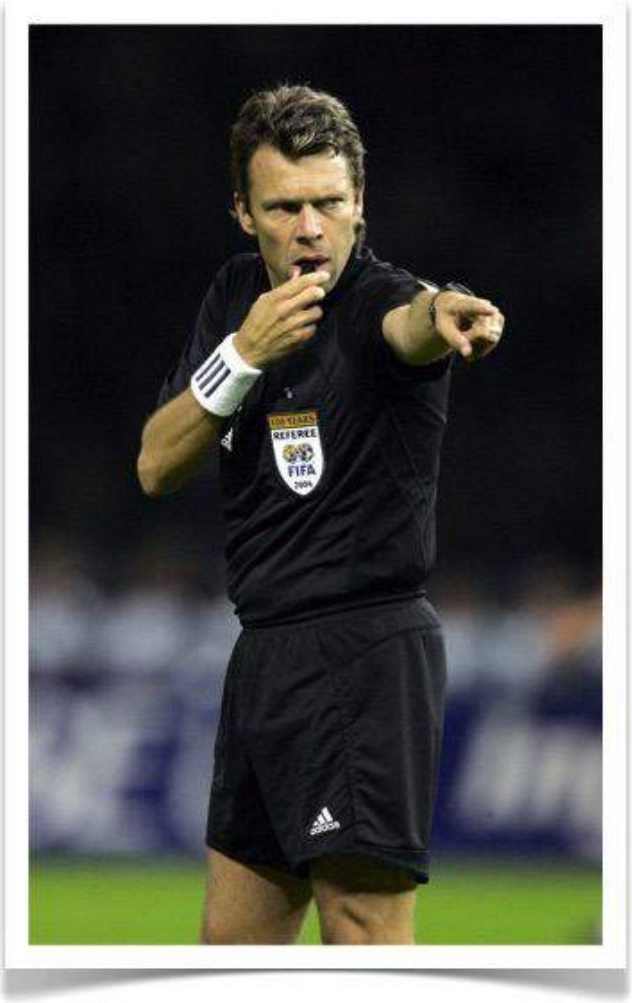
Contour Obstacles

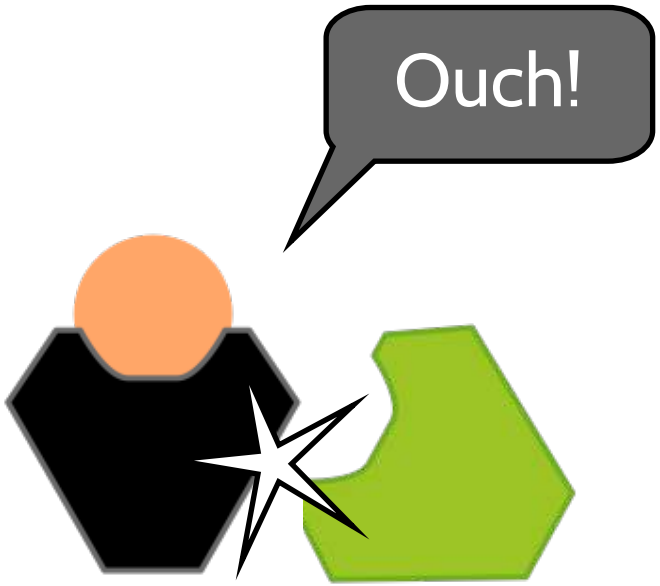


Contour Obstacles

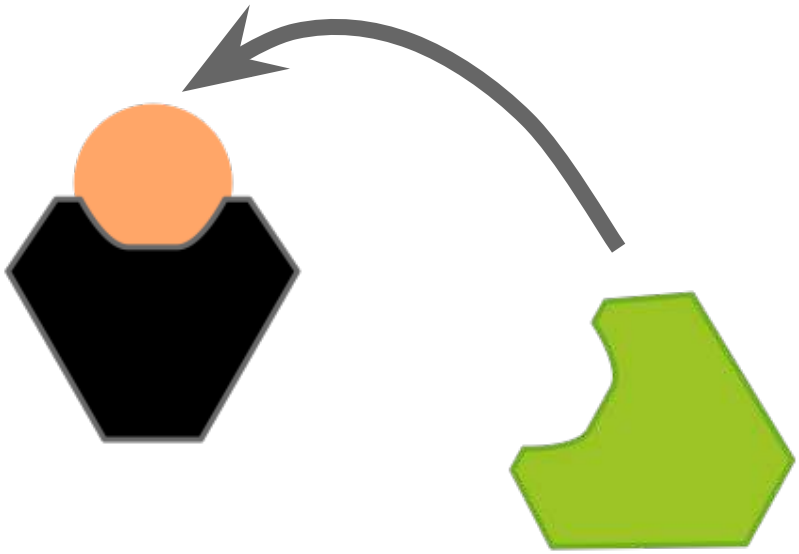


Contour Obstacles

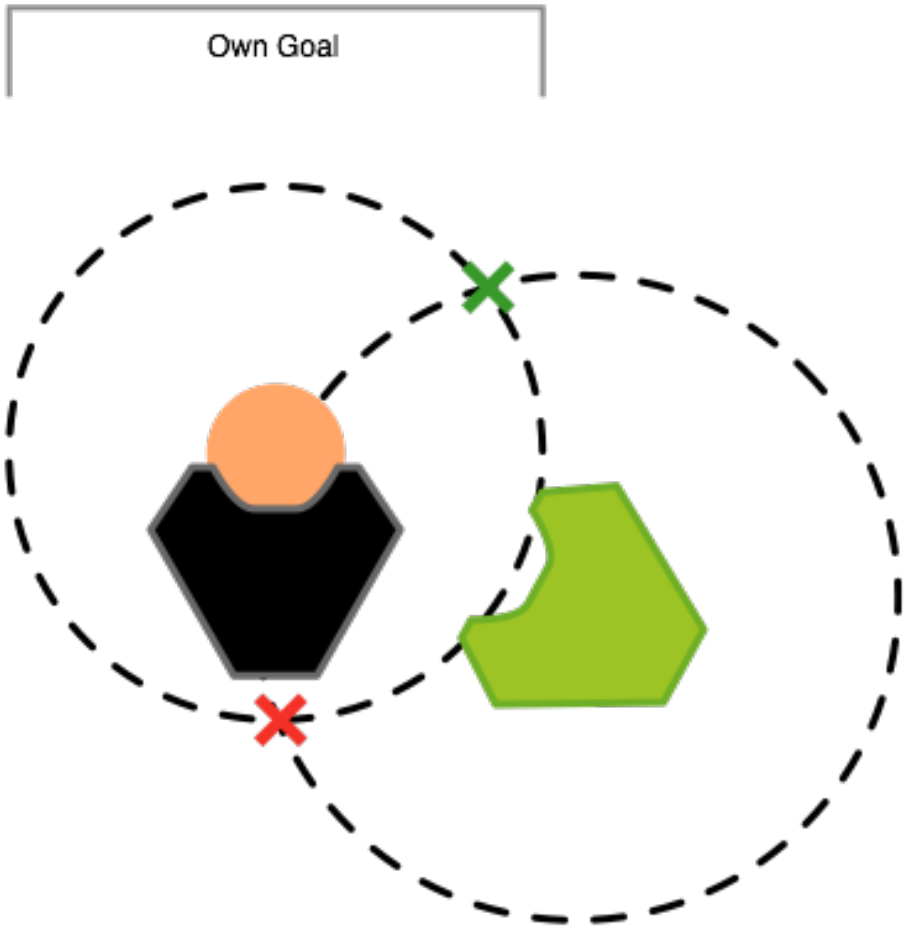




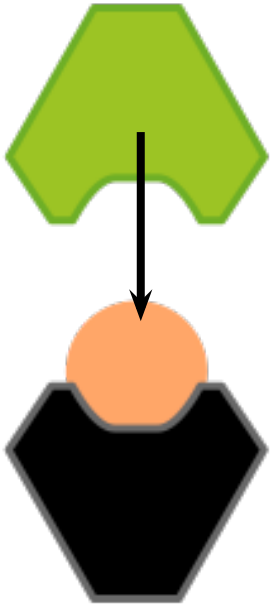
Contour Obstacles



Contour Obstacles



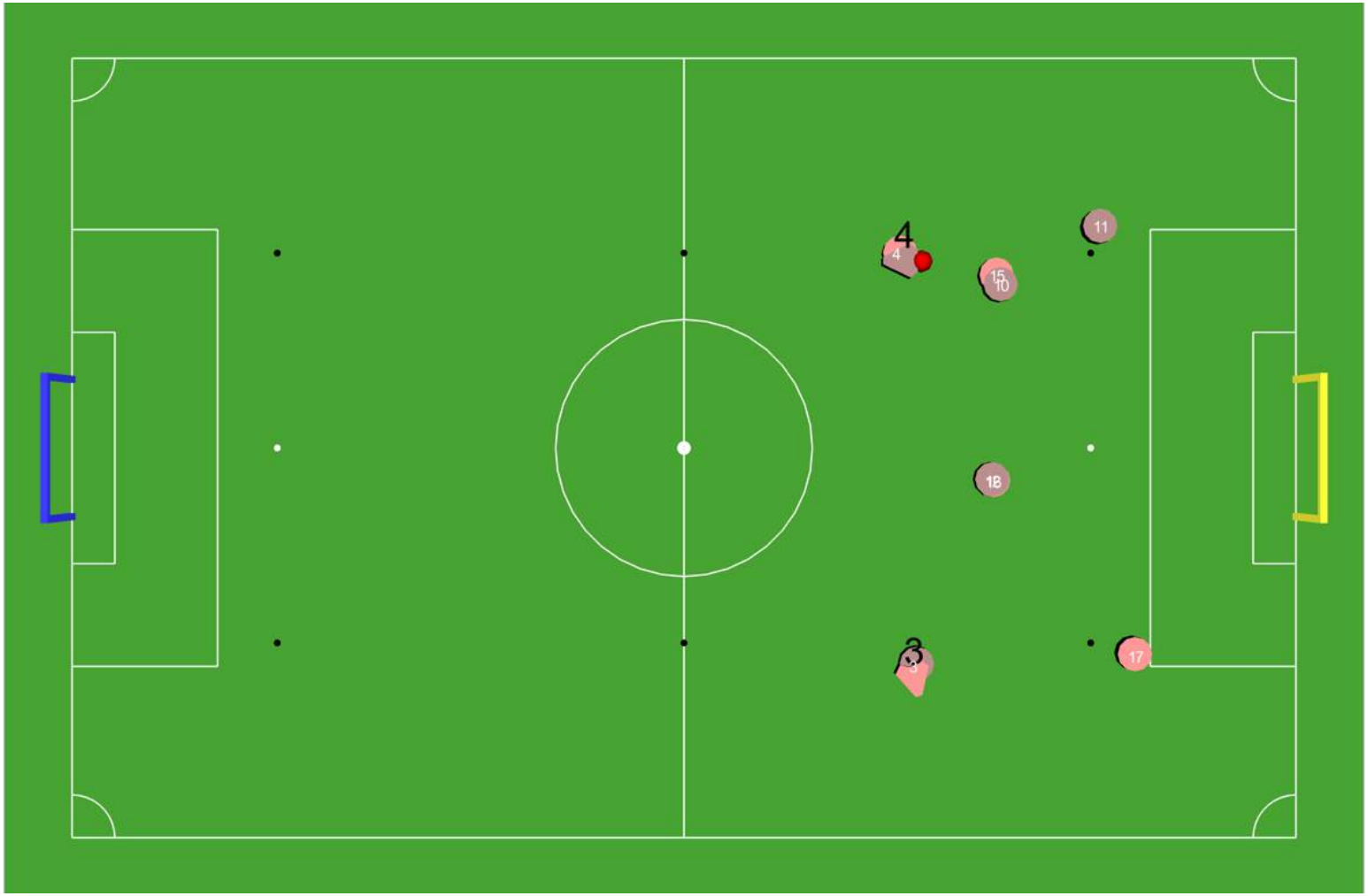
Contour Obstacles



Contour Obstacles

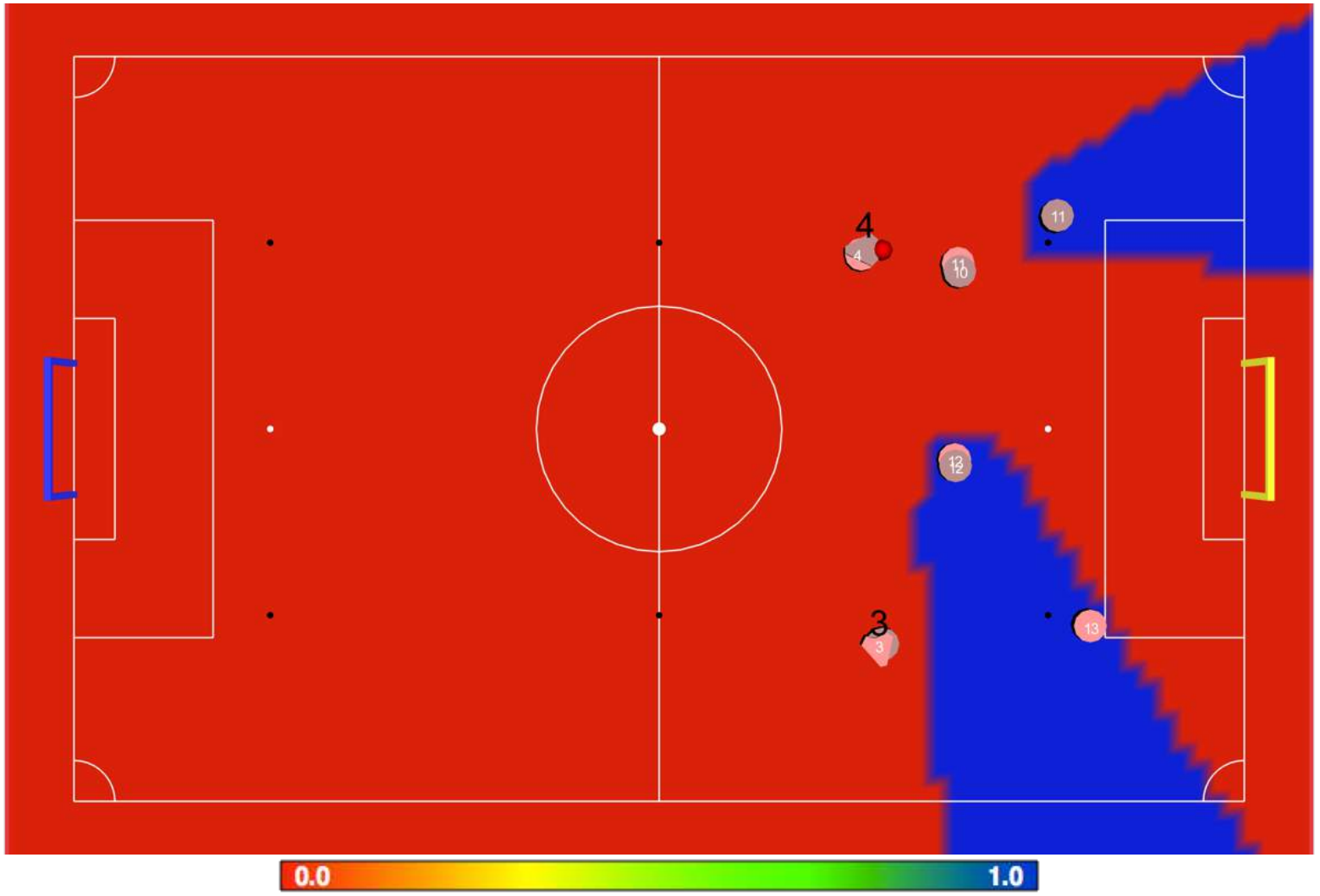
- Introduction
- New Agent Architecture
- Developed Behaviors
- **Heightmap Integration**
- Results
- Conclusions

Heightmap Integration



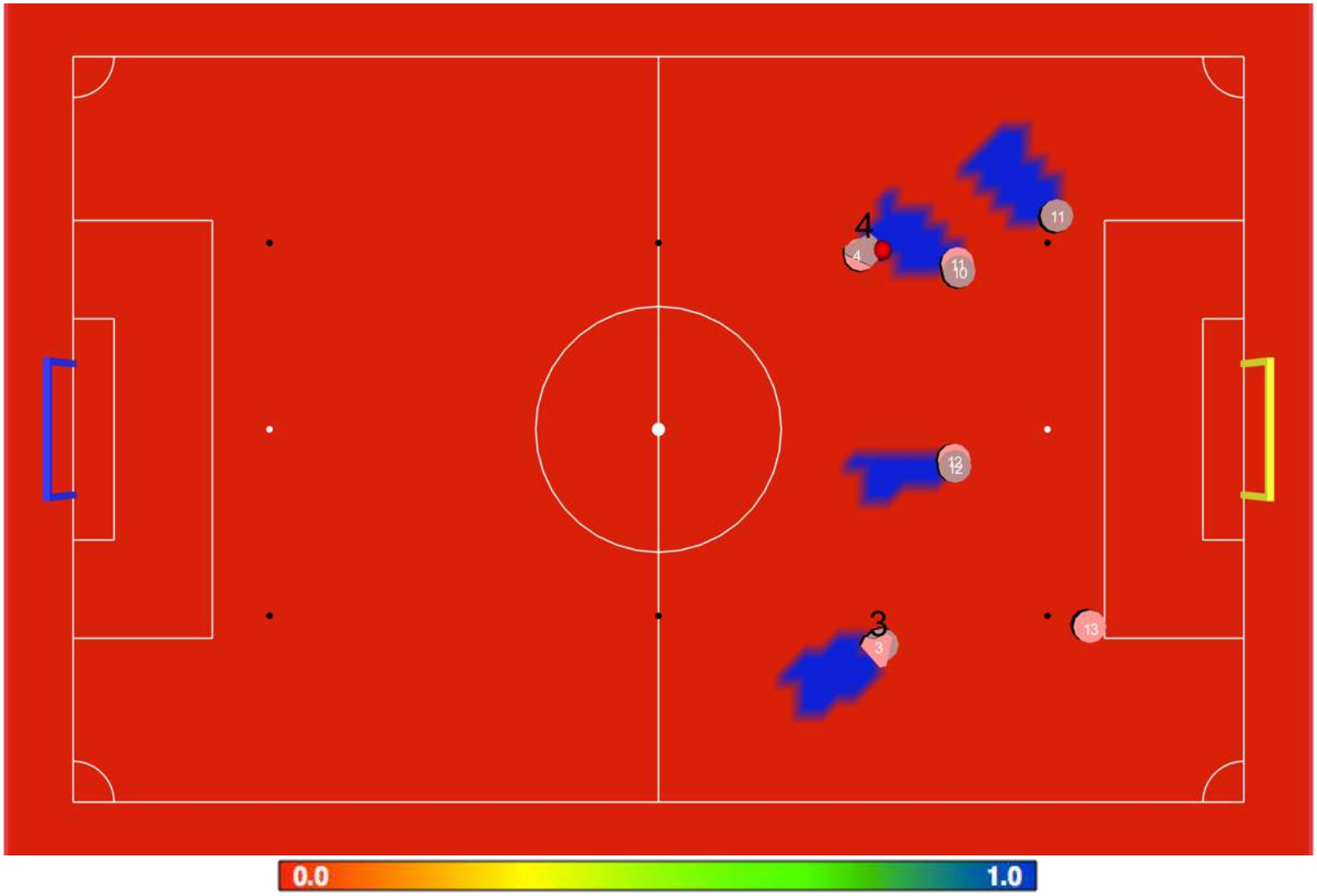
Heightmap Integration

Field of View from the ball position



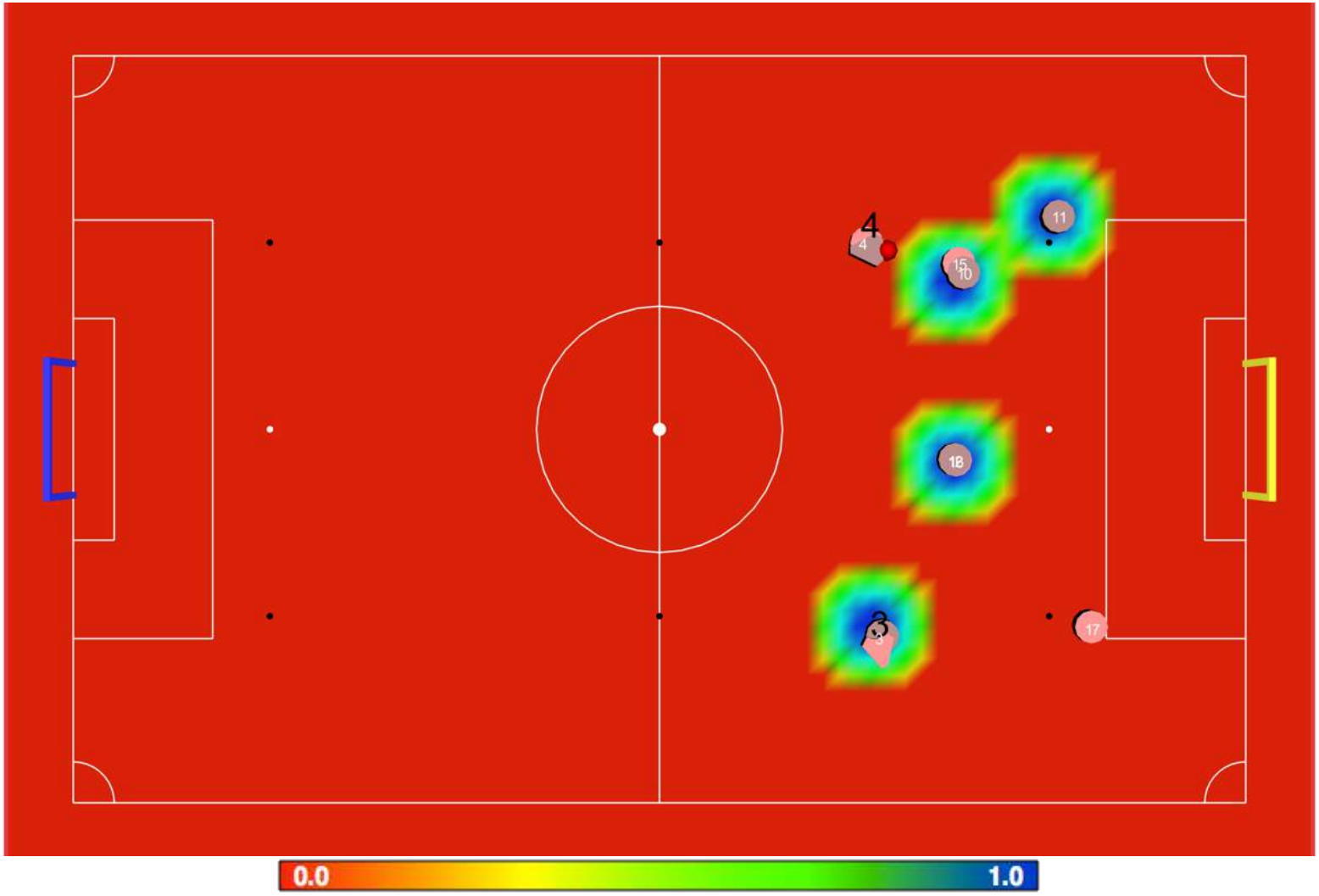
Heightmap Integration

Field of View from the opponent goal



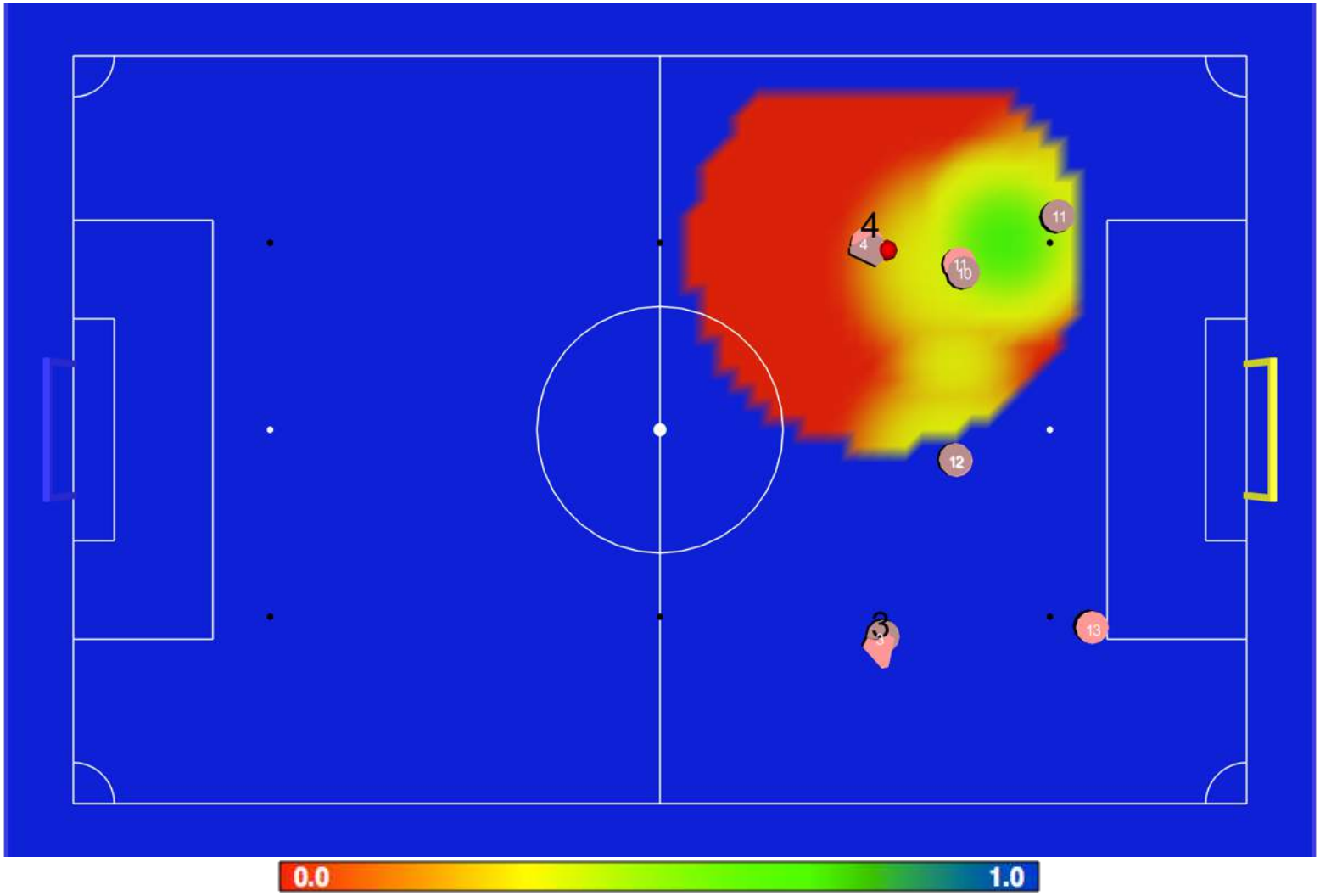
Heightmap Integration

Occupancy Map with some persistency



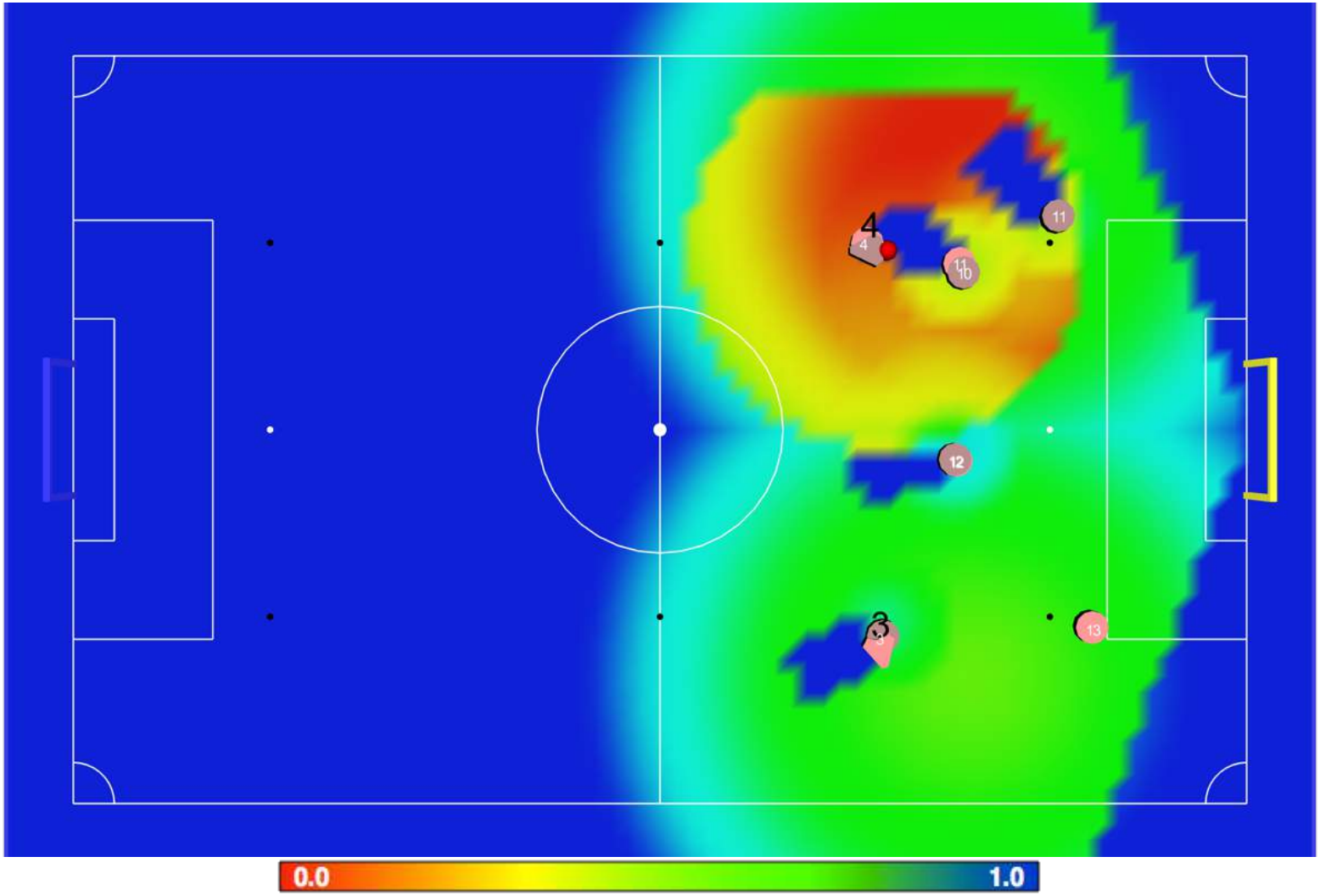
Heightmap Integration

Dribble Map



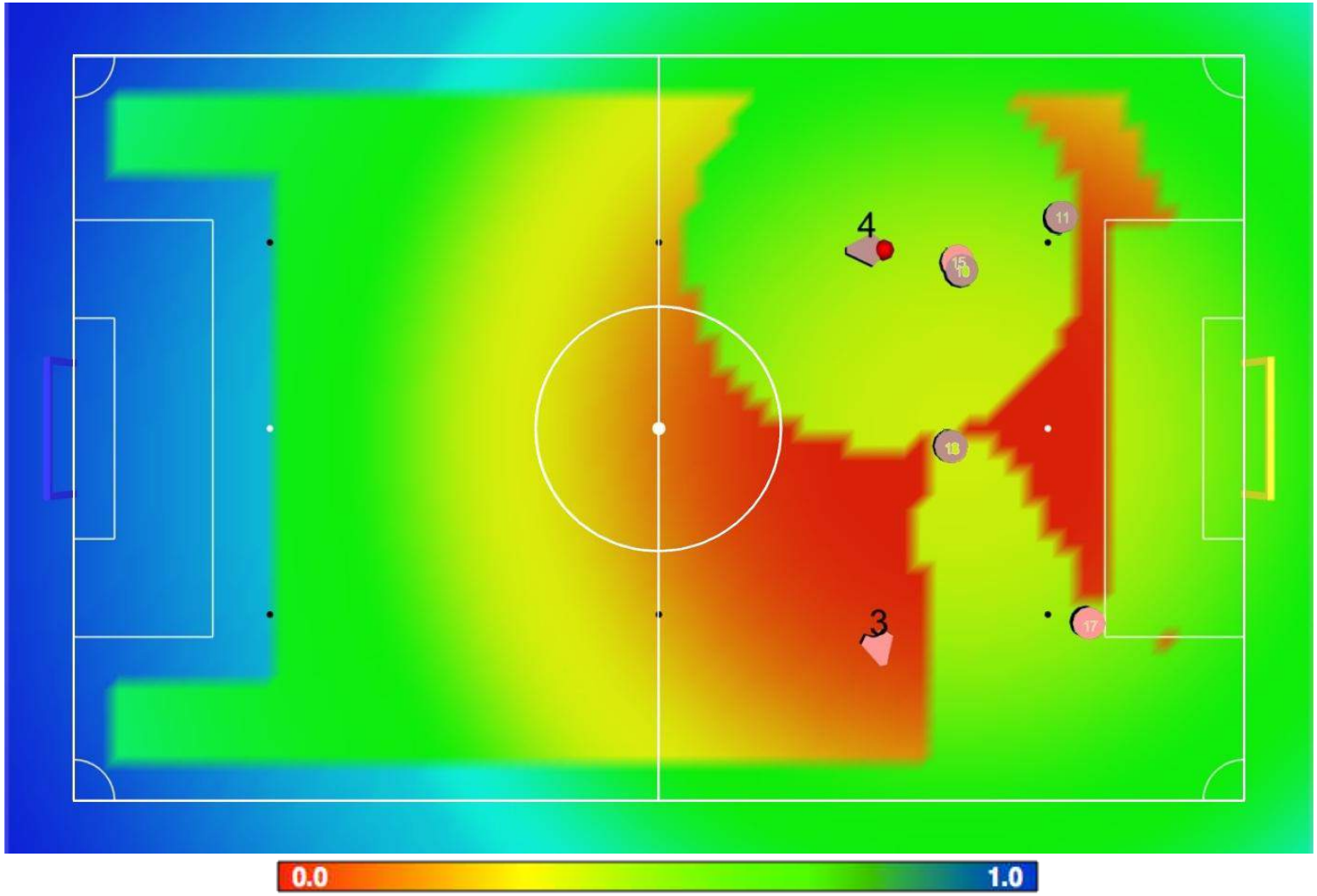
Heightmap Integration

Dribble map + kick



Heightmap Integration

Free-play ball reception map

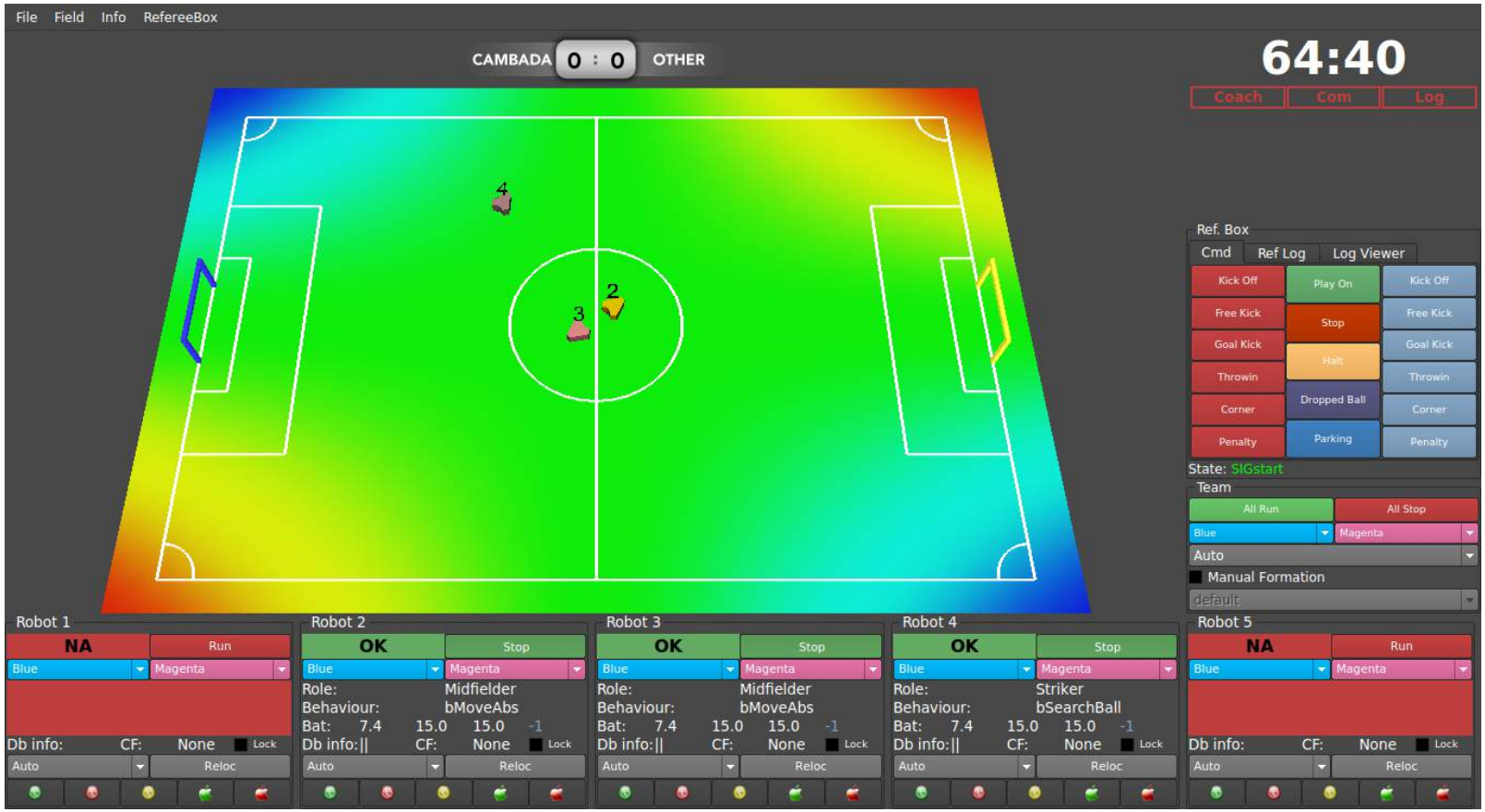


Heightmap Integration

Basestation Tool 2D (2013)

Heightmap Integration

Basestation Tool 3D (2014)



- Introduction
- New Agent Architecture
- Developed Behaviors
- Heightmap Integration
- **Results**
- Conclusions

Results New Architecture

- `Roles`
  - Reduced ~80% the lines of code

Role	Agent v1	Agent v2	Ratio
Striker	583	28	4.8%
Midfielder	193	29	15.0%
Goalie	176	28	15.9%
Replacer	541	94	17.4%
Receiver	675	322	47.7%

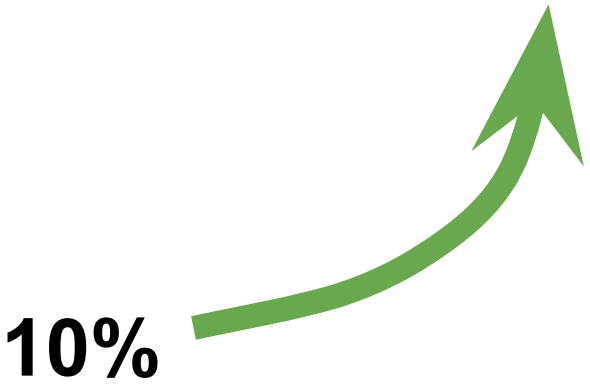
  - Simplified logic
    - Increase in the **number** of `Behaviours`

Results New Behaviours

- **Aim and Kick Behaviour**
  - Prev. Algorithm (IranOpen): ~59.5% accuracy
  - New Algorithm (Robotica2014): **~74.0% accuracy**
    - **Increased** chance of scoring a goal
- **Contour Obstacles Behaviour**
  - Firstly used in Robotica2014
    - **~35%** the Striker time
    - Improved the defensive attitude of the team
      - Less fouls (and also less damage, safer)

Results Ball Passes

- Lab test
  - 2 robots
  - 5 minutes
  - 52 successful passes / 63 total



HW still has room for improvement!!

- Introduction
- New Agent Architecture
- Developed Behaviors
- Heightmap Integration
- Results
- **Conclusions**

Conclusions

- **Problem solved**: `Role` complexity

```
#include "RoleStriker.h"

namespace cambada {

RoleStriker::RoleStriker() : Role(rStriker){
     bList->addBehaviour( new BStopRobotGS );
     bList->addBehaviour( new BKickToTheirGoal );
     bList->addBehaviour( new BStrikerPass );
     bList->addBehaviour( new BStrikerGoToBall );
}

RoleStriker::~RoleStriker() {
}

} /* namespace cambada */
```

Conclusions

- **Problem solved**: `Role` complexity

```
#include "RoleStriker.h"

namespace cambada {

RoleStriker::RoleStriker() : Role(rStriker){
     bList->addBehaviour( new BStopRobotGS );
     //bList->addBehaviour( new BKickToTheirGoal );
     bList->addBehaviour( new BStrikerPass );
     bList->addBehaviour( new BStrikerGoToBall );
}

RoleStriker::~RoleStriker() {
}

} /* namespace cambada */
```

Conclusions

- Current rules easily implemented
  - In the **new architecture**!

- **Problem solved**: history loss
  - `Roles` and `Behaviours`:
    - Instantiated in the constructors
    - They exist during the agent's lifetime

Thanks for your attention!

…

Thanks for your attention!

Public code release soon

(check robotica.ua.pt/CAMBADA)