A ROS-based framework for collaborative robot teams

Andreas Witsch

Distributed Systems Group
Kassel University

VERTEILTE SYSTEME

CARPE NOCTEM
ROBOTIC SOCCER
DIVIDE ET IMPERA

Outline

RoboCup

RoboCup is a multi-national research effort aiming at:

- Combining and advancing artificial intelligence, robotics, and related fields
- Providing a setting for comparison of different approaches
- Improving public visibility

Robotic Soccer & the Middle Size League

- Five robots per team
- Fifteen minutes per half time
- 18 x 12m field
- Robots are 80cm high, and weigh ∼35kg
- FIFA rules apply

Carpe Noctem

Founded in 2005, Carpe Noctem participates in RoboCup succesfully since 2006:

- 2006: 7th at WC in Bremen
- 2007: 5th at GO in Hannover
- 2008: 4th at GO in Hannover
- 2009: 4th at GO in Hannover
  5th at WC in Graz
- 2010: 4th at GO in Magdeburg
- 2011: 3rd at GO in Magdeburg
  7th at WC in Istanbul
- 2012: 4th at DO in Eindhoven

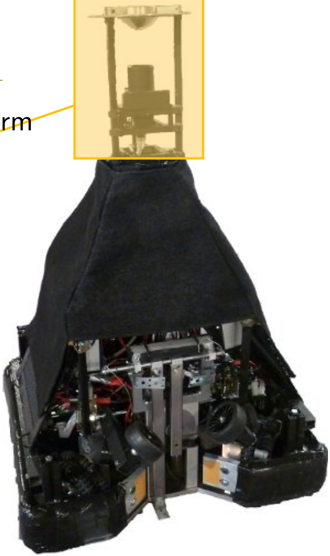
Carpe Noctem Robot

- Specifically designed hardware platform

Carpe Noctem Robot

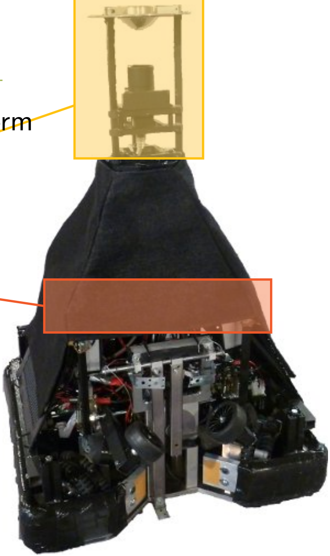
- Specifically designed hardware platform
- Omnidirectional camera

  IEEE1394a, 640 × 480, 30 fps



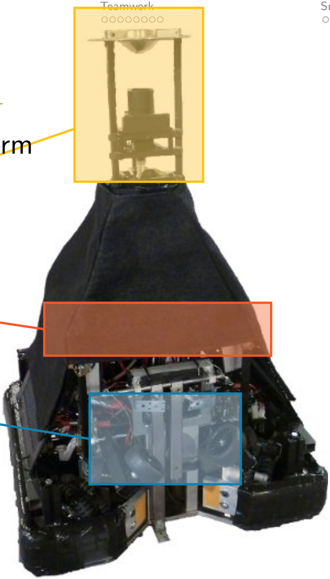
Carpe Noctem Robot

- Specifically designed hardware platform
- Omnidirectional camera
  IEEE1394a, 640 × 480, 30 fps
- IPC
  Intel Core i7, 4 GiB
  802.11a WLAN



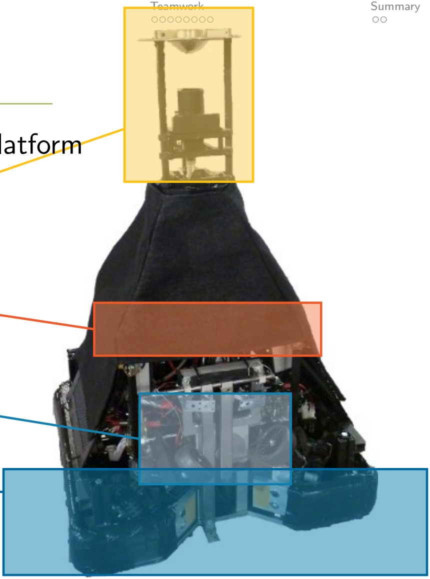
Carpe Noctem Robot

- Specifically designed hardware platform
- Omnidirectional camera
  IEEE1394a, 640×480, 30 fps
- IPC
  Intel Core i7, 4 GiB
  802.11a WLAN
- Kicking device
  350 V, 12 m Kicking Range

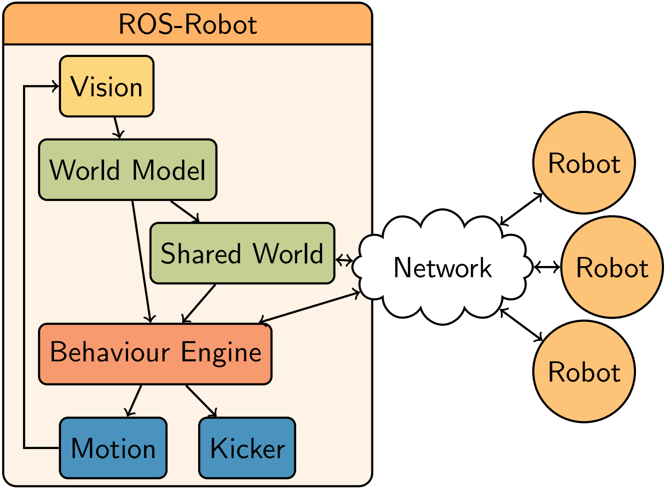


Carpe Noctem Robot

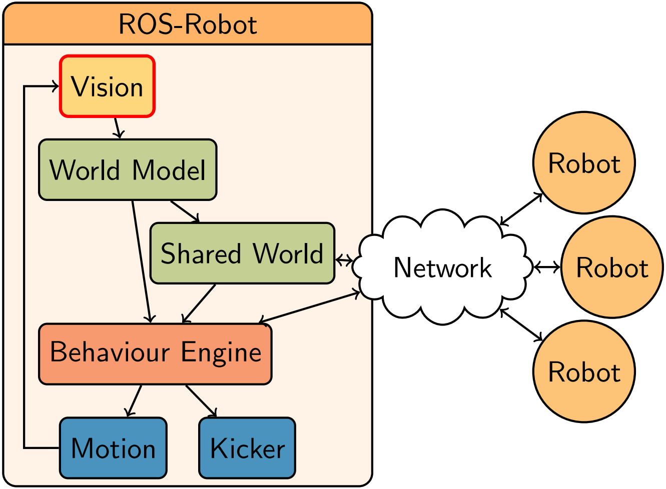
- Specifically designed hardware platform
- Omnidirectional camera
  IEEE1394a, 640 × 480, 30 fps
- IPC
  Intel Core i7, 4 GiB
  802.11a WLAN
- Kicking device
  350 V, 12 m Kicking Range
- Holonomic Drive
  4 Maxon 200W
  Up to 5 $\frac{m}{s}$; Maxon MC



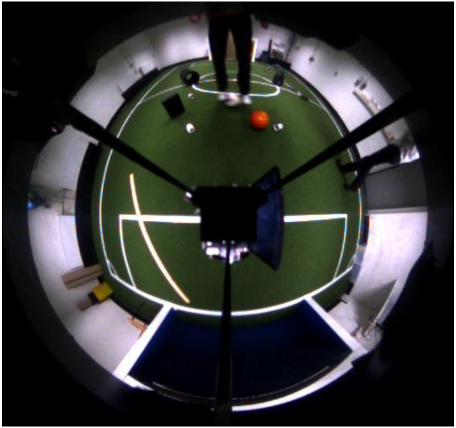
ROS Architecture Overview



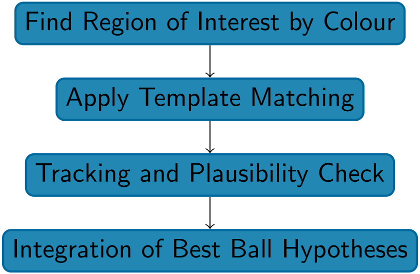
ROS Architecture Overview



Robot Camera Perspective

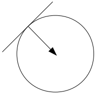


Balldetection

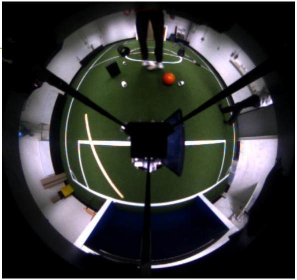


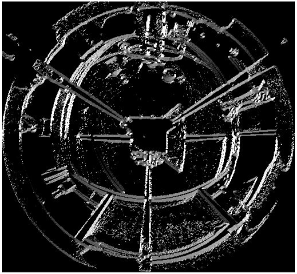
Template Matching

From the insight, that every normal vector of a ball edge is directed to the ball center, the template matching makes use of the edge direction.

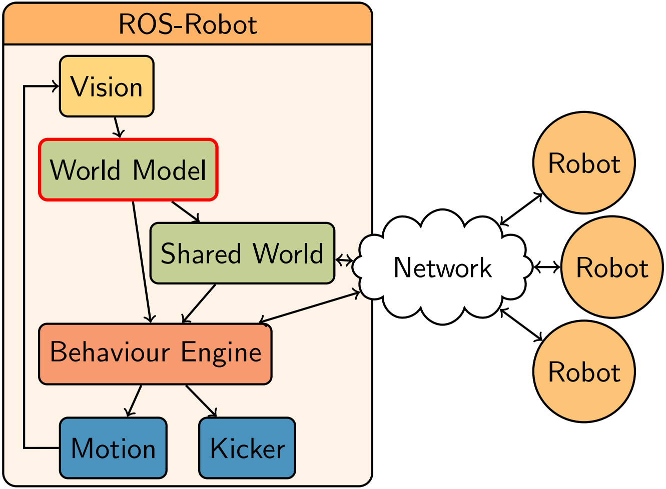


About 12 representative pixels are enough for a suitable classification and reduces computation time.

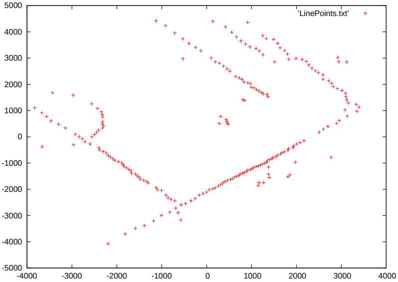


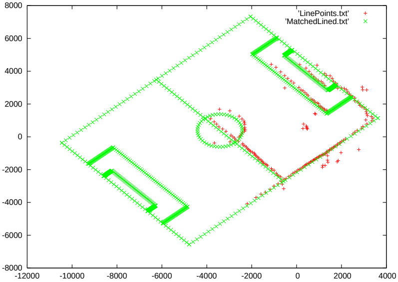


ROS Architecture Overview



Localisation





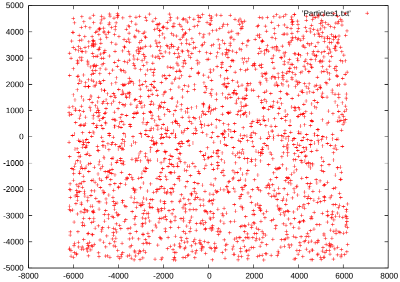
Line points

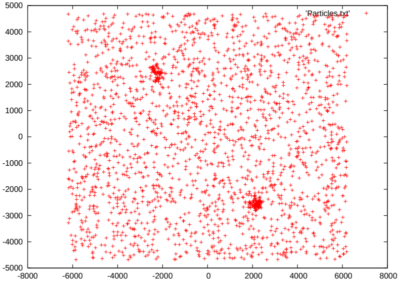
These line points were extracted from the omni-vision image

Line points compared to reference set

An overlay extracted line points and a reference with perfect line points

Particle Filter





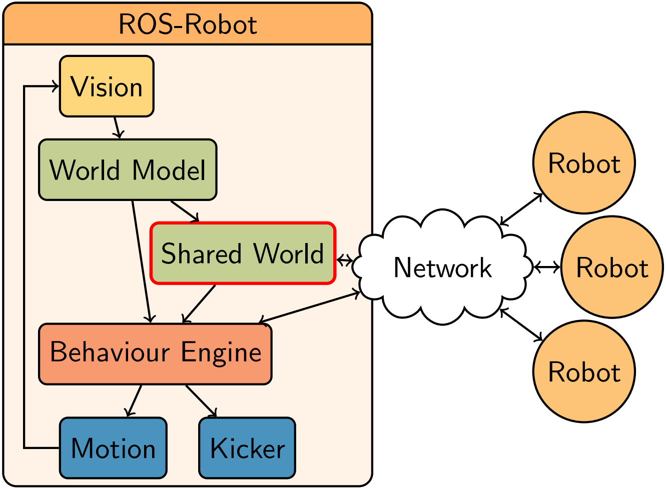
Particle initialisation

A new set of particles as position hypotheses

Particles after second iteration

Already clearly converged towards two possible positions

ROS Architecture Overview



Sensor Fusion

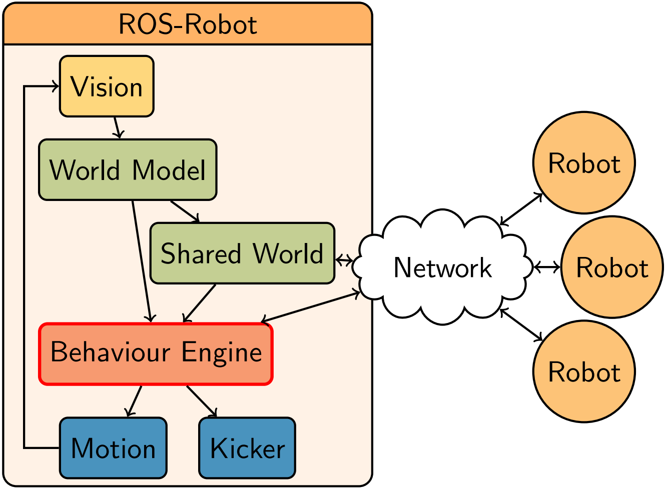
Quality of the extracted information is limited due to:

- Sensor noise
- Limited sensor range
- Estimation errors

To extract more precise information we use sensor fusion techniques to combine the individual informations to a global view, such as:

- Shared ball detection
- Obstacle merging

ROS Architecture Overview



Teamwork in Dynamic Domains

Teams of agents cooperating in dynamic domains require:

- A language to describe team plans ("recipe") from a global perspective
- Ease of Modelling: simple and intuitive way to model multi-agent plans
- Formal semantics according to which these plans are executed
- Fast reaction of the team to changing situations
- Robustness towards breakdown of individual agents

Modelling Language: ALICA

A Language for **I**nteractive **C**ooperative **A**gents

Core Language Elements:

- *Behaviours* – atomic single-agent action programs

Modelling Language: ALICA

A **L**anguage for **I**nteractive **C**ooperative **A**gents
Core Language Elements:

- *Behaviours* – atomic single-agent action programs
- *Plans* – abstract multi-agent activity descriptions

Modelling Language: ALICA

A **L**anguage for **I**nteractive **C**ooperative **A**gents

Core Language Elements:

- *Behaviours* – atomic single-agent action programs
- *Plans* – abstract multi-agent activity descriptions
- *Plantypes* – sets of alternative plans

Modelling Language: ALICA

A **L**anguage for **I**nteractive **C**ooperative **A**gents
Core Language Elements:

- *Behaviours* – atomic single-agent action programs
- *Plans* – abstract multi-agent activity descriptions
- *Plantypes* – sets of alternative plans
- *Tasks* – denote specific activities within plans

Modelling Language: ALICA

A **L**anguage for **I**nteractive **C**ooperative **A**gents
Core Language Elements:

- *Behaviours* – atomic single-agent action programs
- *Plans* – abstract multi-agent activity descriptions
- *Plantypes* – sets of alternative plans
- *Tasks* – denote specific activities within plans
- *Roles* – descriptions of capabilities

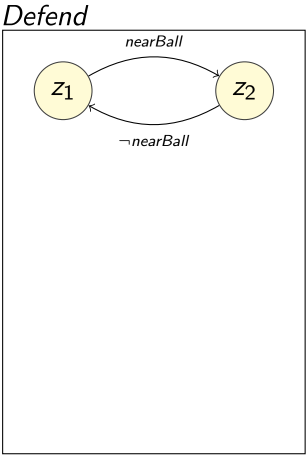
Plans

- A plan is a recipe for achieving a goal or maintaining a condition

Defend

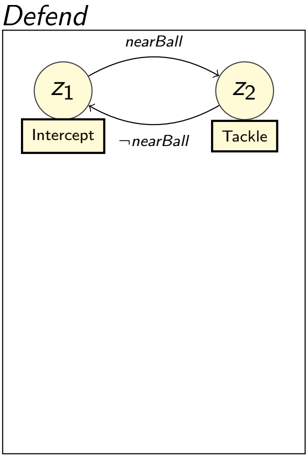
Plans

- A plan is a recipe for achieving a goal or maintaining a condition
- Consisting of *states* and *transitions* between them



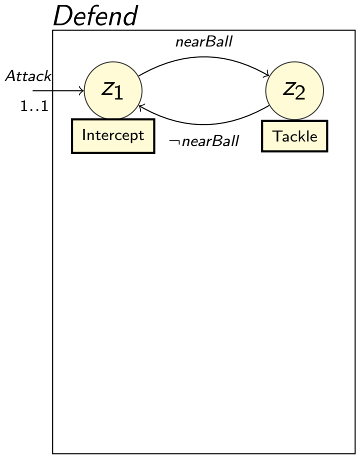
Plans

- A plan is a recipe for achieving a goal or maintaining a condition
- Consisting of *states* and *transitions* between them
- States contain Behaviours and Plantypes



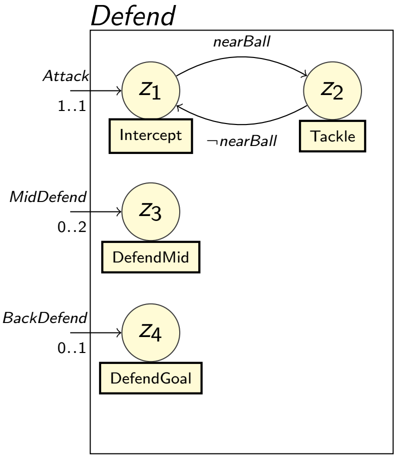
Plans

- A plan is a recipe for achieving a goal or maintaining a condition
- Consisting of *states* and *transitions* between them
- States contain Behaviours and Plantypes
- Initial states are tagged by *Tasks* and *Cardinalities*



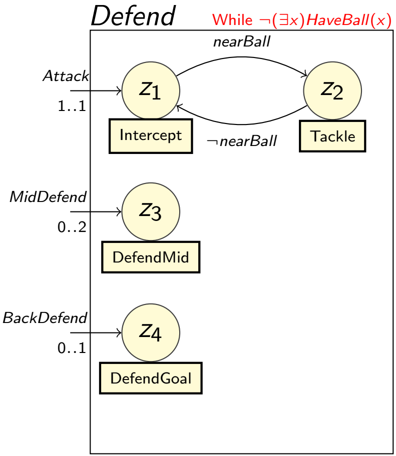
Plans

- A plan is a recipe for achieving a goal or maintaining a condition
- Consisting of *states* and *transitions* between them
- States contain Behaviours and Plantypes
- Initial states are tagged by *Tasks* and *Cardinalities*



Plans

- A plan is a recipe for achieving a goal or maintaining a condition
- Consisting of *states* and *transitions* between them
- States contain Behaviours and Plantypes
- Initial states are tagged by *Tasks* and *Cardinalities*
- Annotated by Pre-, Post-, and Runtime Conditions

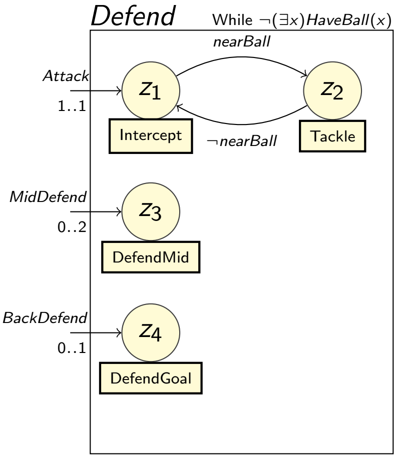


Plans

◇ Evaluated by a Utility function:

$$\mathcal{U}\colon \mathcal{L} \mapsto \mathbb{R}$$

$$\mathcal{U}_p(B) = 1 - \frac{Distance(Attacker, Ball)}{maxDistance}$$



Plantypes

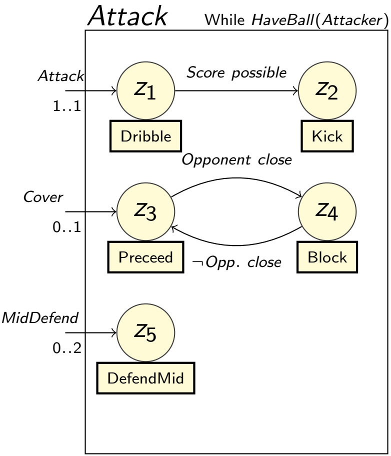
- A Plantype is a set of plans
- Provides a nondeterministic choice

Plantypes

- A Plantype is a set of plans
- Provides a nondeterministic choice

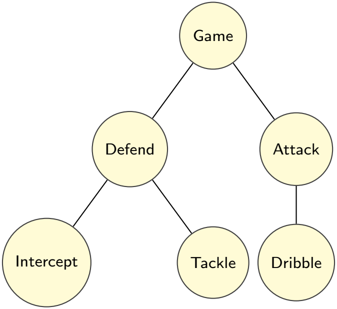
Example

Let Plantype *Play* contain *Defend* and *Attack*.



Plan Hierachy

Plans, States, and Plantypes span a (finite) tree structure:



Example

- Tackle occurs in a state in Defend
- Hence Tackle is a plan possibly executed in Defend

Task Allocation

A task allocation is

- computed locally by each agent
- acted upon *before any communication takes place*
- assigning all relevant agents to a plan p in a given plantype P, such that adopting the allocation will:

Task Allocation

A task allocation is

- computed locally by each agent
- acted upon *before any communication takes place*
- assigning all relevant agents to a plan p in a given plantype P, such that adopting the allocation will:
  - Satisfy plan cardinalities

Task Allocation

A task allocation is

- computed locally by each agent
- acted upon *before any communication takes place*
- assigning all relevant agents to a plan p in a given plantype P, such that adopting the allocation will:
  - Satisfy plan cardinalities
  - Satisfy plan conditions

Task Allocation

A task allocation is

- computed locally by each agent
- acted upon *before any communication takes place*
- assigning all relevant agents to a plan p in a given plantype P, such that adopting the allocation will:
  - Satisfy plan cardinalities
  - Satisfy plan conditions
  - Not violate consistency constraints (e.g., an agent can take on only one task within a plan)

Task Allocation

A task allocation is

- computed locally by each agent
- acted upon *before any communication takes place*
- assigning all relevant agents to a plan p in a given plantype P, such that adopting the allocation will:
  - Satisfy plan cardinalities
  - Satisfy plan conditions
  - Not violate consistency constraints (e.g., an agent can take on only one task within a plan)
  - Maximise the plan's utility

General Alica

So far, we were limited to an essentially *propositional* language.

- Expressivity: Propositional ALICA is as expressive as finite-state automata.

General Alica

So far, we were limited to an essentially *propositional* language.

- Expressivity: Propositional ALICA is as expressive as finite-state automata.

General Alica

So far, we were limited to an essentially *propositional* language.

- Expressivity: Propositional ALICA is as expressive as finite-state automata.
- Reusability: No distinction between algorithm and goal in behaviours.

General Alica

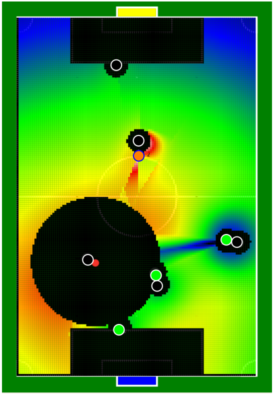
So far, we were limited to an essentially *propositional* language.

- Expressivity: Propositional ALICA is as expressive as finite-state automata.
- Reusability: No distinction between algorithm and goal in behaviours.

Many behaviours for (almost) the same problems

CornerOppPosBlockGoal, CornerPosLongReceiver, ThrowInOppPosBlockSide, GoalKickPosSecondDefend,. . .

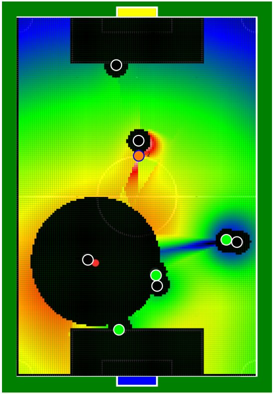
General Alica



Goals:

- Computational Effort
- Noisiness
- Coordination

General Alica



Goals:

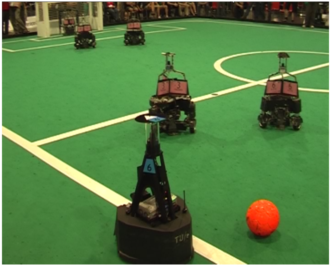
- Computational Effort
- Noisiness
- Coordination

Solution:

- Declerative variable definition for plans/behaviours
- Distributed constraint solver for coordination

Summary

- An expressive language for modelling teamwork
- Allows for swift reaction to changing situations by individual agents
- Robust against communication issues



Thank you for your attention!

Any questions?