Behaviour Coordination for Cooperative Multi-Robot Systems

Bob van der Vecht
vdvecht@ai.rug.nl
1154249

July 2004

Supervisors:
Prof. Dr. Lambert Schomaker
University of Groningen, Groningen, The Netherlands

Prof. Dr. Pedro Lima
Instituto Superior Técnico, Lisbon, Portugal

Artificial Intelligence
University of Groningen

Abstract

This report introduces a general formulation of relational behaviours for cooperative real robots and an example of its implementation using the pass between soccer robots of the Middle-Sized League of RoboCup. The framework supports explicit teamwork between two team mates. This implies that both participants know from each other that they are committed to the relational behaviour and that they will not quit without informing the other team members first. The formulation is based on the Joint-Commitment Theory by Cohen and Levesque [2]. The implementation of the pass concerns furthermore the development of two primitive behaviours, the *intercept* and *aimAndPass* behaviour, and the introduction of heuristics to support coordinated execution. This implementation is supported by past work on soccer robots navigation. Results of experiments with real robots under controlled situations (i.e., not during a game) are presented to illustrate the described concepts. The framework provides an easy way for implementing relational behaviours and takes care of synchronized execution.

Acknowledgement

A short word of thanks for those who have been helping me during this graduation research.

The main part of this project has been done at the Institute for Systems and Robotics, of the Instituto Superior Técnico in Lisbon Portugal. I would like to thank my supervisor overthere, prof. dr. Pedro Lima, for all his advice and efforts. It is too much to mention all members of the SocRob-group personally, so I thank you all in general for the cooperation in both professional and friendly way.

Then, from the Department of Artificial Intelligence of the University of Groningen, I got good advice and guidance from prof. dr. Lambert Schomaker, my internal supervisor. I also want to thank dr. Rineke Verbrugge for revising this report. And I should mention the other students in room 155 and the ping-pong table overthere.

Of course I have family and friends who are absolutely valuable at all moments.

Addresses:

Institute for Systems and Robotics,
Instituto Superior Técnico
Av. Rovisco Pais 1
1049-001 Lisbon, Portugal
http://www.isr.ist.utl.pt/

Department of Artificial Intelligence,
University of Groningen
Grote Kruisstraat 2/1
9712 TS Groningen, The Netherlands
http://www.rug.nl/ai/

Contents

Chapter 1

Introduction

This report presents research done for a master degree in Artificial Intelligence. Research areas of Autonomous Systems and Multi-Agent Systems are combined in this project. The aim of this project is to create cooperative behaviours among team mates and it involves an implementation in a team of real robots of a RoboCup Middle-Sized League soccer team. Showing cooperation among robots in a team is probably one the top goals of RoboCup related research. Furthermore, it is desirable to formulate cooperative behaviours within a formal framework extensible to applications other than robotic soccer. Surprisingly, not many references to work on these two topics can be found in the RoboCup-related literature or in other publications referring cooperation among *real* robots.

In this report a general formulation of relational behaviours for cooperative real robots is introduced. Most of the report describes an example of implementation of this formulation using the pass between soccer robots of the Middle-Sized League (MSL) of RoboCup. The formulation is based on the Joint-Commitment Theory [2], and the pass implementation is supported by past work of the ISocRob team on soccer robots navigation [6].

In chapter 2 the aim of this research is introduced and motivated. Chapters 3 until 5 show the construction of the framework for relational behaviours. Examples of the use of the framework will be given in chapter 6 and the explanation of the pass example is given chapter 7. Results and a discussion of the project are to be found in chapters 8 and 9. A brief description of the research presented here can be found in [19].

Chapter 2

Problem Description

2.1 Motivation for this Project

An important capability of humans and animals is to work together in order to achieve certain goals. Different kinds of cooperation can be distinguished. One can find instinctive cooperation for example in colony behaviours, where animals execute different tasks separately, and the result clearly looks like teamwork. Another example can be seen as planned cooperation, like people who agree on helping each other with an action. In the first example, participants do not need to have the notion of a common goal, whereas in the latter they do. Gärdenfors argues that in the example of planned cooperation the participants need to have a common representation of the goal in their internal world model [8]. In order to coordinate the inner world models of the individuals, symbolic communication is required. In that case participants can explicitly agree on what to do.

Imagine two people who need to move a heavy couch. They need each other to do so. They need to lift the couch at the same moment, so they should communicate about when to start. Then when they are carrying the couch they will know from each other that the other does not stop unexpectedly. The task requires a highly coordinated execution. At certain moments it is essential that the participants are at the same level of execution.

The project described here addresses this kind of planned cooperation. The term used in this report for planned cooperative behaviours is *relational behaviours*. This term is introduced by Collinot and Drogoul, [3] and [7], in their Cassiopeia methodology, designed for multi-agent organizations. In other literature it is often referred to as *cooperative behaviours* or *teamwork*.

Definition: A *relational behaviour* contains a set of individual behaviours, that are to be executed coordinated by a set of agents from a cooperative team. The participants pursue a joint goal and communicate with each other to achieve the required coordination. This implies an agreement between the participants, referred to as a *commitment*.

Much research on agent cooperation has been done in simulated environments. In

this project we have explicitly chosen for a real robot implementation, in order to test theories about cooperation and to get more understanding in the possibilities for cooperation in embodied agents.

2.2 Problem Specification

The aim of this project is to create relational behaviours between two autonomous agents in the RoboCup Middle Sized League (MSL) environment. The agents will on forehand agree with executing the behaviour. To achieve this a framework for relational behaviours is to be created. The framework will be constructed around the behaviour of giving a pass between two soccer robots. The framework should not be restricted to the pass, it should provide an easy way of developing and testing other relational behaviours as well.

The relational pass behaviour in soccer is a dynamic behaviour, which involves two agents, and has to be executed with a high precision of timing. Because of these properties, coordinating the behaviours of embodied agents is one of the key issues in this research. The question may arise why a (for human beings) reactive behaviour of giving a pass is being implemented in the framework for explicitly planned cooperation, as the above mentioned example of moving a heavy couch. The first answer is that giving a pass for robots is not as automatically as reactive as it is for humans. Furthermore, the pass behaviour is a non-innate routine that has to be learned explicitly by the agents. These learned forms of cooperation will allow the players to work towards specific game situations, from which a cooperative behaviour is a useful continuation. This feature can be used in further strategic team play.

When the pass behaviour is started, the two participants have their own role. One of the agents has the ball, he is referred to as the *kicker*. He will try to kick the ball in the direction of the second agent, the *receiver*. The receiver tries to intercept the ball. In order to accomplish a pass successfully, two components should be working well. The participants have to execute their individual behaviours, and this has to be done synchronized with the each other's behaviour. So the first component is the cooperation and communication between both players. This should be taken care of by the framework for relational behaviours, mentioned above. The second part of a successful pass concerns the individual skills of the players. The kicker should pass the ball in the desired direction, and the receiver needs to intercept and control the ball.

This research deals with both components. First the cooperation aspect will be explained in a general way for relational behaviours between two agents, chapters 3 until 5. Second the development of individual behaviours specific for the pass will be explained in chapters 6 and 7.

2.3 Project Environment: RoboCup

This project has been done at the Socrob research group at the Institute for Systems and Robotics, at the Instituto Superior Técnico in Lisbon. The Socrob group par-

ticipates in the RoboCup Middle Sized League (MSL) with the robot soccer team ISocRob. RoboCup, introduced by Kitano [11], is an international research project, in which theories of several research areas are developed and implemented on a platform of robotic soccer. The research is done at different levels, varying from a simulation league, to humanoid robots. The aim of the project is to beat the human world champion of soccer with a fully autonomous humanoid robotic team by the year 2050.

RoboCup provides a dynamic, real world environment for doing scientific research. At the same time the environment contains a restricted number of objects and variables, which makes it possible to test theories more easily than in the normal world. In the MSL league of RoboCup two teams of fully autonomous robots play soccer on a soccer field. At the moment of this research, the field was 9 by 4.5 meters, and the ISocRob team had four Nomadic Super Scout robots. Current rules and regulations of the RoboCup MSL are described in [4]. In the RoboCup MSL soccer world, the only existing objects are a green field with white lines, one yellow and one blue goal and four corner posts. Furthermore there is one orange ball, and the robotic agents of the two teams, which make the world dynamic and non-deterministic. The agents have to play soccer fully autonomously. All information must be collected using the robot's own sensors or by communicating with each other. Research within RoboCup addresses for example image processing, self-localization, decision-making, planning, and multi agent cooperation.

2.4 Research Question

The aim of this research is make two robots in the RoboCup Middle Sized League successfully execute a pass. This specific condition is an example of the more general problem, where two agents with a common goal turn to cooperation to achieve it. This research focuses primarily on a general framework for relational behaviours between two agents, and secondly on the individual behaviours for a pass, and the execution of it. The research questions to be answered are:

- How can we develop a general framework for teamwork by two robots, such that it allows an easy way of defining and implementing new coordinated behaviours?
- How can we define the logical conditions for a dynamic pass between two robots? How accurately and efficiently can two robots execute a pass during a RoboCup Middle-Sized League soccer game?

2.5 Why This Topic Has Scientific Significance

The RoboCup project is an interesting research area for Artificial Intelligence. It offers the opportunity to combine many different disciplines of AI, in a dynamic, real time environment. Theories that are developed and implemented are not only restricted to robotic soccer, but have application opportunities in many robotic environments. The subject of relational behaviours will appear in all areas where

cooperation between robots is involved, one can think of space expeditions or rescue operations.

For humans and animals, cooperation shows to be a strong ability for achieving goals. Examples are the above-mentioned case of two people moving a heavy couch, or lions who work together as a team in order to catch their prey, [17]. In this research it is attempted to let robots use these forms of explicit planned cooperation, in order to execute behaviours as a team, not only as individual. Transforming cognitive capacities of living creatures to systems of embodied agents has always been one of the goals of artificial intelligence research.

Much research to cooperative behaviours has been done in simulated environments. This project is especially engaged in implementing theory in the real world. We will have to deal with real world problems like noise in sensor information. The project will give us more understanding about the aspects of synchronized behaviour execution and timing in a team of robots. Even theoretically simple issues can be difficult in a real world environment.

Within the RoboCup environment, this project tries to take a new step in team strategies and cooperative play. Each year the RoboCup rules are adapted and changed towards new challenges. In this view, cooperative behaviours, like passes, will be more important in the future, and this research tries to give understanding about how precisely a simple pass can be executed at this moment. Bottlenecks will arise, which can direct further research for the future.

Chapter 3

Theoretical Background

3.1 Related Research

At stated in the introduction, this project focuses on planned cooperation. Participants start the execution of a relational behaviour after they have created a common representation of the goal. Other researchers have tried to create relational behaviours and cooperation in multi-agent systems. Roughly the researches can be divided in two groups; behaviour based approaches and logical approaches.

Behaviour based research of teamwork often involves simulated environments, for example the RoboCup simulation environment. Examples of those researches are from Matsubara [13] and Pagello [14]. The first trained a neural network in order to learn the agent choose between a goal shot and a pass. Feedback from a trainer is used for learning. No information is given about a relational commitment or about the communication. Pagello in his research tried to extend behaviour based approaches for individual robot control, to emergent cooperative behaviours of a team. Observing the behaviour of other agents is seen as a non-intentional kind of communication, referred to as *implicit* communication. By identifying actions of other agents, and inserting them into an evaluation function of a world situation, decisions about cooperation are made. There is no form of commitment between players. Since there is no guarantee for inner world model coordination between the agents, the relational behaviour may be kept by one of the team mates even if the other has to withdraw its relational behaviour. The same is the case in Yokota et al., [20]. They use explicit communication to achieve cooperation and synchronization in real robots, but again no commitments are set up.

Within logic-based architectures for decision making it has been tried as well to achieve cooperation between team members. An example of a tool that enables the development of applications based on the Joint-Commitment Theory [2] has been thoroughly reported by Tambe in [18]. They have developed the *STEAM* framework, which provides a method for creating cooperation in a team of autonomous agents. STEAM can handle various team sizes and is strong in failure recovery and the implementation includes communication protocols for the Joint-Commitment Theory. It has been tested in simulated environments for military operations and robotic soccer.

Another framework is presented in recent work of Kaminka [10]. The BITE architecture provides services for automated coordination and collaboration. A flexible framework has been created which supports different protocols for coordination, communication and control. Their implementation involves formation control in real robots.

The research described here, is related to some of the above mentioned subjects. A logical approach is used to create a framework for planned cooperation and the implementation involves embodied agents.

3.2 Decision Making

In order to create a framework for relational behaviours, it is necessary to take a closer look to decision making within a robotic team of agents. Where can such a framework be constructed in a complete decision model? During a game of the RoboCup MSL, all robots have to gather their own world information using there own sensors, or by communicating with team mates. With their knowledge about the world, the robots should decide individually what to do at each point in time. At the same time they should be aware of the fact that they are playing in a team, and thus have the possibility to work together to achieve common goals.

An architecture for behaviour coordination and decision making for a team of agents is proposed by Collinot in [3] and applied to a robotic soccer team by Drogoul, [7]. It considers three types of behaviours: organizational, relational and individual.

- **Organizational behaviours**: those concern decisions involving the whole team, e.g., in robotic soccer, player role selection such as defender or attacker.
- **Relational behaviours** concern more than one player. Commitments among team mates are established here.
- **Individual behaviours**: at this level, primitive behaviours are selected, and motor commands are used to influence the environment in the desired way.

Depending on the role selection at the organizational layer, possible relational behaviours will be enabled or disabled. At his turn, a relational behaviour can enable or disable individual behaviours. A framework as is to be constructed in this project should be implemented in the relational layer of this decision architecture, and it should interact with the individual level.

3.3 Communication

In a relational behaviour the agents should act synchronously, and therefore communication is necessary. Summarizing the works described in paragraph 3.1, we can distinguish two ways of communication that are used to achieve cooperation; implicit and explicit communication.

Using implicit communication, no information will be directly sent, but the agents have to use their own observations to detect and identify the other's behaviour. This implies the more reactive approach of the two cooperation ways.

There is no guarantee for similarity of the common goal representation of the participants. A second agent does not necessarily need to know that the first agent has started the relational behaviour. For example in the situation of giving a pass in soccer the kicker can start to pass without the receiver expecting the pass. In the same way, the agents, use their own sensor information for determining the progress of a relational behaviour. This makes synchronization of the individual behaviours harder. Successful implementations of relational behaviours using implicit communication are mostly found in simulated environments.

With explicit communication there will be direct communication between the two agents. Explicit messages will be sent. At a certain point they agree to start the relational behaviour, and they will tell each other their progress. The same way they will tell each other when the behaviour has been ended. This facilitates the coordination of the inner world models of the participants, which is a requirement for successful planned cooperation.

Discussing different possibilities for communication we come to the conclusion of using explicit communication. With this form of communicating the robots have more certainty about the status of the other within a commitment, than when they use implicit communication. In a simulator environment it is possible to communicate implicitly, but in our real world implementation we foresee too much uncertainty to deal with. Our aim, to create planned cooperation for which we need a common representation of the goal and mutual knowledge about the progress of a task, implies that explicit communication is most useful. This choice is supported by the observation that the team coordination has to be kept as high as possible. The agents need to be sure what the other is doing. A Multi Agent theory that can be used for relational behaviours and at the same time supports our aim of certainty is the Joint-Commitment Theory of Cohen and Levesque [2]. This theory has been implemented in previous work by Tambe [18].

3.4 Joint-Commitment Theory

Cohen and Levesque make a clear distinction between coordination in multi agent systems and teamwork. While common traffic is clearly a coordinated system, it is not teamwork. Driving in a convoy however, is seen as teamwork, because the involved agents share a common goal and need to cooperate in order to achieve their goal.

The Joint-Commitment Theory is based on the idea that agents with a common goal join in a commitment. They inform each other about the progress to achieve the goal. To formalize the theory, the following definitions are given in [2]:

Definition: Persistent Goal (PG)
An agent has a persistent goal relative to q to achieve p if the following three rules hold:

- The agent believes that p is currently false.
- The agent wants p to be true eventually.

- it is true (and the agent knows it) that (2) will continue to hold until he comes to believe either that p is true, or that it will never be true, or that q is false.

Definition: Weak Achievement Goal (WAG)
An agent has a weak achievement goal relative to q and with respect to a team to bring about p if either of these conditions holds:

- The agent has a normal persistent goal to achieve p.
- The agent believes that p is true, will never be true, or is irrelevant (that is, q is false), but has as a goal that the status of p will be mutually believed by all the team members.

Definition: Joint Persistent Goal (Joint Commitment)
A team of agents have a joint persistent goal relative to q to achieve p just in case:

- they mutually believe that p is currently false;
- they mutually know they all want p to eventually be true;
- it is true (and mutual knowledge) that until they come to mutually believe either that p is true, that p will never be true, or that q is false, they will continue to mutually believe that they each have p as a weak achievement goal relative to q and with respect to the team.

A joint goal can consist of a set of subgoals or it can imply a sequence of actions for the participants. According to the theory, when the team jointly intend to achieve a goal in a commitment, the agents in any of the steps will jointly intend to do the step relative to a larger intention. From this feature, it can be concluded that a commitment can contain several parts, which will be executed stepwise.

Summarized and applied to relational behaviours within a robotic environment the following can be said. Predefined logical conditions can establish a commitment between two agents. Once a robot is committed to a relational behaviour, he will pursue this task until one or more conditions become false, or until the goal has been accomplished. Any success or failure has to be reported to all participants in the commitment, before the agent quits the commitment.

In the work described here, the relational behaviour that is to be executed, is known to all participants. If predefined conditions are true, the participants have as a joint goal to finish the relational behaviour successfully. They know the sequence of actions to take, or the subgoals that are to be achieved. The initiative for a relational behaviour is taken by one of the agents. If predefined conditions are valid, he does a *request* for the relational behaviour. A potential partner checks if the conditions to *accept* are valid. If so, the commitment is established. During the execution of the commitment the changing environment can lead to failure or success at any time. In that case the commitment will be ended.

In order to make the use of the Joint-Commitment Theory more clear, examples will be given later on. First the framework around the theory will be explained then it will be applied to the relational behaviour of giving a pass.

Chapter 4

The Framework

4.1 Commitment Definition

The theoretical base of a relational behaviour has been defined now. A commitment will be set up following predefined conditions. The agents should communicate their behaviour progress to each other. To make an implementation possible, we must clearly define a commitment. The communication also has not been defined yet. In this part of the report it will be explained how a commitment is to be defined and implemented exactly.

The tasks for an individual agent within a relational behaviour can be seen as a state machine. For a simple example of a relational pass behaviour the state machines for both participants are shown in picture 4.1.

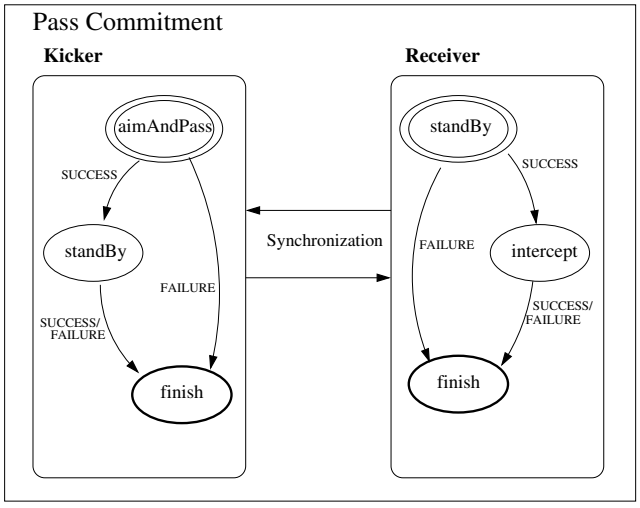


Figure 4.1: *Simple example of the pass commitment, shown in a finite-state machine*

4.1.1 Commitment Phases

As explained before, the decision model of an agent selects first a relational behaviour and knowing this decision, it selects the individual primitive behaviour. This primitive behaviour handles the interaction with the world. During the execution of a relational behaviour, the decision model will select a primitive behaviour concerning the commitment. But during setting up and ending a commitment, a robot that is not committed to the relational behaviour model should not choose the primitive behaviour belonging to the relational behaviour.

In general, within a commitment three phases can be distinguished: *Setup*, *Loop* and *End*. In the *Setup* and *End* phase, the decision model will not select the relational behaviour, so it will be ignored during the primitive behaviour selection. Only in the *Loop* phase participants will select primitive behaviours concerning the commitment in order to achieve their joint goal. In Table 4.1 it is shown which commitment is selected by the relational behaviour layer of the decision model, during all phases of the commitment.

Table 4.1: *Commitment validity during all phases of the pass commitment by the decision model. The commitment only influences the individual behaviour selection in the* Loop *phase.*

Phase:	*Setup*	*Loop*	*End*
Agent1:	none	pass	none
Agent2:	none	pass	none

4.1.2 Commitment States

In order to achieve synchronization during the execution of a relational behaviour, a commitment can be split up in steps. By splitting the commitment in several states and linking each of the states to a set of primitive behaviours for both participants, the participants will synchronize their actions at the beginning of each new state.

So first the commitment has been split up in phases. Now the phases will be split up in states. In the *Setup* phase a commitment *request* is to be done and the commitment has to be *accepted*. Next, in the *Loop* phase, we will execute the relational behaviour. The *End* phase is divided in a *done* and a *failed* state, only to make this distinction.

- *request* and *accept* in the *Setup* phase.
- *state1* until *stateN* in the *Loop* phase.
- *failed* and *done* in the *End* phase.

For all commitments, the states in the *Setup* and *End* phase will be the same, only the states in the *Loop* phase will change. When the commitment runs as planned, the commitment states will be run through sequentially, from *request* until *done*. An error at any time can lead to the state *failed*. Table 4.2 shows a look up table for the decision model to select primitive behaviours during all states of a commitment.

When in Table 4.2 a commitment state does not lead to a behaviour from the specific commitment, it is indicated with "—".

Table 4.2: *Lookup-table for primitive behaviours by the decision model in all commitment states during a commitment. W, X, Y, Z are primitive behaviours.*

Commitment Phase:	Setup		Loop		End	
Commitment State:	*request*	*accept*	*state1*	*stateN*	*done*	*failed*
Agent1	—	—	W	Y	—	—
Agent2	—	—	X	Z	—	—

A commitment will be started only after evaluating the logical conditions that lead to the *setup* and the *accept*. By changing those conditions different commitments can be achieved. Furthermore, when a commitment is to be defined, the conditions for switching from state to state have to be defined, in such a way that the commitment will run from the request state until done or failed. Different versions of a relational behaviour can be created by changing these switch conditions. Finally the individual behaviours that are linked to each commitment state can vary between different commitments. Specific examples of lookup-tables for primitive behaviours will be given in chapter 6.

4.2 Synchronization

The agent's decision model runs continuously in a finite state machine, and selects every iteration first a relational and then an individual behaviour. By dividing a relational behaviour into several states, and attempting to keep both agents in the same state all the time, synchronization of the actions of the participants is achieved. The commitment states are defined in such a way that they can be run through sequentially. We will define the execution of a commitment in such a way that each iteration it will be checked in the decision model of the agents whether the world situation allows the agent to step to the next state of the state sequence. The agent communicates a state change always to the commitment partner. Finally if an agent notices that his partner has moved on to the next commitment state, he will follow him to this new state. So the loop during the execution of the commitment will be:

- Synchronize commitment state with the state of the partner, if partner has moved on to a next state.
- Switch to new state yourself, if the situation allows you to do that.

4.3 Conclusion

By splitting the commitment instantiation in phases and states, we created a way to achieve synchronization using explicit communication. A relational behaviour can be created by defining the following things:

- commitment states
- setup conditions
- switch conditions
- related individual behaviours in each commitment state

New commitments or other versions of commitments can be created under the same framework just by redefining one or more of the above listed things. Below in Table 4.3 is stated how commitment is executed from beginning until end.

Table 4.3: *Stepwise actions during all phases of the commitment of the execution of relational behaviour R.*

Setup	- When setup conditions $<c1,..,cN>$ are true, agent1 does a setup *request* for relational behaviour *R*. - When setup conditions $<d1,..,dN>$ for agent2 are true, he sends the *accept* of R. His setup conditions include a *request* for R. - At this point the commitment between agent1 and agent2 has been established.
Loop	- *R* is to be executed in states *s1* until *sN*. Switch conditions to state *s1* includes acceptance of R. - An agent in state *si* switches to next state *sj*, when switch conditions $<e1,..,eN>$ hold, or when he notices that is partner moved to state *sj* - State *done* or state *failed* are entered following switch conditions or following the partner as well
End	- When the participants know they are both in state *done* or state *failed*, they quit the commitment.

Chapter 5

Implementation

5.1 Implementation Platform

The framework for relational behaviours as explained above has been implemented in the RoboCup MSL team ISocRob, of the *Socrob* research group at the *Institute for Systems and Robotics* in Lisbon. The robot team consists of four Nomadic Super Scout II robots.

The software architecture used in the robots, contains several micro agents, running in separate threads. Each micro agent handles a different part of the robot control [12]. Examples are the micro-agent *Vision*, which does image grabbing and image processing, and *Control*, which executes the basic behaviours selected by a decision agent.

Decision making and behaviour coordination is done in a micro agent called the *Logic Machine*, explained in [16]. It is based on the previous mentioned work of Drogoul [7], and divides the behaviours in three levels; organizational, relational and individual. In the *Logic Machine*, only the organizational and the individual levels were implemented to make the robots play soccer. In the work described here, the relational layer is added and the individual layer will be used as well.

The *Logic Machine* uses first order logic statements for a fast evaluation of the current world situation. At the individual behaviour layer, a hierarchical list of basic behaviours is checked for pre-conditions. The first one that has all pre-conditions true, will be selected. Those pre-conditions for the basic behaviours differ between player roles. Figure 5.1 shows a hierarchical list of basic behaviours that can be used for playing soccer.

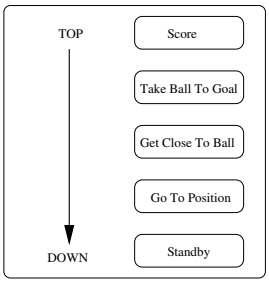


Figure 5.1: *Hierarchical list of individual behaviours in a soccer game when robot is not involved in a relational commitment.*

The decision layers are processed sequentially. This means for the robot that he first chooses a player role, next he selects a commitment, and the individual behaviour is selected after knowing the robot's role and commitment.

Blackboard

The world model for each individual robot is stored in a memory called the *Blackboard*. The *Blackboard* is a data pool accessible to the above mentioned micro agents. Variables contain world information, like field size and own position. When a variable is updated by one of the micro agents, a time stamp is added, which makes it possible to check validity based on recency. Two types of variables are used, so called local and global ones. Local variables store information that is especially relevant for the robot where it is stored or that information will not change, for example the field size. Global variables are automatically broadcasted to all other agents when a value is set. For example the robot position is associated with a global variable, and this way the robots know each other's position. One way to send information to other robots is by using a global variable.

5.2 Implementation of the Framework

The relational behaviour framework as presented above has been implemented in the *Logic Machine*. The loop, which continuously evaluates the world model in order to make a decision, has been constructed in GNU Prolog. Decisions will be made by checking predicates, whose truth value depends on other predicates and predefined conditions.

First the distinction is made between the situation that the robot is already involved in a commitment, and the situation that this is not the case. In the first situation, the agent is executing the loop given in section 4.2, so each iteration first synchronize the own commitment state with the commitment state of the partner and then try to switch to a new commitment state. If the robot is not involved in a commitment, the setup conditions for all possible relational behaviours will be checked. To give an idea of the implementation, here are the main rules for the relational behaviour layer:

```
chooseCommitment( DefCommitment ) :-
    getCommitmentInfo( Commitment, MyRole, MyState, PartnerRole, PartnerState ),
    synchronizeStates( Commitment, MyRole, MyState, PartnerRole, PartnerState ),
    switchToNewState( Commitment, MyRole, MyState, NewState ),
    selectCommitment( Commitment, NewState, DefCommitment ).

chooseCommitment( none ) :-
    setupConditions( _ ).
```

Several functions to recall or change variables of the *Blackboard* memory and functions to handle the communication, have been created in program language C. Those functions can be called by the prolog *Logic Machine* as well. Via pointers, variable values can be passed through from C to Prolog.

5.3 Synchronization Variables

Variables have been defined, that store all the commitment information in the *blackboard*. For the decision making, the agent checks whether he is involved in a commitment. If so his commitment role and commitment state are used for selecting a behaviour. This information is stored in local blackboard variables. For the communication between the agents about each other's state, four global variables have been created for each relational behaviour.

Local:

`current.commitment`
`current.commitment.role`
`current.commitment.state`

Global:

`commitment.agent1`: contains the identity of agent1
`commitment.agent2`: contains the identity of agent2
`commitment.agent1.state`: contains the commitment state of agent1
`commitment.agent2.state`: contains the commitment state of agent2

One of the issues that has to be taken into account is the restricted communication that is available. The robots communicate via a wireless network with a restricted capacity. Experience from the past showed the importance of minimizing the communication. In order to reduce communication to a minimum, a variable only needs to be sent to the partner if it is changed. So the two identity variables only need to be set in the *Setup*-phase and should be cleared in the *End*-phase. The state variables also change during the commitment, but maximally as many times as the number of states of the commitment. As stated in chapter 4, the state-variables are synchronized in every iteration of the decision module. This means that if your partner has moved on to a next commitment state, you will change to the same state.

5.4 Imperfect Communication

During first tests of the working of the framework, it appeared we had to deal with a real world situation, of which the framework had not taken care yet, namely imperfect communication. When a message was sent to another agent it appeared not to be obvious that it arrived at the other agent. Previous use of the communication concerned mainly the robot positions and the ball position. Every time a new position had been estimated, it was broadcasted by setting a global variable. If a new value somehow was not noticed by another agent, reasoning could just be done with the old value without many problems. Also the frequency of new position values made the implication on the robot behaviours almost zero.

However, in the execution of relational behaviours it is really important that the agents know from each other in which state of the commitment they are, since the synchronization is based on that. In order to reduce broadcast intensity it was decided to send variables only when the value was changed. An agent who takes the initiative to switch to a new commitment state, needs to know that his state change is known by his partner. A confirmation is obtained by the incoming message that the partner has synchronized his state. Note that an incoming message of a partners state change implies that the partner already is in that state, since the agents use their own, locally kept commitment state for further decision making.

The communication protocol, developed to handle the imperfect communication problems, takes care of the cases that:

1. The communication fails completely.

2. The broadcasting of a single message does not necessarily mean arrival.

The first case is solved by a maintaining the contact by regularly sending the own commitment state following a defined time period. By defining a timeout variable, whose value has to be larger than the repeat period, the *"Lost Contact"* situation is defined. Then the commitment has to be ended. The second problem is now taken care of by simply repeating a switch to a new commitment state, until a confirmation is received, or until a maximum number of repetitions is reached. The counter of repetitions is set back to zero each time an incoming commitment state of the partner is a previous one of the own commitment state. The maximum number of repetitions can be set quite low, since information loss is more exceptional than regular. In the theoretically possible situation in which the message still did not arrive after the repetitions, the sender will notice that, because the rule of maintaining contact makes that the partner will send his commitment state. If the partner is still sending the previous state, the repetitions counter of the first agent is set back to zero and the new state will be sent again.

The above explained variables are shown in Table 5.4. They are defined as local variables in the *blackboard*. Different values have implications for certainty about message arrival, and for broadcast intensity. In the case of perfect communication, both mentioned time variables can be set to unlimited, and the number of repetitions to zero. Table 5.4 shows two other time variables. One defines the time period a commitment request holds, the other defines a minimum period between two commitments.

Table 5.1: *Blackboard variables, used for the communication by the relational behaviour framework.*

Name	Dim.	Meaning
contact.repeat	Sec	To keep contact during a commitment, the current commitment state is sent regularly to the partner. This value is the number of seconds between two repetitions.
contact.timeout	Sec	If the commitment state of the partner hasn't been updated for this number of seconds, the commitment will be ended ("lost contact"). This value should be bigger than **contact.repeat**.
repeat.max	Nr	A state change with a single variable broadcast does not always arrive at the other robots. As long as no confirmation of a state change arrived, the agent will repeat his new commitment state. This is the maximum of repetitions.
request.timeout	Sec	If a request hasn't been accepted after this number of seconds, if will be removed.
setup.minTimeDiff	Sec	To avoid noise in values, there is a minimum time difference between the ending of the last commitment, and doing a request for the next one.

5.5 The Relational Behaviour Loop

The relational layer of the logic machine continuously evaluates the current situation, to select a commitment that is used for primitive behaviour selection. The relational component takes care of all communication between agents, in order to synchronize the behaviour. Here the loop that is executed every iteration is summarized. First the distinction between three situations is made. Here is stated what is done in all situations:

1. No commitments are allowed:
   - Selected commitment is "none".

2. The agent is already involved in a commitment:
   - Should the commitment be ended?
   If TRUE, end the commitment.
   - Is the partner's commitment state ahead of own state?
   If TRUE, set own state to the partners state (= synchronization).
   - Is the partner in a previous state relative to the own state?
   If TRUE, repeat sending of own commitment state.
   - Is it possible to switch to the next state?
   If TRUE, switch to new commitment state and communicate this to partner.
   - Select commitment, knowing my commitment state.

3. The agent is currently not involved in a commitment:
   - Is it possible to setup a commitment (request/accept)?
   If TRUE, setup the commitment.
   - Selected commitment is "none".

5.6 Primitive Behaviour Implementation

When the relational layer of the *Logic Machine* has selected a commitment, the individual layer has to select a primitive behaviour. As explained before, each commitment state is linked to a set of individual behaviours for both agents. Those primitive behaviours have been defined in a hierarchical list, and by turn they are checked for predefined conditions. The *Logic Machine* already knows the player role and the selected commitment. Examples of basic behaviour definition is given below. Standard rule is: `basicbehaviour( behaviour, commitment, playerRole )`. In the example below, the player role is *defender* and the selected commitment is *pass*:

```
basicbehaviour( aimAndPass, pass, defender) :-
    commimentRole(kicker),
    commitmentState(prepare).

basicbehaviour( standBy, pass, defender) :-
    commimentRole(receiver),
    commitmentState(prepare).


basicbehaviour(standBy, pass.  defender ).
```

Chapter 6

Commitment Examples

6.1 When to use the Framework?

In this chapter examples will be given of the use the relational behaviour framework. First it will be analyzed when to use the framework; for what type of relational behaviour this framework is developed.

As stated in the introduction, the framework has been developed to create planned cooperation between two robots. It has been built around the relational behaviour of giving a pass in soccer. Typical of this behaviour is that it requires a high precision of timing and synchronization. In our implementation environment, it is not possible to achieve this synchronization by implicit communication, the agents can not achieve enough precision in timing only by observing the other, so explicit communication is used.

To illustrate the type of behaviour we want to create, imagine that two people are moving a heavy couch. They will have to communicate about when to start exactly and, maybe more important, when to stop. When one of them suddenly drops the couch, the other has a serious problem. So timing and synchronization is important. At the same time, they might not all the time be able to know each other's state without communicating, when the carrying gets to heavy for one of them, he has to tell the other, so they can put the couch down and take a break. The combination of required synchronization and impossibility to be aware of the whole situation asks for an explicit commitment. Using the presented framework for commitments, **both** agents know that both are involved in the relational behaviour and **both** agents inform each other about the progress.

A real robot implementation of the pass fits in this picture. The explicit communication allows the kicker to tell the receiver that the ball has been kicked. For the receiver it is very difficult to decide this fast enough by himself. In this section the implementation is shown of a simple version of the pass behaviour. After that another more dynamic version of the pass behaviour will be defined, which is fully integrated in the soccer code of the ISocRob team. Finally, to show that the framework is not restricted to the behaviour of giving a pass, a small game-like situation has been implemented.

6.2 Example: The pass behaviour

6.2.1 The Simple Pass Commitment

The use of the framework will be explained using the pass commitment that is shown in Figure 4.1 in Chapter 4. The following pass situation has been defined: a defender has the ball on its own half of the field and he will pass the ball to an attacker, who is on the other half of the field. The commitment states of this pass commitment are to be found in Figure 6.1. The state machines for both individual agents are shown, and the behaviour is divided in states. A new state starts at the point where synchronization is required.

The conditions to set up the commitment are shown in Table 6.2.1, as well as the conditions to switch to other pass states within the commitment. Finally table 6.1 shows the primitive behaviours for both agents during all pass states.

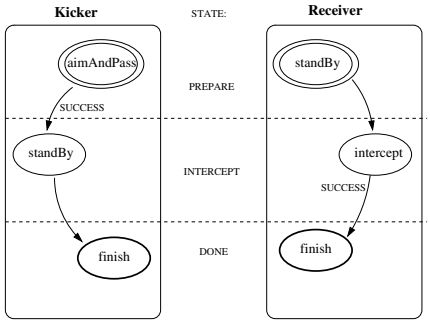


Figure 6.1: *Pass states for simple pass commitment.*

Table 6.1: *Primitive behaviour selection by the* Logic Machine *in the* Loop*-phase of the simple pass commitment.*

	Loop-phase	
Pass States:	*preparing*	*intercept*
Kicker	aimAndPass	standBy
Receiver	standBy	intercept

Table 6.2: *Conditions that allow the player to switch to a new pass state during the simple pass commitment.*

Switch to pass state:	Conditions:
request	player role is defender player has the ball player is on own half of the field
accept	**pass.kicker.state** is *request* player role is attacker player is on other half of the field
preparing	commitment role is kicker **pass.receiver.state** is *accept*
intercept	commitment role is kicker individual behaviour is aimAndPass ball has been kicked
done	commitment role is receiver intercept behaviour done with success
failed	any behaviour failure
none	pass state is *done* or *failed*

6.2.2 Example: The Integrated Pass Commitment

The version of the pass behaviour as defined in the previous section is more a demo version of the pass. During a game it might be unwanted that the receiver is only passively waiting for the ball. In a more dynamic version of the pass commitment, the receiver could be moving to a receive position instead of waiting in the standBy mode. Furthermore the role of the kicker, after he has shot the ball, is in the demo version to wait until the receiver has intercepted the ball, or failed to intercept the ball. One could argue that actually the pass commitment is finished, at the moment the receiver starts to intercept. From that moment on the receiver is the only one who is acting in the relational behaviour, and the actions of the kicker will not influence the result anymore. Thinking about this, a more dynamic version of the pass commitment has been defined. This example shows a version of the pass as it can be integrated in real robot soccer model.

Again we take a defender, located at his own field half, who wants to pass to a team mate on the other field half. Extra added is that the first agent selects the pass partner himself. Following heuristics, explained in section 7.4, he chooses the best receiver, and sends a request only for him. In the example given here, the receiver will always accept, he trusts the partner selection heuristic of the kicker.

Again picture 6.2 shows the state machines for both agents actions and the division in states. Setup and switch conditions are shown in Table 6.2.2 and table 6.3 shows the primitive behaviour selection for both agents during the commitment. This version of the pass has been fully integrated in the soccer model of the ISocRob team. In this pass, three heuristics are used as well, in order to improve the behaviour coordination. They will be explained in chapter 7 together with the individual

behaviours *aimAndPass* and *intercept*.

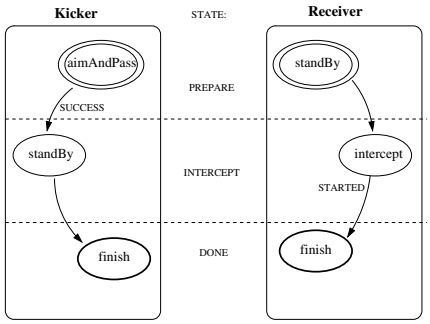


Figure 6.2: *Pass states for integrated pass commitment.*

Table 6.3: *Primitive behaviour selection by the* Logic Machine *in the* Loop*-phase of the integrated pass commitment.*

	Loop	
Pass States:	*preparing*	*intercept*
Kicker	aimAndPass	standBy
Receiver	goTo	intercept

Table 6.4: *Setup and switch conditions for the integrated pass behaviour.*

Switch to pass state:	Conditions:
request	player role is defender player has the ball player is on own half of the field pass heuristic has selected a possible partner
accept	**pass.kicker.state** is *request* pass request is addressed to player
preparing	commitment role is kicker **pass.receiver.state** is *accept*
intercept	commitment role is kicker individual behaviour is aimAndPass ball has been kicked
done	commitment role is receiver individual behaviour is intercept
failed	any behaviour failure
none	pass state is *done* or *failed*

6.3 Example: The *FirstHome* Game

A third example will be given here to illustrate the use of the framework. This example shows another application than the pass environment.

Maybe the reader is familiar with a game for children, in which the children are walking randomly through a room, while they are listening to music. In the room several chairs are placed. At a certain moment the music unexpectedly stops. Then the children have to find a chair as fast as possible. The one who is last at a chair, loses the game. The relational *FirstHome* behaviour presented here can be compared with this game.

Two agents are randomly moving though the field. Both agents have a predefined home position in the field. At a certain moment, for the agents randomly taken, they have to return to their home position. The first agent who arrives home, tells the other agent that he won. He waits until the second agent also arrived home then they start driving around again. Figure 6.3 shows the states of the FirstHome behaviour. Note that state *waiting1* and state *waiting2* are parallel states. One of them will be selected, the other one will be skipped.

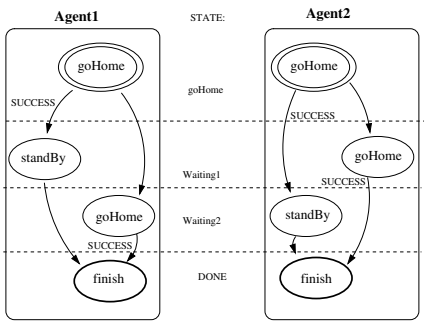


Figure 6.3: *States division for FirstHome commitment*

Table 6.5: *Primitive behaviour selection by the* Logic Machine *in the* FirstHome *behaviour.*

	Loop-phase		
Commitment States:	*gohome*	*waiting1*	*waiting2*
Agent1	goHome	standBy	goHome
Agent2	goHome	goHome	standBy

Table 6.6: *Setup and switch conditions for FirstHome.*

Switch to state:	Conditions:
request	player has the ball
accept	**firsthome.agent1.state** is *request*
gohome	commitment role is agent1 **firsthome.agent2.state** is *accept*
waiting1	commitment role is agent1 commitment state is *gohome* gohome behaviour ended with success
waiting2	commitment role is agent2 commitment state is *gohome* gohome behaviour ended with success
done	commitment role is agent1 commitment state is *waiting2* gohome behaviour ended with success
done	commitment role is agent2 commitment state is *waiting1* gohome behaviour ended with success
failed	any behaviour failure
none	commitment state is *done* or *failed*

Chapter 7

Individual Pass Behaviours and Heuristics

7.1 Introduction

After the *Logic Machine* has selected an individual behaviour, the behaviour will run in a *control* module. Here the world situation will be evaluated continuously, and motor commands will be calculated and sent to the robot in order to influence the environment. The world model of the robot can be seen as a map with all identified objects in a xy-coordinate system.

In the **standBy** behaviour, the robot will stay at the same position and will try to keep its front towards the ball. He will only rotate at its position. This behaviour has been implemented earlier in the Socrob project [12]. The other two behaviours, **aimAndPass** and **intercept**, had to be developed from scratch. The aimAndPass and intercept behaviour is explained in the next sections. Furthermore some heuristics will be described that are used in the pass behaviour to improve the coordinated execution.

7.2 AimAndPass

When the pass commitment is started, the *kicker* has the ball. He wants to rotate with the ball to a certain direction and then shoot. This has been implemented in the **aimAndPass** behaviour.

Rotating with the ball can not be done by just rotating at the same position, because the robot would lose the ball. In earlier research a controller has been developed to dribble with a ball [6]. The **aimAndPass** behaviour uses this controller. The dribble controller gets a goal position as argument, so in order to turn to the target direction a subgoal has to be defined as a position relative to the robot. This subgoal is calculated as shown in Figure 7.1. It is positioned on a fictional circle around the robot. The subgoal is the point on this circle in the target direction. The coordinates of this point will be given to the dribble controller.

$$subgoal(x, y) = (x_r + R\cos(\alpha), y_r + R\sin(\alpha)) \tag{7.1}$$

Where x_r and y_r are the robots position, R is the radius of the fictional circle around the robot and α is the angle between the target direction and the robot orientation.

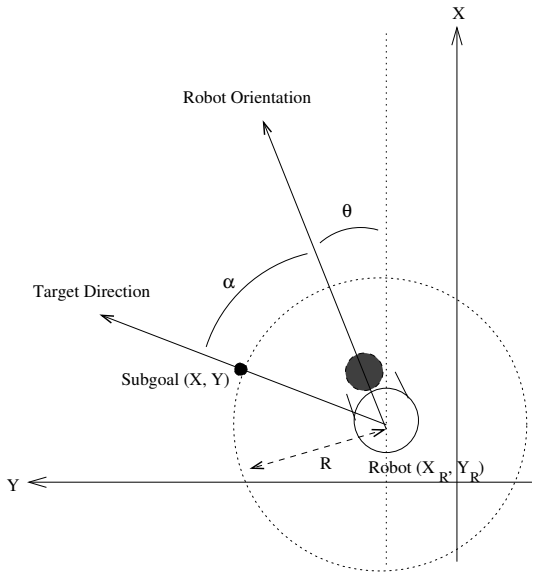


Figure 7.1: *Geometry in the aimAndPass behaviour.*

The dribble controller returns motor commands to the robot in order to move the robot to the subgoal. A new evaluation of the world situation then leads to a new subgoal and new motor commands. Note that the subgoal will always be calculated on a distance R from the robot, so the robot never reaches the subgoal. It is only used to force the robot to the desired direction.

7.3 The Intercept behaviour

In previous work in the Socrob project, the *getClose2Ball* behaviour has been developed. This behaviour returned the ball position to the navigation controller. While approaching the ball, the settings of the navigation controller were changed in such a way that the robot would reach the ball with a very low speed, in order to control the ball immediately. This behaviour was especially designed for approaching and getting control over a not moving ball. It appeared in no way to be able to intercept a moving ball. Reactions to any ball movement always came too late. In this section we show the design of a behaviour that does take the ball velocity into account.

7.3.1 Visual Servoing

The task of the *receiver* is to intercept the ball after it has been shot in its direction. The **intercept** behaviour controller has been designed to guide the robot to the ball, taking the ball speed into account. The literature on visual servoing has solutions for similar problems [5], [9], but only for a situation when the robot tracking the object is not moving within an environment cluttered with obstacles. In [1] a controller

for target-tracking using fuzzy logic has been developed, applied to ball tracking in Nomadic Scout robots, but again in a non obstacle environment. The MSL soccer field is a dynamic world containing moving objects. Therefore, the solution described here is based on a navigation algorithm described in [6] which can take the robot to a given goal posture while avoiding obstacles, using a modified potential fields method that takes into account the non-holonomic nature of the robot. The presented solution is based on iteratively estimating the fastest possible interception point. The coordinates of this interception point are calculated, after predicting the path of the ball and the path of the robot.

7.3.2 The Intercept Controller

Guiding the robot to the right position has been done by continuously estimating the interception point given the positions of ball and robot, and their predicted path. The interception point will be defined in xy-coordinates. At the interception point the following equation is true:

$$(x_b(t), y_b(t)) = (x_r(t), y_r(t)) \tag{7.2}$$

where $(x_b(t), y_b(t))$ is the estimated path of the ball as function of time, and $(x_r(t), y_r(t))$ the robot path. From (7.2), the interception point can be calculated as follows.

Figure 7.2 shows a ball with position (x_b, y_b) at time $t = t_0$ and velocity (v_{bx}, v_{by}), and a robot with position (x_r, y_r). The predicted path of the ball can be expressed as a function of time:

$$(x_b(t), y_b(t)) = (x_b(t_0) + v_{bx}t, y_b(t_0) + v_{by}t) \tag{7.3}$$

Here it is assumed that the ball has a constant velocity; the only variable is t.

For the path estimation of the robot, an extra assumption has been done. The robot will move with a constant velocity in a straight line towards the interception point. This implies that the orientation of the robot will not change during the movement to the interception point. Furthermore the real forward speed of the robot, v_r, will be replaced by an assumed constant average speed, v'_r. This has been done, because when the speed of the robot is very low, or even zero, the estimation of the interception point will have a very big error.

Now, the robot path can be expressed as function of time, using the robot's current position (x_r, y_r), its absolute forward speed v'_r, given in the x-direction, and the angle to the interception point α. In the equation, α is still unknown.

$$(x_r(t), y_r(t)) = (x_r(t_0) + v'_r t\cos\alpha, y_r(t_0) + v'_r t\sin\alpha) \tag{7.4}$$

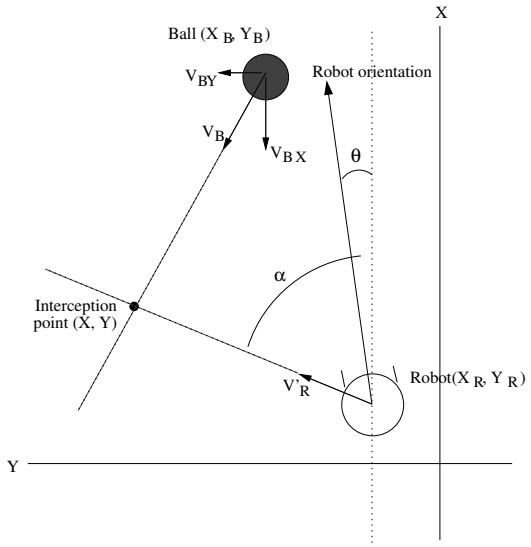


Figure 7.2: *Calculation of the interception point with interceptor robot velocity V_r' and ball velocity V_b.*

After replacing (7.3) and (7.4) in (7.2), $\cos\alpha$ can be eliminated from the x-component of equation 7.2 and t can be calculated from the y-component:

$$t = \frac{v_{bx}d_x + v_{by}d_y \pm \sqrt{v_r'^2(d_x^2 + d_y^2) - v_{bx}^2 d_y^2 - v_{by}^2 d_x^2 + 2v_{bx}v_{by}d_x d_y}}{v_r'^2 - v_{bx}^2 - v_{by}^2} \tag{7.5}$$

where $d_x = x_b - x_r$ and $d_y = y_b - y_r$; the distances between the ball and robot positions in respectively the x- and the y-direction at time t_0.

The t calculated above represents the time it takes for the ball and the robot to get to the interception point. The equation returns two possible values. For the interception behaviour, the smallest positive value will be chosen, since the ball is to be intercepted as fast as possible. By replacing t in the ball path from Equation (7.3), xy-coordinates for the interception point are achieved.

When the function returns two negative answers, the ball is too fast to intercept. Therefore it is possible to check such situations as well. The robot will not give up immediately in those situations. He will always keep chasing the ball, until the ball is outside the field. In those cases the xy-coordinates of an impossible interception point will be the ball position after a short time period. This was designed this way, because a single estimation error of ball position can lead to a wrong ball velocity, and finally to no interception point. Also the ball might change direction, and there will be new interception opportunities.

The xy-coordinates of the interception point are given to the navigation controller of the robot, an earlier developed controller using modified potential fields [6]. This process is repeated every iteration. The intercept controller block diagram is depicted in Fig. 7.3.

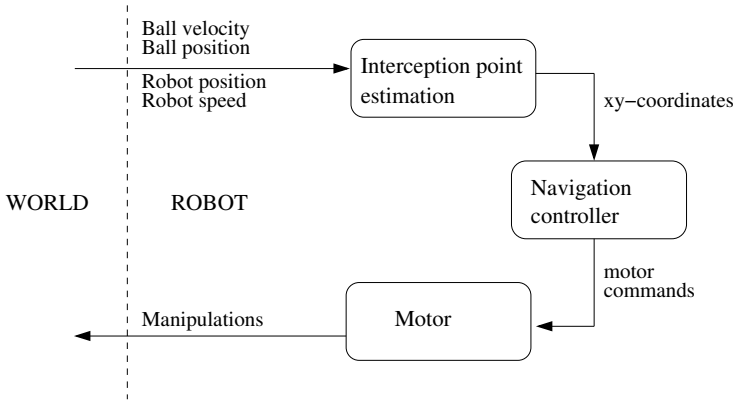


Figure 7.3: *Interception controller block diagram*

Note that, since the robots used are non-holonomic (differential-drive), the assumption that the robot moves in a straight line to the interception point, ignoring the initial robot orientation, leads to an error in the estimation of the interception point. However, the estimation is iteratively applied and becomes more accurate at each new iteration step, until when the robot faces the correct heading, where the straight line motion assumption is fully correct.

7.3.3 Ball-Velocity Estimation

Essential for the presented intercept controller is a correct estimation of the ball velocity. The world model of the agents provides the ball velocity already. Here is explained how the estimation is being done.

The micro-agent *Vision* is continuously grabbing images from the robot's camera, and processes the image for object recognition. One of those objects is the ball. After estimating the centre of the ball, a ball position is returned relative to the robot position. This way the ball position is calculated about 20 times a second. Another micro-agent, *Fusion*, puts n position samples with their time stamp into a list, and from this list the ball velocity is estimated. Three methods for the velocity estimation are available; the Normal Average-method (NA), the Weighted Average-method (WA) and the First Last-method (FL).

- **NA** The ball velocity between all position samples is calculated and the average of them is returned as the ball velocity estimation.
- **WA** This method is the same as the previous, but the velocity units calculated from each sample pair are provided with a validity weight, based on their recency.
- **FL** The ball velocity is estimated using only the first and the last position sample from the sample list. This way one big time period is used for the estimation. An error in ball position will have a smaller effect

All methods have their advantages and disadvantages. In order to determine which one would be best to use in the interception controller a small test has been done. The NA method came out as best for the interception behaviour, because although the average error of the FL method is slightly smaller, the NA method is more consequent in the output, which will lead to a more stable intercept controller.

Sensor Fusion

In order to get the best possible ball position samples, also sensor fusion is applied by the *Fusion* micro-agent, using multi-sensor Bayesian techniques [15]. Since the robot has two cameras, a *front* camera and a omni-directional *up* camera, the ball position can be determined twice within one robot. This information from both cameras is fused locally to get the local ball position estimation.

7.4 The pass heuristics

For the pass behaviour a few heuristics have been developed that are used by the robots during the decision making and execution. One is the heuristic mentioned in section 6.2.2, to help the kicker with choosing the best receive partner. A second heuristic one is used by the kicker to determine the exact coordinates for the pass target, which are not necessarily the coordinates of the receivers position. Finally there is a heuristic for the receiver, which is used in the integrated pass behaviour, to determine the receiver position. This is the position where the receiver is heading to, in the preparing state of the pass behaviour. The presented heuristics are meant as initiative for strategic team play. They have been developed for the previous explained integrated pass behaviour, but they can be improved or specified for other game situations.

Pass target determination

To determine the pass target for the kicker, the observation has been used that it is difficult for the receiver to control the ball after intercepting, when the ball is shot straight towards the receiver. When the receiver grabs the ball from the side, there is a better change for good ball control. In order to help the receiver, the kicker will determine a target position at the receiver's side. Figure 7.4 shows an example. The kicker looks if the receiver is left or right of a virtual line from the kicker to the goal centre. The pass target coordinates are defined as the receiver coordinates, placed 50 cm in y-direction towards the virtual line.

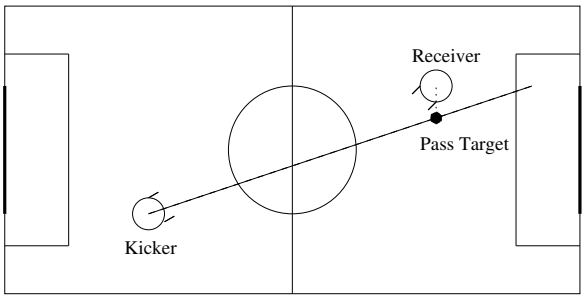


Figure 7.4: *Pass target determination.*

Pass partner selection

In the integrated version of the pass behaviour, the kicker chooses a potential receiver by himself. An easy heuristic, developed to make the kicker select one of his team mates, is only looking at the positions of the team members in the field. The kicker selects the partner who is closest to the opponent goal. It might be clear that in soccer the positions of opponent robots are also important for partner selection for a pass, since they can be located in between the kicker and the receiver. Since the tool of identifying other robots is not available yet, we do not know the opponents' positions yet, and only the own team members are taken into account.

Receive position determination

This third heuristic is also used in the integrated version of the pass behaviour. In the *prepare* state, while the kicker is preparing the pass, the receiver moves towards a certain position, that should facilitate the pass and create a better situation for continuation after the pass. This makes the pass behaviour more dynamic. Again a strategic good heuristic should take the opponent robots into account, but as stated, they can not be identified yet. The used heuristic is quite simple: two receiver positions have been defined on forehand, namely coordinates (1.2, 3) and (-1.2, 3). He selects the closest of the two, without crossing a virtual line between the kicker and goal centre.

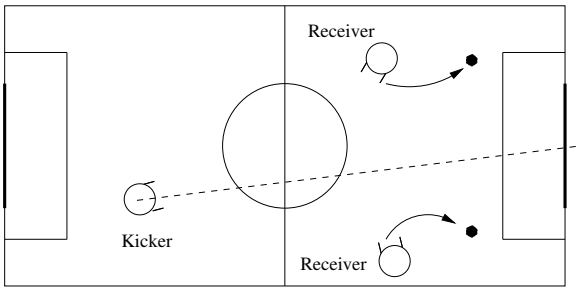


Figure 7.5: *Receiver position determination.*

Chapter 8

Experimental Results

8.1 The Joint Commitments

The framework works as expected. As the commitment has been defined well, the framework takes care of the set up, the execution and the ending and it handles the communication as well. The communication protocol variables have been defined empirically, Table 8.1. There is not one perfect value for those variables. Important is to find a balance between the broadcast intensity and the certainty about the communication.

Table 8.1: *Values of communication variables.*

Name	Value
contact.repeat	0.4 s
contact.timeout	1.5 s
repeat.max	5 times
request.timeout	1.2 s
setup.minTimeDiff	0.4 s

The time it takes from sending a message until a reaction from the receiving robot between 0.1 and 0.2 seconds. That represents the precision of timing that can be achieved by using explicit communication, it is the time it takes for one agents to react on the other.

The framework provides the possibility for easily implementing different kinds of relational behaviours, as shown with implementing two different versions of the pass behaviour and the *first-home* behaviour. However, the results of the relational behaviours strongly depend on the results of the corresponding individual behaviours of both robots. The *first-home* behaviour uses relatively simple primitive behaviours, and there the results of the use of the framework were good. The pass behaviours showed some satisfying results, but a successful realization of a complete pass from beginning until the end is only achieved exceptionally. Direct advantage of passing the ball from the back of the field to the front is time gain. Passing goes much faster than dribbling the ball, and even if the pass is not ended with complete

success, there can occur better game situations. In order to get more understanding about the quality of the pass behaviour, the results of the individual behaviours **aimAndPass** and **intercept** in a controlled test environment will be shown in the following sections.

8.2 AimAndPass

The **aimAndPass** behaviour has been implemented following the methods described in previous chapters. After some tests, the variable R, the radius of the fictional circle around the robot, has been set to 0.5 m.

The performance of the behaviour is as expected. Figure 8.1 shows the result of a test with a real robot on a MSL soccer field. The robot start position is at the middle of the field, and his target is the middle of the goal which is located diagonally behind him. The robot turns with the ball and shoots in the desired direction. Tables 8.2 and 8.3 show results of 10 trials.

These results are obtained in a controlled test environment, However, on the field, during a play, the direction does not always seem to be as accurate as desired. Reasons for this can be found in self localization errors for the kicker and the receiver. A small difference in the kicker orientation leads to a significant spatial error when the distance to the target grows. In the discussion we will examine this more closely.

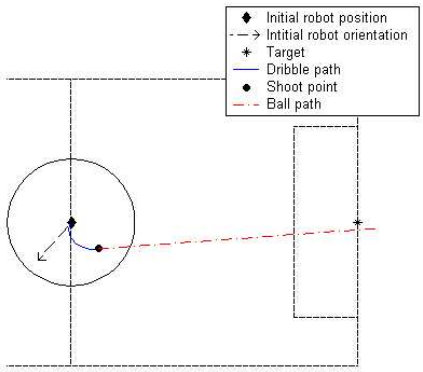


Figure 8.1: *Plotted result of the* aimAndPass *behaviour*

Table 8.2 shows results of the precision of tests with the aimAndPass behaviours. Trial number is in chronological order within the experiment.

Table 8.2: *AimAndPass results.*

Trial nr.	Angle (deg)	Distance (m)	Time (s)	Error (cm)
1	132	4.5	3.26	-17
2	138	4.5	2.59	Failure
3	133	4.5	3.30	+22
4	133	4.5	3.30	-20
5	135	4.5	3.28	-9
6	137	4.5	1.98	Failure
7	134	4.5	3.48	+3
8	134	4.5	3.20	-23
9	134	4.5	3.16	-9
10	133	4.5	3.23	+4

Table 8.3: *Average results of aimAndPass tests.*

Number of failures:	2/10	20%
Over turned:	3/8	37.5%
Under turned:	5/8	62.5%
Average time for success:	3.28 s	
Average absolute error:	13.4 cm	

8.3 Intercept

For the **intercept** behaviour, the assumed average robot speed, v_r', has been set to 0.5 m/s. An example of the interception of a moving ball by a real robot in a MSL soccer field is shown in Fig. 8.2(a). Clearly it can be seen that the robot takes the ball velocity into account, and moves directly to the interception point. In this example, the ball rolls with a speed of 1.2 m/s in a straight line. The ball path that is shown is the ball path as observed by the robot. Table 8.4 shows the results of 12 trials. The results are divided in successes and failures. A success means that the robot ended with control over the ball. A failure can occur when the ball rolls out of the field, or when the robot doesn't see the ball anymore. Furthermore the intercept trials are separated by whether the ball bumps away from the robot or not, and whether the robot misses the ball or not. These cases are also illustrated in the following pictures.

Table 8.4: *Intercept results of a controlled situation where the ball passes diagonally in front of the robot.*

	Number	Bump	Miss	Ok
Success	5	2	1	2
Failed	7	3	4	
Total	12	5	5	2

The pictures 8.2(a) until 8.2(d) show several examples of what can happen in the intercept behaviour. In the first one a successful interception is depicted. Other possibilities are that the ball bumps away, Fig. 8.2(b), or the robot misses the ball before grabbing it, Figs. 8.2(c) and 8.2(d). Fig 8.2(d) also differs from the first three examples by letting the robot start with his back towards the ball. All the examples shown here ended with success.

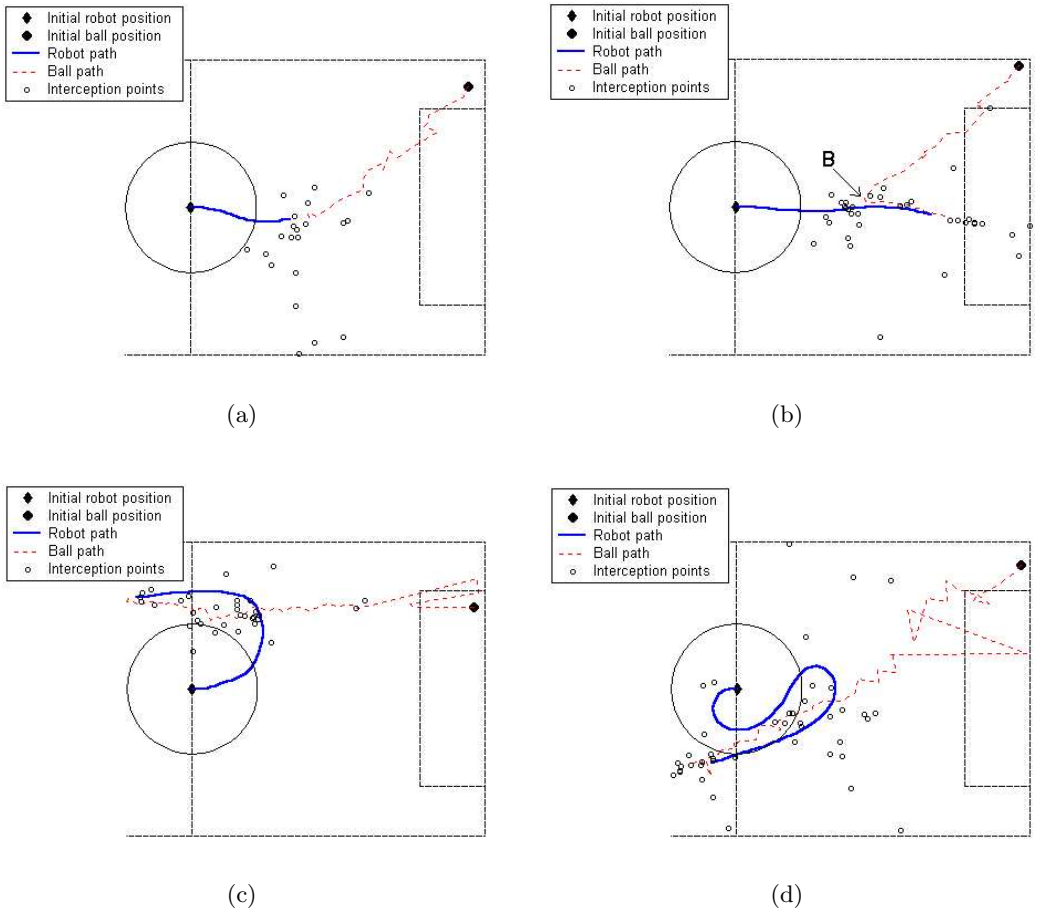
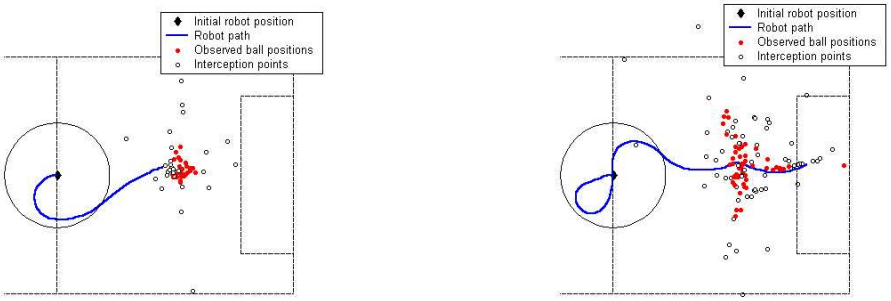


Figure 8.2: *Intercept results of moving ball: a) immediate success; b) success after bump at B; c) and d) success after the robot misses first*

Interception of a stationary ball

The intercept behaviour is developed in such way that it should work independent of the ball velocity. If the ball is not moving, the calculated interception point is the same as the ball position. Figure 8.3(a) and 8.3(b) show that the intercept behaviour also works for a ball without velocity. In the test situation the ball was positioned 2.25 meter behind the robot. Figure 8.3(a) is an example of how it should go, but 8.3(b) shows a more common occurrence; first the robot can not choose between turning right or left, and after he has chosen, he overreacts. Both pictures show interception points, not corresponding with the ball position, although the ball is not moving. This means that the robot observes ball movement. This happens especially when the robot is moving himself.



(a) (b)

Figure 8.3: *Intercept results with stationary ball (i.e., a ball that is not moving).*

The only available primitive behaviour that was used before to guide the robot to the ball was until now the *getClose2Ball* behaviour. This behaviour did not take the ball velocity into account, and guided the robot straight to the observed ball position. In Table 8.5, results of guiding the robot to a stationary ball show a better performance of the *getClose2Ball* behaviour than of the *intercept* behaviour. The ball is positioned 2.25 meter behind the robot and results are given in numbers of failures, bumps.

Table 8.5: *Experimental results of comparing the behaviours* getClose2Ball *and* intercept*, with guiding the robot towards a stationary ball.*

	GetClose2Ball	Intercept
Number of Trials	5	5
Success (Ball controlled properly)	4	1
Success (after Bump)	0	1
Failed (Ball out of sight)	1	3

Chapter 9

Discussion

9.1 The Relational behaviour Framework

9.1.1 Flexibility of the Framework

The framework provides an easy way for defining a relational behaviour, and then handles the set up, behaviour execution and end. The behaviour has to be defined by:

- commitment states
- setup conditions
- switch conditions
- related individual behaviours in each commitment state

Completely defining a relational behaviour in advance, implies that there is no variation possible during the execution and that almost everything that will happen is already decided and that because of that only a limited degree of complexity can be reached. In this section some of the properties of the framework will be discussed, and the capabilities of the framework will be examined. As we will see, there are possibilities for variation, but the designer of the commitment has the responsibility of a correct execution trajectory.

Parallel States

As stated in the problem description,this framework is meant for relational behaviours, in which both participants know what can happen and they know what the other is doing. This is true by the fact that both agents should have the same set of commitment states in the executed relational behaviour. But as can be seen within the *Firsthome* behaviour, there are opportunities for parallel states. Depending on the changing environment during the execution one of the options will be chosen. In theory, the framework allows an unlimited number of states, also parallel states. Then even the most complex relational behaviours can be created. Still, the restriction that all commitment states have to be defined on forehand holds.

Flexible Conditions

By keeping the setup conditions open, different kinds of decision making are possible. The conditions have been defined using logical predicates and propositions. The values of the predicates and propositions can be generated by a neural network or even a random answer generator.

The same thing holds for the switch conditions. Those values can be defined as well using a neural network. Note that here it is more important to have the conditions of both participants coordinated, because the current framework does not do anything if none of the participants switch to a new state. Coordinating the switch conditions is also important when one starts to work with more reactive approaches of a relational behaviour. For example if the receiver in a pass behaviour is allowed to switch to *intercept* by himself, what will happen if he does so while the ball has not been shot yet? Following the current framework, the kicker will follow to he *intercept* state, even though he should know this is not the right thing to do. The designer of a behaviour will have to take care of this situation.

Synchronization

The framework provides synchronization only at predefined moments; within a commitment state there will be no synchronization. One can think of extended relational behaviours where this is useful. For example a whole soccer game can be seen as one relational behaviour, where the agents do their individual behaviours, and only need to synchronize when a referee demands so, for example when the ball is out of the field or when it has been scored. The behaviour set of individually playing soccer can be put in one commitment state.

9.1.2 Possible Points of Improvement

The presented framework is a general framework for relational behaviours. Still there are some restrictions. As stated in the previous section, 9.1.1, the division of a relational behaviour in states should be known by both participants. This is seen as a consequence of the kind of behaviours that the framework has been designed for, and this is not seen as being a restriction. Other improvements could make the framework still more general. One of the restrictions of the framework is that it handles only two agents. Another one is that the agents can only be committed to one relational behaviour at the time. In this section we try to reason about options for expanding the framework in these two directions.

More Agents

One of the restrictions of the framework presented here is that the relational behaviours only take place between two team mates. One can think of relational behaviours involving more than two agents. Since the framework is based in the Joint Commitment Theory, which is a general theory for teamwork, the basic ideas of the framework can be the same. In the implementation all functions, like communication functions, are specifically built for behaviours with two agents. Adapting

them to handle larger teams would be an interesting future work and it would make the framework more general.

Complex Commitments

Another limitation of the current framework is that an agent can only be committed to one relational behaviour at a time, and it has to finish this behaviour completely before starting a new one. More research should be done to check the possibilities of making the framework somehow recursive. Maybe a theoretical analysis is necessary in order to reason about relational behaviours nested within other relational behaviours, but constructed at the same relational decision level.

9.2 The Pass Behaviour

Both pass behaviours showed some satisfying results, but a successful realization of a complete pass from beginning until the end is exceptional. Both individual behaviours, aimAndPass and intercept can fulfill their task with success under controlled situations, but apparently they are not yet robust enough for a real world situation. The four bottlenecks that are identified are:

- Self-localization errors
- Non-dynamic shooting mechanism
- Difficulties with ball control
- Ball velocity estimation errors

The first two concern mainly the **aimAndPass** behaviour, and the last two the ball interception. Both behaviours will be analyzed in this section, and possible improvements suggested. The pass behaviour itself shows that it has potential to get better. The presented heuristics that are used, are just an initiative for more strategic ones. For improvement in the pass partner selection heuristic and the receive position determination, a great help would be to be able to identify other robots, from the own team and from the opponent team. Furthermore, more research could be done to different versions and pass situations. The pass presented here is useful for the fact that passing the ball would go faster then dribbling the ball to the other side of the field. When a better accuracy of passing is achieved the pass can be used for example to create better score opportunities directly.

9.2.1 AimAndPass

As shown, in a controlled environment, the kicker shoots the ball well in the desired direction. However, on the field, during a play, the direction does not always seem to be as accurate as desired. Reasons for this can be found in self localization errors for the kicker and the receiver. A small difference in the kicker orientation leads to a significant spatial error when the distance to the target grows. Those errors caused by localization can be avoided if the camera is used directly to determine the target direction. If the kicker recognizes the receiver only by its own camera, the global position is not important anymore. Figure 9.1 shows a situation where

the real kicker and receiver positions are shown, and the positions where they think they are (the *phantom* positions). Using the phantom positions to determine a kick direction, the ball is shot from the real kicker position, towards the pass target. This case only involved position errors, not even orientation errors.

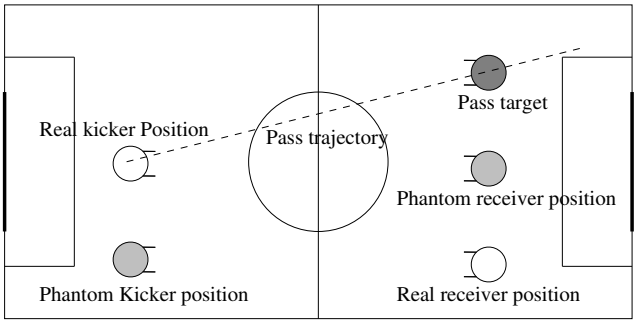


Figure 9.1: *Example of the consequences of the localization error.*

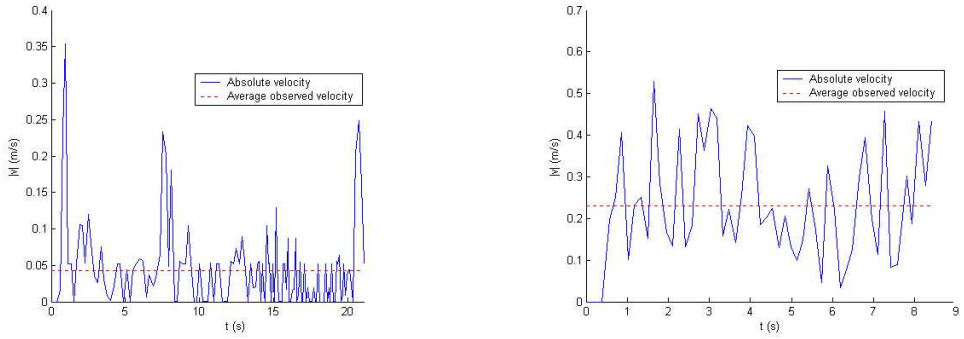
Furthermore, although kick direction might be good, the robot does not have a mechanism for controlling the ball speed. A goal shot to score, a long distance pass and a short distance pass will all be shot with the same force. The reason for this is the pneumatic mechanism of the kicker, which has no possibilities for adaptation during a game. For the ISocRob team perspectives for improvement of this issue can be found in new robots for the team, which are currently under construction. They will be equipped with an electro-magnetic shoot mechanism, which allows controlled shoot velocities.

9.2.2 Intercept

The intercept behaviour shows some satisfying results in the controlled tests, but many failures are to be found as well. Trying to find some aspects of improvement, we will discuss some issues in this section, including ball velocity and ball control. Furthermore the compared behaviour in *intercept* and *getClose2Ball* is discussed.

Ball Velocity

Essential in the *intercept* behaviour is the ball velocity estimation. As can be seen in the results section, the robot does not observe the ball trajectory as a fluent line. Since the velocity is estimated using the previous ball positions, the velocity is not constant as well. To give an impression of the error in the ball velocity estimation, the following graphs show the observed ball velocity when the ball is not moving. In the first situation the robot is not moving as well, and the observed ball velocity is plotted during a period of 21 s. In the second graph, the robot is moving with a constant speed in a straight line, along the ball. The velocity is plotted during a period of 8.4 s.



(a) (b)

Figure 9.2: *Observed ball velocity: a) robot is not moving; b) robot is moving with a constant velocity.*

Ball Control

The robots do not have the same ability of controlling a ball at the interception point as human beings have. Currently the ball control is achieved with a lucky arrival of the ball between the small springs in front of the receiver, which are designed for the controlling of the ball, but they can not be controlled actively. It is tried to make the receiver reach the ball at the same speed as the ball is moving in the direction of the robots velocity. But no negative speed for the robot is used. When the ball is moving towards the robot, it is likely to bump. When the ball is moving in the same direction as the robot, the ball control is better.

GetClose2Ball Comparison

In the results section, a comparison is made between the first getClose2Ball behaviour, which directs the robot to the ball without taking ball velocity into account, and the intercept behaviour, that does. Although the intercept behaviour theoretically should guide the robot without problem towards a not moving ball, the getClose2Ball behaviour behaves much better in this situation. Reasons for this difference can be found in the errors in ball velocity estimation. In the pass behaviour, the receiver can assume that the ball is moving, and use the intercept behaviour. But if no pass is being executed, it would be great if the robot could use the getClose2Ball behaviour in the case the ball is not moving and intercept when the ball has a velocity. Problem that occurs is: how to determine whether the ball is moving or not? The same ball velocity estimation can not be used, because of the error it has. Then if the velocity estimation error could be reduced, the intercept behaviour would act better with no ball velocity. So here we can reason in a circle about when to use which behaviour. Ideal would be a perfect ball velocity estimation. Then the intercept behaviour could handle both situations. In the current situation, the intercept behaviour is only used in a pass commitment, and otherwise the getClose2Ball behaviour guides the robot to the ball.

Chapter 10

What have we learned from this research?

In this report the general formulation of relational behaviours among real robots, based on the Joint Commitment Theory, has been introduced through an illustrative example concerning a pass behaviour between RoboCup MSL robots. The formulation uses individual decision making and behaviour synchronization among intervening robots and has been tested successfully during laboratory games without an opponent. Results of the implementation of the key individual behaviours (**aimAndPass**, **intercept**) in real robots were presented.

With this research we have tried to obtain a deeper understanding of the realization of relational behaviours in embodied agents. The presented framework uses explicit communication and explicit agreements. Exchanging information about the progress of a relational behaviour is shown to be a good way to achieve synchronized actions, especially when the agents' own observations are not sufficient enough. The framework provides a way to create cooperation in a goal-directed approach, which implies that participants have a common representation of the joint goal. This is not the case in many behaviour based approaches for cooperation. At the same time the framework allows reactive coordination as well as directed coordination for the relational behaviour execution, depending on how the relational behaviour has been defined. The framework provides clear rules for the implementation of a relational behaviour, which facilitates developing new or other cooperative behaviours.

Being able to execute planned cooperative behaviours, this research takes a new step towards strategic team play in MSL robotic soccer. The same way human beings can reason about cooperative behaviours in order to achieve future goals, strategies in robotic soccer can be defined.

Restrictions of the presented framework are mainly that it allows only cooperation between two team mates and it allows only one commitment at the same time. Concerning the quality of the pass behaviour in RoboCup MSL soccer, we can say that although coordinated teamwork has been achieved, the pass behaviour we have developed still shows some bottlenecks. The results show that observation errors and ball control problems make intercepting a moving ball difficult for robots.

Future work will concern the development of new relational behaviours under

this framework, as well as the refining of the individual behaviours, particularly by using team mates visual recognition to eliminate the self-localization error and the required communication during the **aimAndPass** behaviour, and by providing the robots with force-controlled kicking ability, so as to enable ball interception by the receiver robot, by reducing the ball speed for pass kicks, as compared to goal kicks.

Bibliography

[1] N. de Castro, R. Matias, and M. Isabel Ribeiro. Target tracking using fuzzy control. In *Proceedings of the Scientific Meeting of 3rd. Portuguese Robotics National Festival*, 2003.

[2] P. R. Cohen and H. J. Levesque. Teamwork. *Nous*, 35:487–512, 1991.

[3] A. Collinot, P. Carle, and K. Zeghal. Cassiopeia: a method for designing computational organizations. In *Proceedings of the First International Workshop on Decentralized Intelligent Multi-Agent Systems*, pages 124–131, 1995.

[4] MSL Technical Committee. Middle sized robot league, rules and regulations for 2004. http://www.er.ams.eng.osaka-u.ac.jp/rc2004msl/msl-rules-2004.pdf, 2004.

[5] P. Corke. *Visual Control of Robots: High-performance Visual Servoing*. Research Studies Press LTD, 1996.

[6] B. Damas, L. Custódio, and P. Lima. A modified potential fields method for robot navigation applied to dribbling in robotic soccer. In *RoboCup 2002: Robot Soccer World Cup VI*, pages 65–77. Springer-Verlag, 2003.

[7] A. Drogoul and A. Collinot. Applying an agent-oriented methodology to the design of artificial organizations: A case study in robotic soccer. *Autonomous Agents and Multi-Agent Systems*, 1:113–129, 1998.

[8] P. Gärdenfors. Cooperation and the evolution of symbolic communication. In *Proceedings of Conference on the Evolution of Communication*, 2001.

[9] S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on vision servo control. *IEEE Transactions on Robotics and Automation*, 13(5):651–670, 1996.

[10] G. A. Kaminka, Y. Emaliach, I. Frenkel, R. Glick, M. Kalech, and T. Shpigelman. Towards a comprehensive framework for relational behaviours. In *Proceedings of 8th Conference on Intelligent Autonomous Systems (IAS-8)*, 2004.

[11] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of IJCAI-95 Workshop on Entertainment and AI/Alife*, pages 19–24, 1995.

[12] P. Lima, L. Custódio, and et al. Isocrob 2003: Team description paper. In *RoboCup 2003: Robot Soccer World Cup VII*. Springer-Verlag, 2003.

[13] H. Matsubara, I. Noda, and K. Hiraki. Learning of cooperative actions in multi-agent systems: a case study of pass in soccer. In *Adaptation, Coevolution and Learning in Multi-agent Systems: Papers from the AAAI-96 Spring Symposium*, pages 63–67, 1996.

[14] E. Pagello, A. d'Angelo, F. Montsello, F. Garelli, and C. Ferrari. Cooperative behaviors in multi-robot systems through implicit communication. *Robotics and Autonomous Systems*, 29(1):65–77, 1999.

[15] P. Pinheiro and P. Lima. Bayesian sensor fusing for cooperative object localization and world modeling. In *Proceedings of 8th Conference on Intelligent Autonomous Systems (IAS-8)*, 2004.

[16] V. Pires, M. Arroz, and L. Custódio. Logic based hybrid decision system for a multi-robot team. In *Proceedings of 8th Conference on Intelligent Autonomous Systems (IAS-8)*, 2004.

[17] P. E. Stander. Cooperative hunting in lions: the role of the individual. *Behavioral Ecology and Sociobiology*, 29:445–454, 1992.

[18] M Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[19] B. van der Vecht and P. Lima. Formulation and implementation of relational behaviours for multi-robot cooperative systems. In *Proceedings of 8th Robocup International Symposium*. Springer-Verlag, 2004. To appear.

[20] K. Yokota, K. Ozaki, N. Watanabe, D. Matsumoto, A.and Koyama, T. Ishikawa, K. Kawabata, H. Kaetsu, and H. Asama. Uttori united: Cooperative team play based on communication. In *RoboCup-98: Robot Soccer World Cup II*, pages 479–484. Springer-Verlag, 1999.