Strategy for a robot soccer team

Roelof Heddema

Augustus 2007

Strategy for a robot soccer team

- Let's play robot soccer -

Master's Thesis by:
R. R. Heddema

Graduation committee:
Dr. M. Poel
Dr. A.L. Schoute
Ir. T. Verschoor
Ir. I.H.C. Wassink
Ir. H. van Welbergen

Faculty of EEMCS
University of Twente
The Netherlands

13 August 2007

Abstract

Vital for the success of the robot soccer system is a streamlined strategy. Robot soccer strategy is a form of decision making for robots. On the soccer pitch, robots, divided in two teams, interact with each other and a ball, each trying to win a soccer match. In a streamlined strategy, robots in the same team try to maximize the scoring opportunities for their team and minimize the scoring opportunities of the opponent by means of strategic plans.

Already for robot soccer, strategy research has been directed towards the use of weighted linear functions. However, for creating and adapting strategies, the use of a weighted linear function seems to be complex.

This thesis presents a different direction in strategy design. This uses finite state machines (FSM) to create directed positioning for the robots. The transitions of the state machines consist of guarded plans, where the guards are based on existing observations. Observations represent expert knowledge about situation on the robot soccer pitch and plans represent expert knowledge about the strategic means for the robots. In the FSM approach, strategies are represented explicitly by assigning roles to robots and modeling the desired behavior for each role with an FSM. Two of such strategies are given, namely the “2-2 formation” and “1-3 formation with rotational tactic”.

The two strategies have been tested in robot soccer tournaments. On these tournaments, universities have the means of testing their robot soccer system against each other. The “1-3 formation with rotational tactic” proved most successful. This strategy was able to recover from failed shoot attempts and used multiple coordinated robots to perform scoring attempts. Nevertheless, the “2-2 formation” seemed to have a more robust defense. A future direction to a “2-2 formation with rotational tactic” is suggested, which combines the advantages of both strategies.

Preface

To conclude the study Computer Science at the University of Twente I choose to participate in the robot soccer project. In this project, different aspects of the study are combined. I joined this project because it brings theoretical knowledge, learned in my study, into practice.

With the robot soccer team, we attended at international tournaments. These tournaments were most enjoyable. During the attendance to the different tournaments, we showed real progress and this resulted in a 2nd place at the European Championship in 2007.

Finally, I would like to thank some people. First, I would like to thank my supervisors during this project; Dr. M. Poel, Dr. A. L. Schoute, Ir T. Verschoor, Ir. I.H.C. Wassink and Ir. H. van Welbergen. They helped to bring structure to the many incomprehensible expressions of my ideas. In addition, I want to thank my project members; Joost van der Linden, Maarten Buth and Paul de Groot. The teamwork with these project members made a lot of demonstrations and matches possible. I want to thank the rest of the students in room 2070 for the great lunches and discussions. Furthermore, I want to thank my parents Rein Heddema and Piety Heddema-Calsbeek. Without them, it is very doubtful that I ever attended university at all.
I want also to thank my brothers and sisters, because they stayed interested during my never-ending study. Last, but not least, I want to thank my lovely girlfriend Akke Kelder, for the support during the whole process of my final assignment.

Enschede, 13 August 2007
Roelof Heddema

Contents

1 Introduction

Robot soccer is a very attractive game that gives researchers all over the world the means to demonstrate the current state of new technology. It provides a test bed for new technology in image processing, robot hardware, motion control and artificial intelligence. This chapter first discusses a set-up for a robot soccer game. Second, it discusses the system with which the MI20 team plays robot soccer. In addition, existing problems are mentioned. Third, the problem definition is given and this chapter ends with the outline of this thesis.

1.1 The game

The MI20 team [1] participates in a FIRA league [2]. MI20 plays its matches in the MiRoSot league [3]. In this league, robots have maximum dimensions of 7.5 x 7.5 x 7.5 cm, and use a two-wheeled differential drive. At the University of Twente Dr. M. Poel and Dr. A.L. Schoute lead the project.

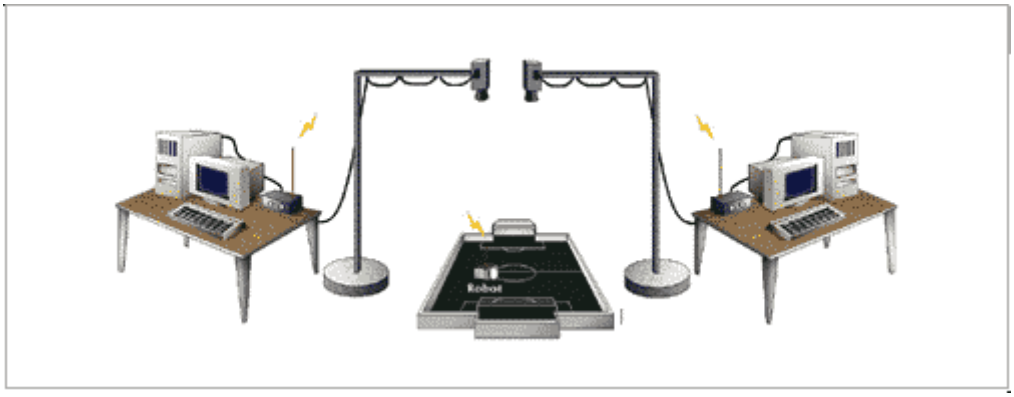

Figure 1 Set-up of the MiRoSot game

Figure 1 shows the typical set up of a MiRoSot game. In a game, two teams play against each other. Each team has its own camera, transmitter, robots and computer. The camera observes the field and the transmitter sends radio signals to the robots.

1.2 The current system

The first version of the MI20 system was developed in 2002 [10]. Maarten Buth [8], Paul de Groot [9] and I, have developed the currently used system, also known as the second version.

This section will first show the architecture of the current system. Next, it discusses the previous developed Strategy modules in the MI20 project and its relevant experienced problems regarding Strategy.

1.2.1 Software modules

To play robot soccer our system has delegated different tasks among software modules. These software modules often represent the different research. This section discusses

the different software modules; Vision, State Estimator, Strategy, Motion and RFcomm. Figure 2 shows the system architecture.

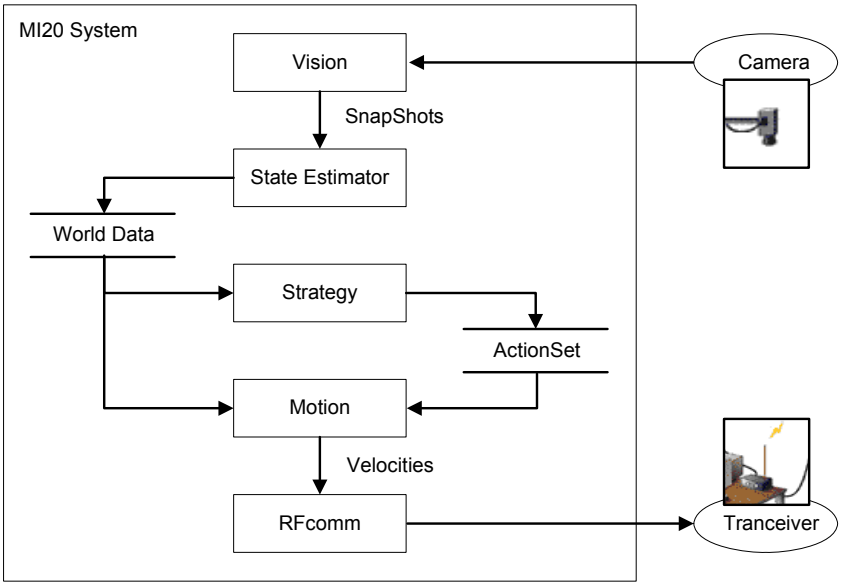

Figure 2 System architecture MI20 v2.0

Vision

The Vision module is responsible for retrieving relevant information from the images send by the camera. A data type called Snapshot combines the positional information of all robots and the ball. For the team robots, the Snapshot contains also their orientation.

State estimator

The State Estimator its responsibility is, to approximate the real world situation as good as possible. Filtering techniques and prediction correct the noisy delayed vision information. It also determines derivates like linear and angular speed from the Snapshot. The estimated information is being stored in a data structure called World Data.

Strategy

This module creates strategic Actions for the robots. The strategy depends upon the information given by the State Estimator. The strategy module has to retrieve meaningful information from so-called World Data and creates Actions for robots to execute. It is necessary that the Motion module can adequately execute Actions. These Actions are stored in a data structure ActionSet. Yet it is not the responsibility for the strategy module to create smart team play; Human operators and strategy module designers should realize smart team play.

Motion

The Motion module is important with regards of the ability to utilize the robot effectively. It depends upon the functioning of the robot on the field and the task it has been assigned. Some important tasks are moving as fast as possible to a certain point, following as accurate as possible a moving target and intercepting a ball. The Motion module also depends upon the information created by the State Estimator and it creates Velocities for the robots from the given Actions stored in the ActionSet.

Rfcomm (Radio Frequency Communications)
The RFcomm module utilizes the different communication devices of the robots. RFcomm sends Velocities, translated into adequate radiosignals, and sends it to each robot by using the transceivers.

This thesis will describe work related to the strategy module. In the MI20 project, research has already been done for strategy module.

1.2.2 Previous work on Strategy

Seesink [1] developed the the Strategy module in the first version of the MI20 system. Using feature extraction, the module transformed the input space into more suitable strategic information. Adaptations to this strategy module were being made by Poelman [6] and van Amstel [7]. Petit [5] investigated a second approach for the Strategy. Petit created a decision system based upon potential fields.

Next the so-called Single Neuron Approach developed by Seesink [1], which has also been implemented and used in tournaments, and the Potential Field Approach by Petit [5] will be discussed.

Single Neuron Approach
The Single Neuron Approach uses a weighted linear function, like a single neuron, to create output from a given input. The idea of the single Neuron Approach is first to transform the state vectors in World Data[1] of different objects on the robot soccer field into features [1]. Each feature has a semantic meaning (expert knowledge) about a certain aspect of the game. For example a feature could indicate to which extend a team player has a scoring opportunity. In Seesink's [1] design these features are not based upon game statistics, but only upon the locations of the different robots and the ball. Second, the method combines the values of the features with reward and penalty weights and these weights can differ for the different roles. For each logical plan [1] the value of this combination is a measurement for the desirability for this plan. Third, the method chooses the most desired plan. A logical plan can be queued after or replace the current plan which has to be executed by the motion part of the system.

Potential Field Approach
The potential field approach proposal by C. Petit [5] distinguishes two parts in Strategy. One part chooses an interception strategy that is case-based and selects one robot as an on-ball player [5]. The other part assigns plans for the other robots in a supporting strategy. Potential fields are the basis for the design. The approach creates one potential field for all robots. The method retrieves all strategic positions from this potential field. First, a desirability value is computed for each position in the field, this leads to a potential field. The desirability value is based upon an additive combination of sub-functions. Such a sub function creates desirability values for a certain plan at each position on the field. The method creates a combined field from the desirability fields. Second, the method selects the top positions from the combined potential field. Third, the method allocates these positions among the robots.

The advantage of the Single Neuron approach is that it facilitates in easy adding expert knowledge to the system by means of features. This approach also has some disadvantages. The next section will discuss these disadvantages. The main advantage of the potential field approach is that it always appoints an on-ball player. The main disadvantage that it has never been implemented for the system and that its concept has not been proven being able to work.

1.2.3 Problems current system

First with regard of strategy, general problems, which previous project members observed, are considered. These problems have to be considered with regard of the design for the new strategy module.

- **Limited computational time** [9]
  Due to the real-time nature of the system, which has to process camera images with approximately 33 fps, strategy has limited time to make strategic decisions. For example, this means that for strategy it is not practical to search for all possible states in the robot soccer game to calculate the 'perfect' decision.
- **Strategy has no recovery techniques**
  Stuck robot detection is one part of resolving deadlock situations in the robot soccer game [1]. For example not recovering robots with regard of failure with shooting attempts [12], [9], [13] exists.

Second, with regard of the implemented strategy of Seesink[1] some problems are considered. These problems are discussed because this one has been implemented and used by several people. According to Petit [5], Maatman and Poelman [6] and van Amstel [14] there is a great disadvantage for this design:

- **Strategy design is complicated.**
  The way to adapt strategy is nontransparent [5], therefore it is difficult to change team play [6]. Coordinated team play, with the use of resource claiming [1], is realized at a laborious way [6]. Furthermore, the evaluation function that is realized with a combination of different matrixes is difficult to adjust to let the team play well [14].

Third, problems in the current version for the Strategy with regard of input and output for this module are mentioned.

- **Unreliable World Data**
  The State Estimators robot-tracking algorithm often fails during game play. As result, the State Estimator sends faulty World Data to the Strategy module. If tracking fails, the Strategy does not know where which robot is.
- **Inadequate transformation of Actions to Velocities**
  Frequently changing the Actions results in slow control speeds being sent to the robots by the Motion module. This makes the execution of the Actions inadequate. Especially a moving ball causes the change of the attributes for these Actions.
- **No collision avoidance for practical use in the Motion Module**
  The Motion module does not provide in collision avoidance. This also makes the execution of actions inadequate.

The new design for the Strategy Module shall consider these problems.

1.3 Problem definition

Since 2002, MI20, the robot soccer team of the University Twente has participated in the MiroSot league. It was difficult to give demonstrations and to play games after a number of year's development. This is caused by several problems. Among other things the strategic planned behavior of the robots was difficult matched with the developed Strategy and difficult to explain by the unclear communication with other modules.

The most important task includes designing a new Strategy, which can be used for playing games and giving demonstrations. An important requirement of the Strategy is that strategic planning takes place for the robots. Strategic observations and strategic positions based upon expert knowledge will be used. Second, it must be easy to translate strategy to behavior and the other way around. This can be reached by changing the set-up of the Strategy and to simplify the decision process.

For this task, the next research questions are found:

RQ1: How can the Strategy module be designed in such way that it creates strategic Actions for robots to execute?

RQ2: How can the strategy be designed that it is easily recognized in the observed behavior?

Another task is to ensure that the requirements, which are necessary for the adequate execution of the Actions of a Strategy for MI20 the system, are met. The requirements for a good strategy are reliable information and adequate supervision of the robots. A good Strategy alone cannot ensure team play. For that there is looked first at the following components of the system of which the methods will be improved; Robot Identification, Collision Avoidance, Motion.

For this task, the next questions to be answered are found:

Q1: Which method can be used to provide strategy with consistent information?
- **Q1.1**: How can robot identification be designed in such way that identification can be performed at each moment during a robot soccer game, without the need for a tracking method?

Q2: How can adequate supervision of the robots be achieved?
- **Q2.1:** Which techniques can be used to keep the robots pursuing the team goals while keeping the robots free from deadlock situations?
- **Q2.2:** Which techniques can be used to move the robots in such a way that they are able to respond adequately to frequently changing actions?

1.4 Requirements for strategy

Different requirements are distinguished. One is with regard to the design of the Strategy Module in which a strategy for team play can be defined. A set of metrics is chosen to measure the improvement of the new system and the new approach for the strategy should perform equal or better for these metrics than the previous methods. Pressman [11] writes more about the use of metrics to measure model quality. The selected metrics are partly inspired by Pressman and partly specific for the robot soccer application. The other requirement encapsulates the behavioral team goals for a strategy for team play.

Metrics
This section considers the metrics for the Strategy module. With this metrics, the different approaches for the Strategy module are compared. These metrics are focused at the problems mentioned in 1.2.3. The Strategy module has to create strategic Actions and it should facilitate the support of a strategy definition that is easily recognized in the observed behavior of the robots.

M1: Support of time based behavior.
To enable stuck robot recovery and provide solutions for failure in shooting attempts. Time based behavior is characterized by behavior that is parameterized by time.

M2: Computational complexity
To keep the CPU load of the system in mind, the computational complexity for the strategy module should be kept low.

M3: Understandability.
The more information programmers can comprehend about the strategy module, the smarter a strategy for robot soccer play can be designed. The module should be well understood and dependencies between internal, external and shared components should be well understood. Team play should not be realized in a time-consuming way.

M4: Reductive explainability.
Important aspect of this metric is that, what one can see in the team play is the same as what has been defined for team play. System states and important variables should be visible during execution. With good reductive explainability, incorrect output can be identified. It should not be difficult to see how the combination of different techniques leads to team play.

M5: Adaptability.
This metric is mainly driven by the effort one has to take to locate and fix an error in the system, or to change something to strategy which is needed as response to some specific weakness against opponents.

M6: Extensibility.
The degree in which the architectural data and procedural design of the strategy module can be extended. Furthermore, the degree in which one understands how they can extend the strategy module.

The listed metrics will be used to evaluate the Strategy module.

Team goals
In 1.2.1 it is already mentioned, that defining smart team play is the task of the designer of the Strategy module. Two important team goals focus the team play of the robots on winning the robot soccer game.

TG1: Scoring

TG2: Prevent the opponent from scoring

The listed team goals are guidelines for team play strategy development, in other words; guidelines for the strategy developers.

1.5 Outline

The purpose of this thesis is to describe ways to improve the Strategy module and strategic team play for the robots of the MI20 team. Chapter 2 describes the design of the Strategy module. The Strategy module facilitates in the selection of different strategies that can be used for game play and two different strategies are described in chapter 3. Chapter 4 describes the changes, which are necessary to supply the Strategy module with consistent information and the adequate execution of the Actions. Chapters 5 will state conclusions and recommendations.

2 A FSM approach to strategy design

The Strategy Module is responsible for assigning an Action to each robot. These Actions are stored in the ActionSet. Strategy is provided with World Data to make adequate decisions. This chapter will discusses the possibility of if Action generating with a state machine (FSM) driven approach.

The responsibility for the Strategy Module is to assign an Action to each robot. Then strategy uses distributed Player agents to get a PlanSet. In Strategy, the Action Creator converts each Plan in the PlanSet into an adequate Action to store in the ActionSet. In Figure 3 the architecture for the Strategy Module is shown.

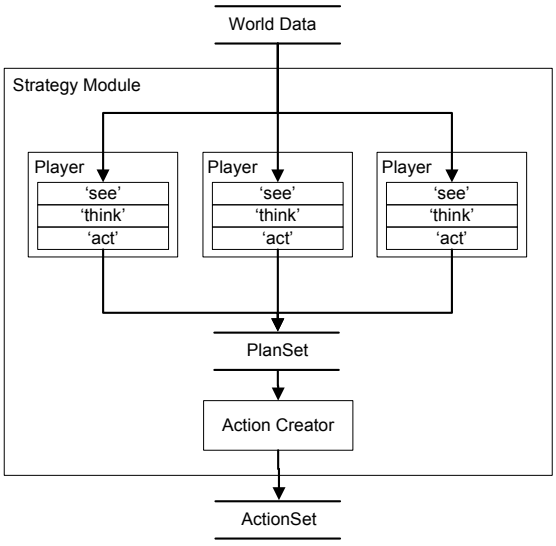

Figure 3 Architecture for the Strategy Module

The Player is created according to the (multi) agent paradigm. Therefore it can 'see', 'think' and 'act':
- 'See' : It sees strategic information (explained later) in the provided WorldData.
- 'Think' : It thinks with the use of a state machine (FSM).
- 'Act' : It acts with use of strategic Plans (explained later).

The first section of this chapter discusses the Strategy, Players and the interaction between the Strategy and the Players. Furthermore, the first section discusses the use of the FSM to create the rationale for the Player. The second section explains how one can develop a FSM that creates a Role based player. The third section will give an example how one can develop a team strategy with use of Players with different FSM's. The metrics used to evaluate the strategy approach are support of time based behavior, computational complexity, understandability, reductive explainability, adaptability and extensibility (also listed in § 1.4). The evaluation elaborates also upon coordination.

2.1 Introduction to FSM and Strategy

In this section the function of the Strategy module shall be explained first. Second, the contribution of FSM in Strategy shall be elaborated.

For a better understanding, the relevant modules and data or agent structures are discussed before the ordering of their messaging to each other is given.

World Data
World Data is a data structure which contains information, such as position, orientation and velocities, about robots and the ball.

Strategy
The Strategy Module is responsible for assigning an Action to each robot. These Actions has to be stored in the ActionSet. Strategy is provided with World Data to make adequate decisions.

Player
A Player represents a soccer player with a robotic body. This representation is created according to the (multi) agent paradigm. It can 'see' through Observations, it contains a state machine for 'thinking' and it 'acts' by means of Plans.

Observation
An Observation represents a state variable about the robot soccer game with a strategic meaning. Observations are named like "BALL_IN_DEFENSE" and can represent expert knowledge about (robotic) soccer.

Action
An action is a higher-order strategy decision, such as moving to a particular position or trying to score a goal. An Action has to be translated into adequate control signals by the Motion Module.

ActionSet
The ActionSet is shared by the Motion Module and the Strategy Module. Motion accesses the ActionSet to determine which functions have to be used for creating control signals and Strategy accesses the ActionSet to makes its decisions recognizable for Motion.

Plan
A Plan represents the means of the Player in which it can act 'strategically'. Plans can also represent expert knowledge about (robotic) soccer and therefore the plans are named like "KICK" or "BACK_UP_ATTACKER"

PlanSet
The PlanSet contains the different Plans for each Player. Strategy accesses the PlanSet to store the Plans retrieved from the Players and when Actions are created.

The process of assigning to each robot an Action is shown in Figure 4. The Strategy receives World Data from the StateEstimator. The Strategy has a number of Players and in Figure 4, there are two Players, named Goalie and Forward. The Strategy passes its World Data on to all Players. Then Strategy asks each Player to calculate a Plan. This Plan is stored in a PlanSet. When a Player is asked to calculate a Plan it takes 3 steps to do so. First, the Player determines the different

Observations by analyzing World Data. Second, the Observations are used as input for his own finite state machine (FSM), and the FSM produces a Plan. Third, for the selected Plan, attributes are determined. When a Plan has been calculated for each Player, the Strategy converts each Plan into a suitable Action. The last step of Strategy involves the storage of the Actions in the ActionSet.

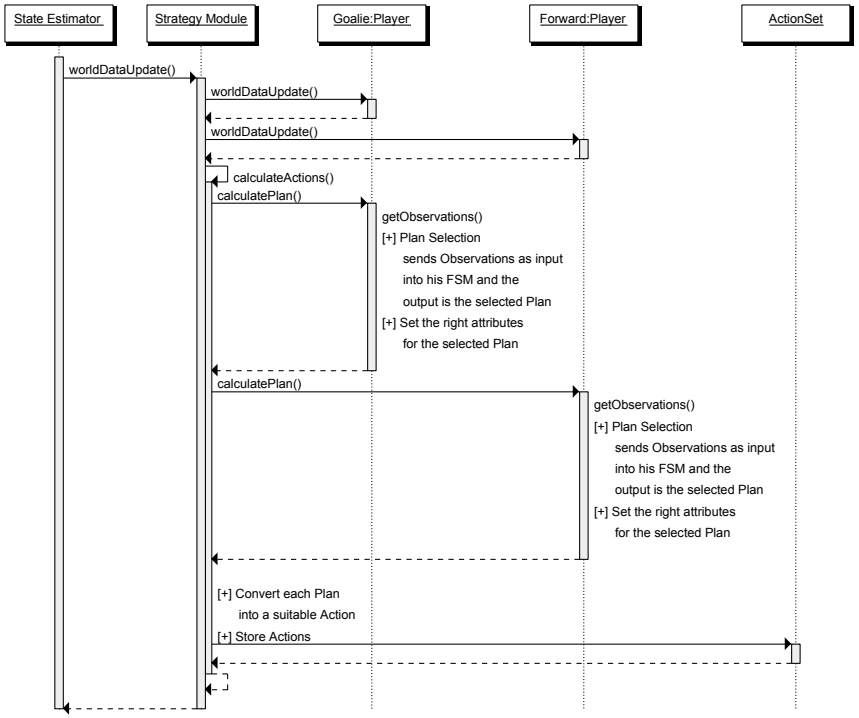

Figure 4 Sequence diagram for Strategy

The new approach for the Strategy module contains state machines for roles such as goalkeeper. Other options are kept open for further research because simplicity is considered important. The Player represents our soccer player. Each Player has its own FSM for plan generation. Before plan generation by means of a FSM is explained, different core terms are discussed.

StateMachine

A StateMachine is a set of Transitions and a set of States. It represents the behavior of the Player and it is named by its Role, Formation and Tactic.

State

A State represents a memory element for the Player. This enables different output with the same positional input.

Transition

A Transition represents a state change. A Transition contains an exit and entry State, which represent which State the Player is currently in and which State the Player will be in when the Transition is selected, respectively. It has a Plan type as output. A Transition is enabled when its ConditionExpression is ***true*** and the State of the Player is the exit State from the Transition.

Condition
A Condition represents whether an Observation is below ('<') or equal higher ('>=') some predefined value. A condition is ***true*** or ***false***. A condition is represented like '(BALL_IN_DEFENSE>=0.5)'

ConditionExpression
A ConditionExpression is an expression with Conditions. A ConditionExpression is the conjunction of multiple conditions. A ConditionExpression can also represent expert knowledge or being used to hide all conditions in an abstracted view of the state machine. A ConditionExpression can be named like "Ball is Left Forward" to express the expert knowledge. An example ConditionExpression is (BALL_X>0.5)^(BALL_Y>0.5).

Preferability
The Preferability of a Transition is used to determine which Transition is selected when multiple transitions are enabled; the one with the highest Preferability is selected.

Role
A Role represents expert knowledge about the intention of the single Player to dedicate itself to attack and/or defense, typical role names are "Attacker" and "Defender".

Formation
A Formation represents expert knowledge about the positioning of the robotic body's and Roles of the Players in the back or forward. Formations are named like '"1-3", which means 1 defender and 3 attackers.

Tactic
A Tactic represents expert knowledge about how the positioning and shooting is coordinated for the different Players. Tactics are named like "Using Wings" and "Rotational"

In Figure 5, the state machine for a Player is shown. The filled circle points to the initial state. The hollow circle containing a smaller filled circle represents a final state. A rounded rectangle represents an intermediate State. The arrow denotes a Transition and the ConditionExpression and Plan are divided with a "/". The Player calculates from the World Data the Observations. Then the Transitions in the Players StateMachine are being evaluated. Each Transition has a Preferability value, and when multiple Transitions are allowed to happen, the Transition with the highest Preferability is executed; a Plan is determined and the State of the Player is being updated. When Preferability is not relevant, it is disregarded in the state diagram.

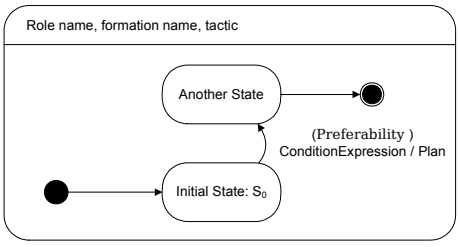

Figure 5 FSM of the Player

The next section discusses two FSM's for a single role.

2.2 A FSM for a single role

This section describes a FSM, which is characterized by a certain role. Roles are meant to assign a specialized function to a certain Player. These specialized functions are related to the overall team goals:

TG1: Score

TG2: Prevent the opponent from scoring

When a player is dedicated to TG1, the role is often Attacker or Forward and when a player is dedicated to TG2, the role is often Keeper, Defender or Back. Another way to organize coordinated roles is by assuming that some defender operates in some area of the field. This results in roles like “Left Back” or “Right Forward”. Next, a Player that is dedicated to scoring, an Attacker, will be shown.

2.2.1 A FSM for a role dedicated to scoring

Figure 6 shows the state diagram of an Attacker. This section will give a conceptual overview of the different states and transitions to reach certain behavior for the robot. After that, more detailed Conditions and Plans of the informal descriptions are shown.

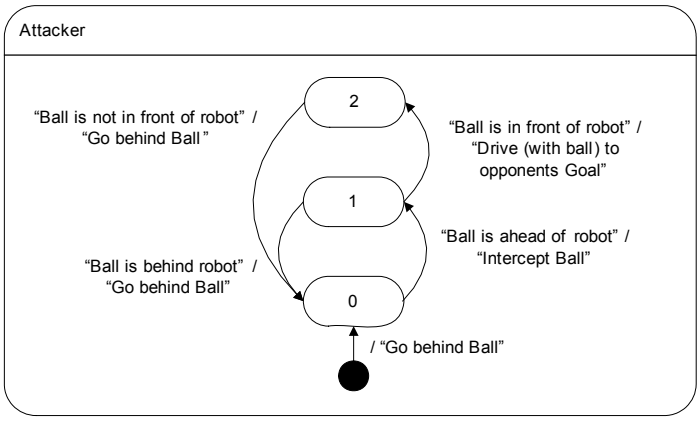

Figure 6 State diagram for an Attacker

The attacker has several states, first the ideal trail through the states is discusses. It starts in state 0 and it first objective is to get behind the ball. As soon as the robot is behind the ball, the ball is in front of the robot, it succeeds to state 1. The robot is one step closer to scoring a goal, and starts to intercept the ball. As soon as the robot has arrived at the ball and it has the ball directly in front of him it succeeds to state 2. In State 2 the robot can start driving towards the opponent’s goal to put the ball in the goal of the opponent. A goal for the MI20 team!

Of course, things can go wrong during execution, when the robot is in state 2 and it starts driving toward the opponents goal it can loose the ball. If that happens the scoring attempt is considered having failed and the player falls back towards state 0. When the robot is in state 1, something similar can happen: the ball gets behind the robot again so the player falls back to state 0 and it has to get behind the ball again.

Table 1 shows the descriptions with their detailed Plans and ConditionExpressions.

	Description	***Detailed***
ConditionExpressions	Ball is in front of robot	`( BALL_IN_FRONT > 0.5 )`
	Ball is not in front of robot	`( BALL_IN_FRONT <= 0.5 )`
	Ball is ahead of robot	`( BALL_IS_FORWARD > 0.5 )`
	Ball is behind robot	`( BALL_IS_BACKWARD > 0.5 )`
Plans	Drive (with ball) to opponents Goal	`( REACTIVE_MOVE { x=1.0 , y=0.5 } )`
	Intercept Ball	`INTERCEPT`
	Go behind Ball	`GO_BEHIND_BALL`

Table 1 Description with their details

2.2.2 A FSM for a role dedicated to prevent scoring

In Figure 7, the state diagram of the keeper is shown. The keeper has a defensive stance, translated into state S_{active} and a passive state $S_{passive}$. When the ball is in the offense, the keeper stays back and is maximizing goal coverage. The keeper comes in the defensive stance when the ball is in the defense. When defending, the keeper has for solutions for 3 situations. The first situation is when the ball is in the clear area, the keeper will clear the ball. A clear area is defined as a region next to the goal, in this example this clear region does not depend on the position of other robots. Clearing the ball in (robot) soccer is an attempt to shoot the ball away from the goal. Sometimes, it happens that the ball is stuck between two robots or between the wall and a robot. The second situation, when the ball is stuck, the robot is commanded to spin, this command often resolves the stuck ball situation. Third, when the ball is not cleared or spinned away, the keeper will defend the goal line by staying at a line before the goal line and staying between the goal line and the ball, this action is called Block Ball.

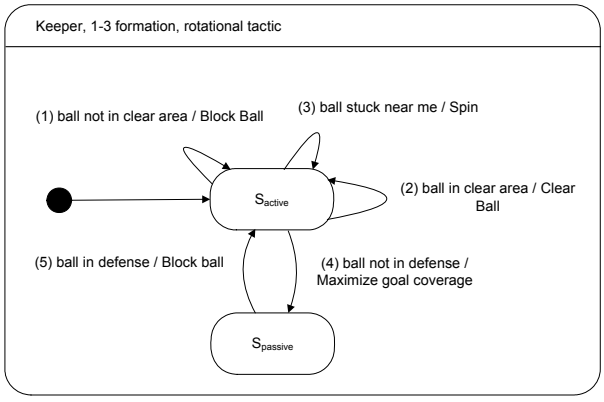

Figure 7 State diagram of the Keeper

2.3 A FSM based strategy for a team

The previous section illustrated two different roles; each of them was dedicated to one of the two team-goals. A combination of the two players already provides in a team existing of two players. Nevertheless, the MI20 robot team consists of five robots. This section will describe the principle of formation and tactic in which coordinated play can be categorized. At conceptual level a strategy for a 1-3 formation is given.

2.3.1 Relationship of strategy with formation and tactic

This section gives more attention to what formations are and how tactics relate to strategies. The use of formations provides in some decomposition of the playfield into areas in which a limited number of players may stay. This decomposition simplifies the problem of solving negative interactions between robots. Instead of solving problems for an unknown number of robots, one can focus on solving negative interactions between 2 or 3 robots.Negative interaction between attacking robots could for example cause many failed shoot attempts. A negative interaction is for example when a robot is moving the ball towards the opponent's goal, and its team robot, which also wants to shoot the ball, is hitting this robot causing ball loss.

Furthermore, the classification of different strategies for robot soccer into a formation with a certain tactic can support the system operator in choosing the right strategy against a certain opponent. Formation selection looks at the positioning of the players in the back or forward, this means the way players are distributed on the field.

Tactic selection is focused at how players perform in the back or forward. For example in a 2-1-1 formation there are different state machines suitable for the robot playing in the middle of the field, and when choosing the tactic "long shot" that FSM is selected that let the robot shoot from the midfield, and not the FSM that let the robot pass the ball to the forward player. In the next section, a conceptual design for a 1-3 formation is illustrated.

2.3.2 Conceptual strategy for a 1-3 formation

This section gives a simple concept of a 1-3 formation. The playfield is divided in five different areas and each of the five players is given an exclusive area in which they may operate. First in the 1-3 formation, the field is divided in two parts; one for the defensive and one for the offensive. The offense contains three players, and the offensive part of the field is divided into 3 zones; left, central, right. This is shown in Figure 8.

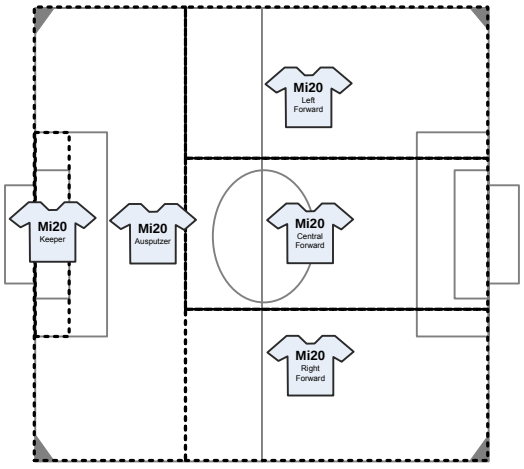

Figure 8 A pure 1-3 formation

If players are only allowed to take action in their areas, the situations in which their actions interact immediately with the other player actions is limited, especially negative interaction is decreased.

The defender's responsibility is to intercept the ball and kick it to the offensive. The attackers try to intercept the ball and drive the ball into the opponent's goal. They only try

to shoot the ball when it is in their zone. Otherwise they return to their home position somewhere in their area, for example in Figure 8 shown by the drawn shirts.

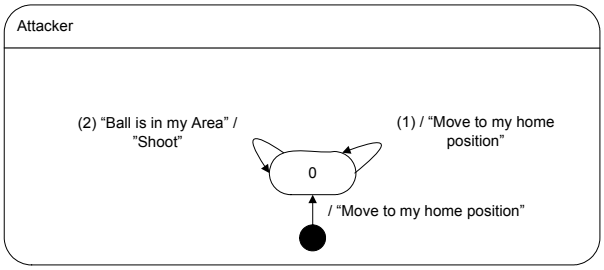

Figure 9 Conceptual FSM for the Attacker

Figure 9 shows a conceptual FSM for the attackers. Table 2 shows the detailed plans for the left forward attacker. The other players have similar plans and conditions.

	Description	***Detailed***
ConditionExpressions	Ball is in my Area	`(BALL_X>0.33)∧(BALL_Y>0.33)`
Plans	Move to my home position	`( REACTIVE_MOVE { x=0.6 , y=0.75 } )`
	Shoot	`SCORE_DIRECT`

Table 2 Description with their details

The other FSM's and substitutions of detailed plans and detailed ConditionExpressions are left open for the reader's imagination.

In the next chapter, a more complex 1-3 formation is described which coordinates behavior between attackers that can operates in the same space.

2.4 Evaluation

This section compares the FSM Approach with the previously researched approaches for the MI20 system. Then it elaborates upon the coordination problem.

Metrics
Some metrics which have to be considered for the strategy module of the MI20 system are adaptability, extensibility, understandability, computational complexity, modeling of time based behavior and reductive explainability (listed in § 1.4). Two approaches have already been under taken for strategy in the MI20 system (summarized in § 1.3). These are the so-called Single Neuron Approach developed by Seesink [1], which has also been implemented and used in tournaments, and the Potential Field Approach by Petit [5]. The metrics used and how they relate to the research question are explained in section 1.4. The argumentation for of the values in Table 3 shall be given in this section. In the end, the FSM Approach shall be evaluated. The Single Neuron Approach, the Potential Field approach and FSM approach will be summarized here with regard to some metrics. After that, the metrics that concerns both methods, at the same way, will be emphasized.

The concept of the Single Neuron Approach has been described in section 1.2.2. The design of the Single Neuron Approach is *adaptable* because reward and penalty values in the design can be changed. Furthermore, the features are parameterized. This means that the features have constants, which can be changed to tweak the features. The system is *extensible* because new features and logic plans can be added. Nevertheless,

a new logical plan could also need new functionality in the motion part to process a new logical plan.

The concept of the Potential Field Approach has been described in 1.2.2. This design of the Potential Field Approach is *adaptable* because the weight factors of the sub functions can be changed. The design is *extensible* because new sub functions can be added. The *computational complexity* of this approach is an important drawback. How many calculations have to be made depends on the resolution one uses on the field, but this is subsequent greater than the number of calculations in the Single Neuron Approach.

The Single Neuron Approach [1] and Potential Field Approach [5] can be considered as transformational systems. They lack support to *time based behavior* so these approaches could never be used when one would create behavior in which different actions have to be performed after some time. For example, timed-behavior can be used as a synchronization method in strategy. Timed behavior could also be useful in test set-ups.

In addition, these methods perform worse in *reductive explainability*. When a robot performs a certain undesired behavior, it is very difficult what has to be changed because many parameters have its influence on the plan generation process. Especially when some parameters or weight is changed, some undesired behavior will disappear but can also cause unwanted behavior to appear. The connectivity between all features or sub-functions with plan generating is too great to be totally overseen by a human. In addition, the used method cannot be decomposed further.

	Single Neuron Approach	Potential Field Approach
Adaptable	+	+
Extensible	+	+
Understandibility	+	+
Low Computational complexity	+	-
Time based behavior	-	-
Reductive explainability	-	-

Table 3 Decision systems of MI20

A “+” means that this metric is being fulfilled, and “-” means that the approach performs worse at this metric.

With the use of the FSM Approach, these last two drawbacks can be solved by the use of state machines. With state based behavior and identical robot soccer field configurations different planning can be performed. State machines can be part of other state machines or decomposed into smaller state machines, which gives humans a better way for explaining observed behavior. Furthermore, a state transition, which depends on one or more features or sub-functions, can easily be split into more states to change behavior for one particular situation. With parallel decomposition, also multi-collaborated behavior can be split into smaller and simpler parts. Also a state machine can be visualized with the use of UML, which will increase understandability. Therefore, this method has an advantage in comparison with the other methods with regard of *reductive explainability*. With adding temporal elements to the finite state machine one can create a so-called TFSM [16], timed finite state machine. With a TFSM, *time based behavior* can be

modeled. With the introduction of TFSM, an event-history can be designed. Upon an event-history, probabilistic analysis can be performed.

The FSM approach defines a strategy consisting of finite state machines which are mapped to the robots, in appendix A.1 alternatives are listed and the reason for this choice is given. Examples of such state machines are shown in section 2.2.1 and 2.2.2. Each finite state machine consists of states and transitions. These states and transitions can be adapted and extended by the designer. But an algorithm to create changes to the finite state machines could also be developed. A transition has conditions, and can create actions. These conditions have to be fulfilled before a transition happens. A condition takes an observation, and this observation has to be above or below some threshold to become fulfilled. A number of these observations used in the system are listed in section 3.1. The observations can also be adapted and extended as well. At strategy level a choice can be made between different finite state machines to compose a team to perform some type of play.

The conclusion can be made that the FSM approach is *adaptable* and *extendible* as well. Because observation calculation, used to determine of a transition may happen, is done for each robot, instead of each position in the field, the *computation complexity* is being considered equal to the Single Neuron Approach and better than the Potential Field Approach. In Table 4, the results for the FSM approach are being summarized and it states that each metric is being fulfilled.

	FSM Approach
Adaptable	+
Extensible	+
Understandibility	+
Low Computational complexity	+
Time based behavior	+
Reductive explainability	+

Table 4 Evaluation for the FSM approach

As can be seen from Table 3 and Table 4 the FSM approach is being considered the best in creating time based behavior and reductive explainability.

Coordination

For reaching coordination, different approaches are possible. Two important dimensions can be distinguished. One characteristic is about distributed or centralized coordination. The second characteristic is about implicit communication are explicit communication.

Distributed coordination is achieved when the plan generating processes for themselves communicate with the other plan generating processes to create coordinated behavior. Centralized coordination is achieved when the plan generating processes uses a centralized mediator to which they send their desired plan and the mediator chooses which plan each plan generating process adopts.

In appendix A.2, a centralized and a distributed algorithm for coordination at agent level are discussed. Neither one of these algorithms are implemented in the current system.

3 Design of different strategies for the MI20 team

For our soccer team, the main objectives are preventing the opponent from scoring and maximizing our own scoring possibilities. When strategies (here we do not mean "strategy modules") are developed, one should consider these to be team goals. This chapter will describe two strategies that have been useful in tournaments. It gives a blueprint of how strategies for a team can be composed of state machines.
It is emphasized that one could create many different strategies and that these two are only particular examples of strategies. Before the two strategies are discussed, the available observations and plans are listed.

3.1 Available observations and plans

This section describes the observations of the MI20 system. The state machine uses the observations to determine the value of its conditions. In addition, the available plans that are used to manage the robots are listed. The state machine can use these plans.

Observations
In Table 6, the available observations are listed. These observations are calculated each time a new WordData arrives, and are used in the input for the state machine.

Observations	Informal description
Ball_in_front	Indicates if the ball is directly in the front of the robot
Ball_is_forward	Indicates if the ball is ahead of the robot
Ball_is_backward	Indicates if the ball is behind the robot
Ball_in_clear_area	Indicates if the ball is next to our goal area
Ball_close_to_own_goal	Indicates in which degree the ball is close to our goal
Ball_moving_to_own_goal	Indicates to which degree the ball moves towards our goal
Has_direct_scoring_oppurtunity	Indicates to which degree the robot can score directly
Ball_pressure	Indicates the threat from the ball
Opponent_direct_scoring_oppurtunity	Indicates to which degree the opponent can score directly
Pass_team_mate	Indicates to which degree the robot can pass to a teammate directly
Opponent_can_shoot_to_goal	Indicates to which degree the opponent can score directly
Ball_moving_to_own_side	Indicates to which degree the ball moves to our side
Ball_lying_still	Indicates to which degree the ball is lying still
Ball_on_their_side	Indicates to which degree the ball is on their side
Ball_on_our_side	Indicates to which degree the ball is on our side
Ball_possession	Indicates to which degree a robot can handle the ball
X_pos	Indicates to which degree the robot is in the back or forward
Should_spin	Indicates if the robot should spin to shoot the ball
In_goal_triangle	Detects a chance to score
Kick_away_from_our_side	Detects opportunities to kick the ball away while defending
Ball_x	Indicates to which degree the ball is back or forward
Ball_y	Indicates to which degree the ball is between the upper or lower bank
Intercept_x	Gives the x-coordinate from the estimated interception point of the robot with the ball.
Intercept_y	Gives the y-coordinate from the estimated interception point of the robot with the ball.
Elapsed_time	Gives the progress of time
Ball_in_triangle_with_own_goal	Indicates if the ball is between the robot and their own

	goal
Ball_in_defender_area	Indicates if the ball is in the robot its defender area
Ball_in_penalty_area	Indicates if the ball in the robot its penalty area
Ball_in_goal_area	Indicates if the ball in the robot its team goal area

Table 5 Observations in the MI20 System

Plans
In Table 6, the available Plans for the MI20 system are listed. These plans represent the strategic means for the players to act in the robot soccer game. The state machine produces plans as output.

Plan	Informal description
Go_behind_ball	Sends the robot behind the ball
Keeper	Lets the robot defend the goal
First_defender	Defend the goal area
Panic_shoot	Shoots the ball and not to its own goal
Intercept	Intercepts a moving ball.
Score_direct	Intercepts a moving ball and dribbles with the ball to a target point.
Block_ball	Deflect a moving ball.
Block_opponent	Obstruct a moving opponent.
Pong	Stay at the same y-coordinate on the middle line.
Offensive_position	Position before a static point before the opponent its goal
Kick	Kicks a moving ball.
Spin	Let the robot spin fast (counter) clockwise
Move	Positions a robot at a certain location
Pen_defender[1,2,3]	Lets the robot defend the penalty area
Score_direct_persuit	Pursuits the ball and with ball possession tries to score
Central_attacker	Lets the robot move on the centre line with respect to the balls x-coordinate.
Halt	Stops the robot.
Reactive_move	Moves the robot to a certain location.
Backup_attacker	Positions the robot between goal and ball.
Run_up_attacker	Positions the robot behind the ball to increase scoring opportunities.
Simple_defender[1,2,3,4]	Positions defenders behind each other to cover the goal line.
Block_defender[1,2]	Positions defenders next to each other to cover the goal line.
Ray_shoot	Shoots the ball to a place on the field.
Secondpile_attacker	Lets the robot move in front of the second pile of the opponent's goal with respect to the balls x-coordinate.
Velocities	Lets the robot move with provided linear and angular speeds.

Table 6 Plans in the MI20 system

3.2 Design of a 1-3 strategy

This section describes the 1-3 strategy with rotational tactic. In the 1-3 strategy, the team is composed of players who perform the roles of keeper, defender or attackers. A 1-3 formation will be described, thus a strategy with 1 defender 3 attackers and a keeper. Furthermore, the attackers will help the defense. Moreover, this type of behavior will bring cyclic (rotational) movement for the attackers. This strategy can be used in the MiRoSot 5vs5 league.

3.2.1 The Keeper

In Figure 7, the state diagram of the keeper is shown. The keeper has a defensive stance, translated into state S_{active} and a passive stance, translated into state $S_{passive}$. When the ball is in the offense; the keeper stays behind and is maximizing the goal coverage. The keeper comes in the defensive stance when the ball is in the defense. In the defensive stance, the keeper has 3 solutions for 3 situations. The first situation is when the ball is in the clear area, the keeper will clear the ball. A clear area is defined as the region next to the goal, in this example this clear region does not depend on the position of other robots. In addition, clearing the ball in (robot) soccer is an attempt to shoot the ball away from the goal. Sometimes in robot soccer, it happens that the ball is stuck between two robots or between the walls and the robot. In the second situation, when the ball is stuck, the robot is commanded to spin, this command often resolves the stuck ball situation. In the third situation, when the ball is not cleared or spun away, the keeper will defend the goal line by staying at a line before the goal line and staying between the goal line and the ball. This is called the Block Ball action.

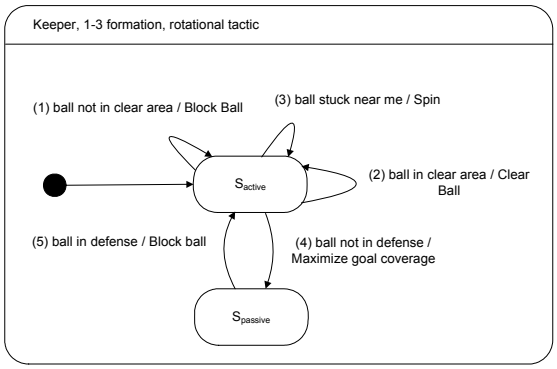

Figure 10 State diagram of the Keeper

In Table 7 the descriptions used in Figure 10 with their corresponding observations from Table 5 and their corresponding plans from Table 6 are shown.

	Description	***Detailed***
ConditionExpressions	ball in defense	`BALL_X > 0.4`
	ball not in defense	`BALL_X <= 0.4`
	ball stuck near me	`(( BALL_LYING_STILL >= 0.7 ) ∧ ( BALL_POSSESSION >= 0.6 ))`
	ball in clear area	`ELAPSED_TIME >= 0.006060`
	ball not in clear area	`ELAPSED_TIME >= 0.003030`
Plans	Block Ball	`BLOCK_BALL`
	Spin	`SPIN`
	Clear Ball	`KEEPER`
	Maximize goal coverage	`PEN_DEFENDER[1]`

Table 7 Descriptions with their details

3.2.2 The Defender

In Figure 11, the state diagram of the defender is shown. This defender has an even simpler behavior than shown for the keeper. It also has an active stance (state S_{active}) and a passive stance (state $S_{passive}$). This defender operates in front of the keeper. The defender will maximize goal coverage when the ball is not in the defense. When the ball is stuck against the robot, the robot will spin to get rid of the ball. A spin plan could also

be created when, for example, the ball is directly approaching the robot and the spin plan is likely to cause the ball to move away from the goal, but finding the right conditions for this situation have not been investigated yet. In other situations, this defender is only meant to defend the goal area by staying at a line in front of the goal area and staying there between the goal area and the ball, this action is called Block Ball. The Block Ball action of the defender operates in the same way, the only difference is that it operates at a different position than the Block Ball of the Keeper. In Figure 11, the strategic plans Block Ball, Spin and Maximize goal coverage are descriptions.

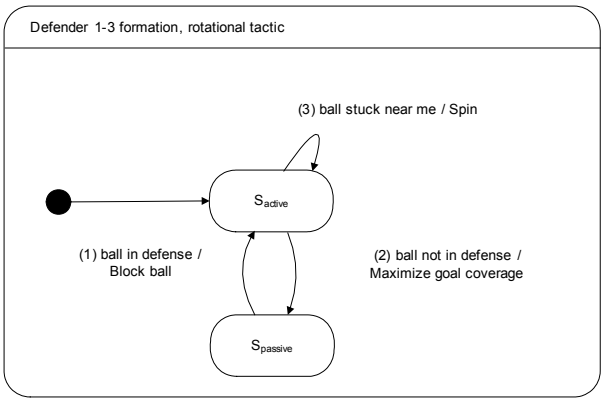

Figure 11 State diagram of the Defender

In Table 8 the descriptions used in Figure 11 with their corresponding observations in Table 5 and their corresponding plans in Table 6 are shown.

	Description	***Detailed***
ConditionExpressions	ball in defense	`BALL_X > 0.4`
	ball not in defense	`BALL_X <= 0.4`
	ball stuck near me	`(( BALL_LYING_STILL >= 0.7 ) ∧ ( BALL_POSSESSION >= 0.6 ))`
Plans	Block Ball	`SIMPLE_DEFENDER[2]`
	Spin	`SPIN`
	Maximize goal coverage	`SIMPLE_DEFENDER[2]`

Table 8 Descriptions with their details

In this section some considerations with regard to this defender are given. One could ask why this defender is never trying to clear the ball. Clearing the ball can be considered very risky for the defender. Clearing the ball is not always successful. This can be due to several reasons, but if it results in missing the ball, it immediately becomes a danger for the keeper. In Figure 12, such a situation is shown. The circled robot is the defender and it did not succeed in clearing the ball, which resulted in an undefended goal area, giving the team of Singapore a better opportunity to score. The risk of clearing also applies for the keeper, but the keeper is nevertheless protected by the boarding and by the defender when the keeper clears the ball.

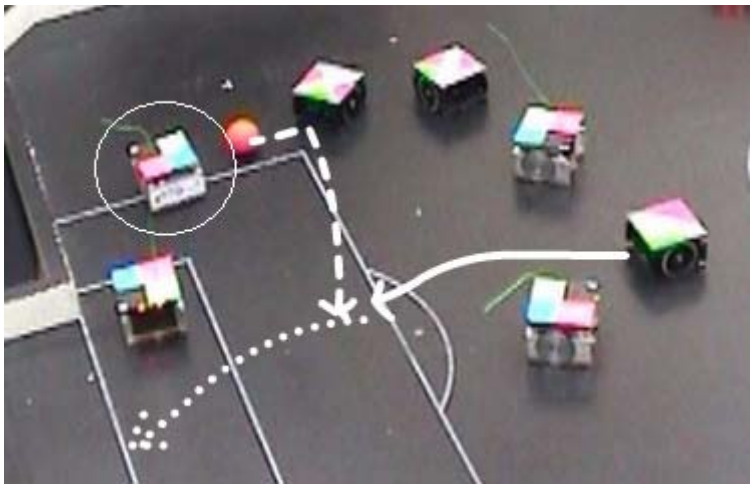

Figure 12 Match against Singapore at 3m 46s, World Cup 2006

3.2.3 The Attackers

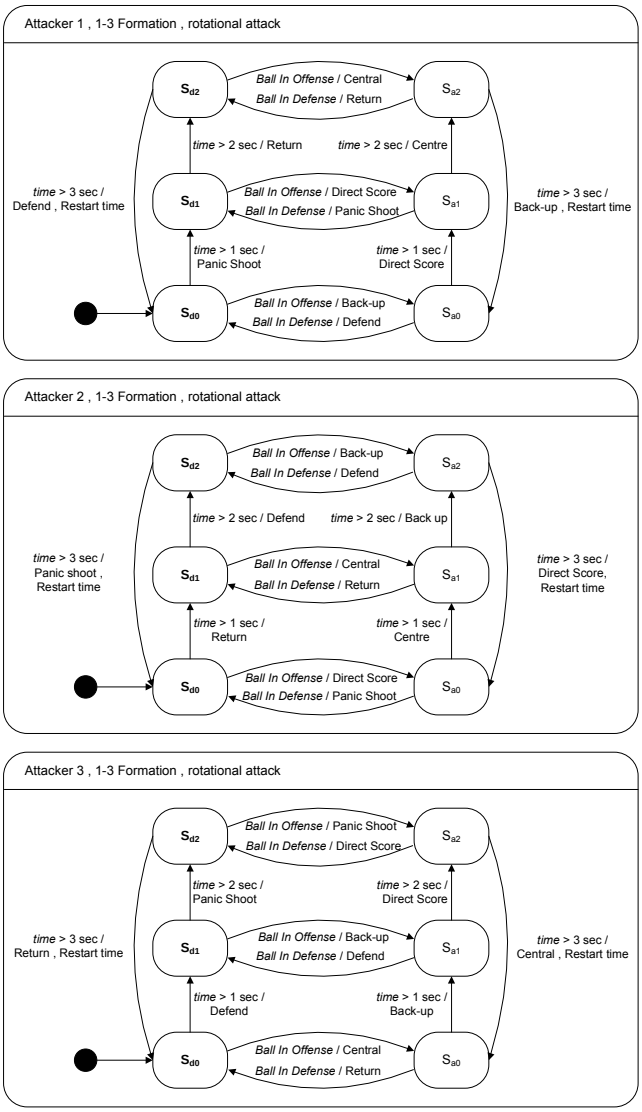

Figure 13 State diagrams of the attackers

In Figure 13, the behavior models of the attackers are shown; the attackers use a rotational way of attacking. Rotation is normally meant as the movement of an object in a circular motion. A rotational way of attacking means that the (positional) target points of the attacker are designed in such way that circular movement for the attacker is commanded: such a movement is sketched in Figure 14, in Figure 13 the behavior models to achieve such motion is shown. Furthermore, it also shows a time-based behavior. The three models have the same structure but they perform different actions when they are for some time in the defense or offensive. This is also listed in Table 10. The time steps have to be at least long enough that the robots have time to travel from one point to another or enough time to intercept the ball and shoot. The distances between the different targets the attacker travels through, in attack or defense mode, are often in a quadrant of the field. It is assumed that the robot can reach an average speed of 1 m/s during team play and with a field that is a little more than 2 meters long, 1 sec

seems to be enough time to reach a next target point. It has some attacking states S_a, when the ball is outside the defense area, and some defending states S_d, when the ball is inside the defense area.

In Table 9 the descriptions used in Figure 13 with their corresponding observations in Table 5 and their corresponding plans in Table 6 are shown.

	Description	***Detailed***
ConditionExpressions	Ball In Offense	`BALL_X > 0.25`
	Ball In Defense	`BALL_X <= 0.25`
	time > 3 sec	`ELAPSED_TIME >= 0.009090`
	time > 2 sec	`ELAPSED_TIME >= 0.006060`
	time > 1 sec	`ELAPSED_TIME >= 0.003030`
Plans	Central	`CENTRAL_ATTACKER`
	Shoot	`SCORE_DIRECT`
	Back-up	`BACKUP_ATTACKER`
	Defend	`PEN_DEFENDER[3]`
	Panic shoot	`PANIC_SHOOT`
	Return	`SIMPLE_DEFENDER[4]`

Table 9 Description with their details

When in the attacking mode, the player moves cyclically through three important points, which are shown in Figure 14, and Figure 15. The attacker tries to shoot. After the shoot attempt, it positions itself in front of the opponent's goal, returns to the back and positions itself at a back-up point. After that, it begins a new attack cycle.

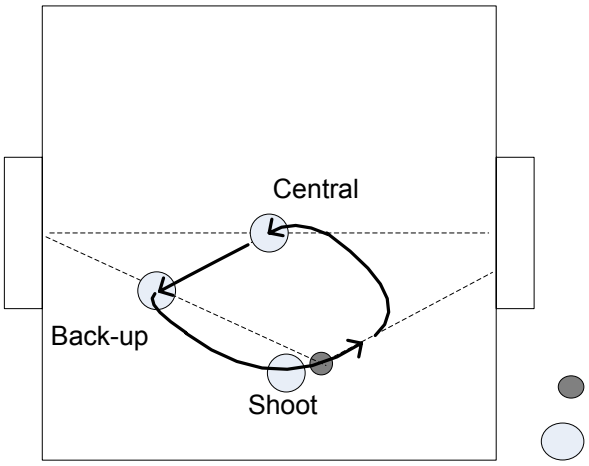

Figure 14 Circular targets for the rotational attacker

When one wants to increase the number of attempts to score, one could decrease the time steps between the moments where a new target point is chosen. Another method is to increase the number of attempts by adding more robots. Due to the time-based behavior, one can keep the number of moments of collision between team-robots low, if the different attackers have different combinations of time with actions. An example is shown in Table 10. Synchronization events for the different attackers are moments when the ball enters or leaves the defense and a same time after which the timer will restart. The timer is based upon the system clock of the operating system, on which all state machines run.

	Attacker 1	Attacker 2	Attacker 3
Time at 0 - 1 sec	**Back-up**	**Central**	**Shoot**
Time at 1 - 2 sec	**Shoot**	**Back-up**	**Central**
Time at 2 - 3 sec	**Central**	**Shoot**	**Back-up**

Table 10 Different time-action combinations for different attackers, when the ball is in the offense

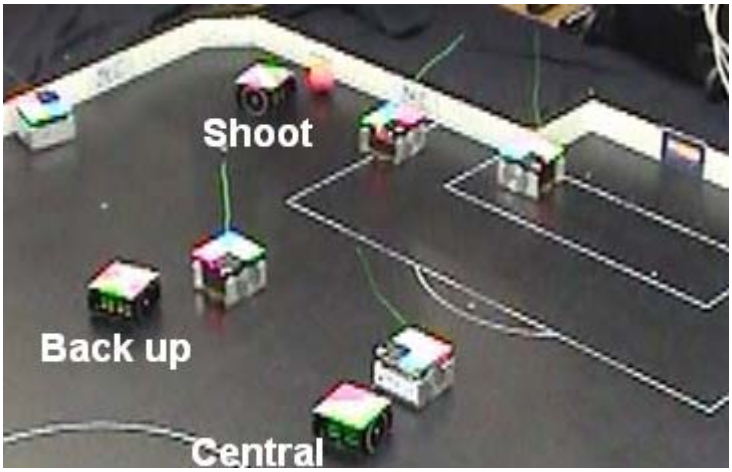

Figure 15 Singapore playing with 3 attackers

The attacking players can also help to clear the ball from the defense. In the defense, also a rotational behavior can be applied. Then the player moves cyclic through three important defensive points, which are shown in Figure 16.

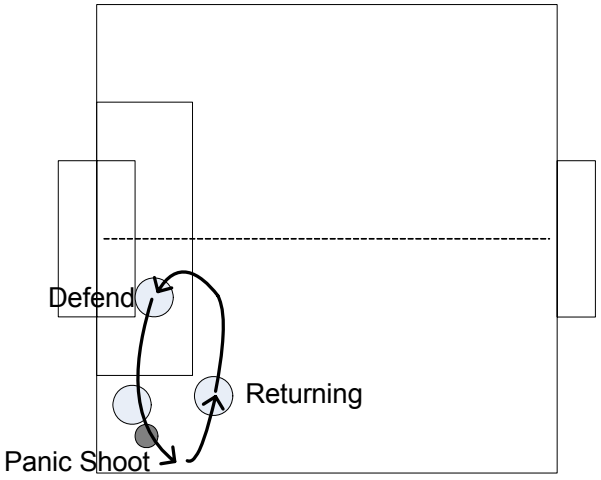

Figure 16 Targets for circular movement in defense

In Table 11, the time-actions combinations for the defensive state of the attackers are listed.

	Attacker 1	Attacker 2	Attacker 3
Time at 0 - 1 sec	**Defend**	**Return**	**Panic shoot**
Time at 1 - 2 sec	**Panic shoot**	**Defend**	**Return**
Time at 2 - 3 sec	**Return**	**Panic shoot**	**Defend**

Table 11 Different time-action combinations for different attackers, when the ball is in the defense

With the time-action combinations for the offense states (Table 10) and defensive states (Table 11) of the player, one can observe two important properties. First, when an attacker in the defense has shot the ball outside the defense area, it will continue to shoot the ball. Second, when an attacker in the offensive performs the back-up action (the most back position in the attack cycle) and the ball enters the defense this attacker becomes the first helping defender. With these properties, a reasonably fast transition between offense and defense is being made. If one compares the plans that the attacker undertakes when the ball is in the defense, one can see a plan “Defend” which is also elaborated by the defender (section 3.2.2). Although the descriptions are the same, not exactly the same positioning is being realized for the different players.

In the behavior diagram for the attackers, one can conclude that when the attacker is in the offensive state, strategic plan generation only depends on time. However, the player execution of a strategic plan can depend on the position and speed of the ball. An example of such plan is a “Shoot”, which results in the creation of control signals that should let the robot intercept a (moving) ball and kick it towards some desired point. When one wishes to create a formation with the rotational tactic for 2 or 4 attackers; for 2 attackers, one could use other timing, but for 4 attackers one could add an additional plan to the behavior of the attackers, in order to prevent robot (players) producing the same plans at the same time.

3.3 Design of a 2-2 strategy

In the 2-2 strategy, the team is composed of players who perform the roles of keeper, defender or attacker. In this chapter, a 2-2 formation will be described, thus a strategy with 2 defenders, 2 attackers and a keeper.

During the EC 2006, all matches were lost to high scores. The offense had no possession because the defense was hardly to stop the scoring attempts of the opponents. By increasing our defensive capabilities, the goal is to create more opportunities and time for creating offensive play. Therefore, a defensive approach after the EC 2006 was adopted focused primarily on performing saves on scoring attempts of the opponent’s teams. The defensive style of play is characterized as ‘low-pressure’. ‘Low pressure’ defense results in passive positioning in the back focused to prevent penetration of the defense and minimize chances for the opponent to out-dribble defenders.

The attackers will take a passive stance when the ball is in the defense, when the ball is in the offense one attacker will be moving at a supportive position while the other will operate as the on-ball player. In the pictures (in this chapter) where robots are shown, the robots with a blue rectangle on top are opponent robots, the others are the robots of the MI20 team.

3.3.1 The Keeper

The main task of the keeper is to stop the ball before the goal line. In addition, the main task for the defenders is to stop the ball before the keeper. Furthermore, the keeper tries sometimes to clear the ball away from the goal without losing the ability to return fast to its position in front of the goal line. The main idea behind the defensive approach was that it would be better to win with 0-1 than to loose with 21-10.

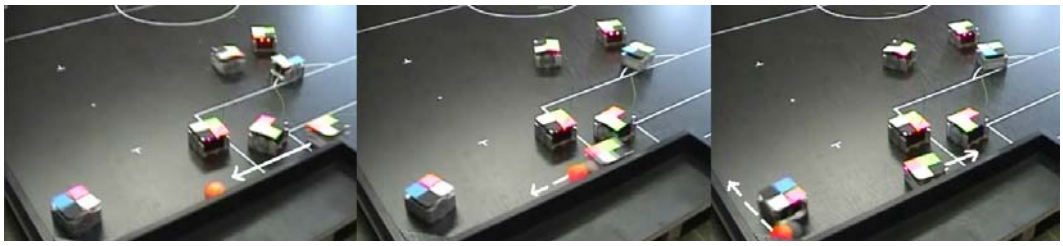

Figure 17 Successful clearing

In Figure 17, a filmstrip is shown when the keeper clears the ball. The keeper clears the ball only when the ball is next to the goal area. 'Clear'-actions directed forward are not performed by the keeper to reduce situations in which the keeper is obstructed by own players. Because of the triangles in the corners of the field, a 'clear' action to the side can still cause the ball to move towards the offense.

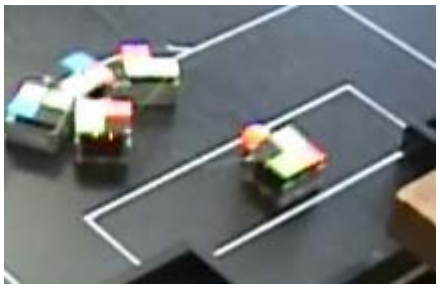

Figure 18 Successful save

In Figure 18, the most important job for the keeper is shown: stopping a ball that would otherwise cross the goal line. Cruijff once said; "*Football is simple. You are in time or too late. When you are too late, you should start sooner.*", therefore the keeper does not always position himself exactly behind the ball but also drives directly to the interception point of the ball with the goal. In robot soccer sometimes an opponent player pushes the goalkeeper with the ball over the goal line. Such an action is considered a foul for the opponent and a goal kick is given to the keeper's team.

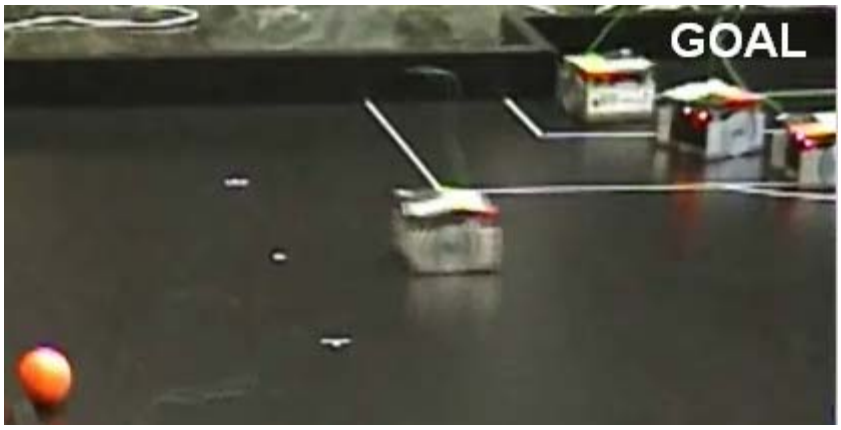

Figure 19 Maximizing goal coverage

In Figure 19, the defending robots are positioned such that the goal coverage is high. If the ball should suddenly move fast toward the goal, it is hoped that at least one of the robots will be on time to intercept the ball in front of the goal.

The behavior for the keeper used in the 2-2 using wings is the same as used by the 1-3 formation with the rotational tactic described in section 3.2.1.

3.3.2 The Defenders

In the 2-2 formation with 'using wings' tactic, the defenders adopt a low pressure style of defense. This means that the distances to the own goal are kept small. In a low-pressure style of defense, the distances to opponent players and ball are kept small. In Figure 20 the positioning of the defenders (robots 2 and 3) are shown and their small distances to the own goal. This is the positioning of the defense when no actual threat is detected.

Figure 20 Low pressure defense

When the ball has entered the defense, the most forward defender can take action to shoot the ball toward the offense by means of the plan SCORE_DIRECT. This is planned when the ball is not moving too fast and when the ball is not between the own goal and the defender, as shown as transition T1 in Figure 22. This is because the 'shoot plan' execution becomes inaccurate when the ball speed increases, furthermore driving the robot around the ball seems to be very difficult. The rearmost defender takes only a

shoot action by means of a PANIC_SHOOT when the ball enters the penalty area and the ball is not between the goal and the defender.

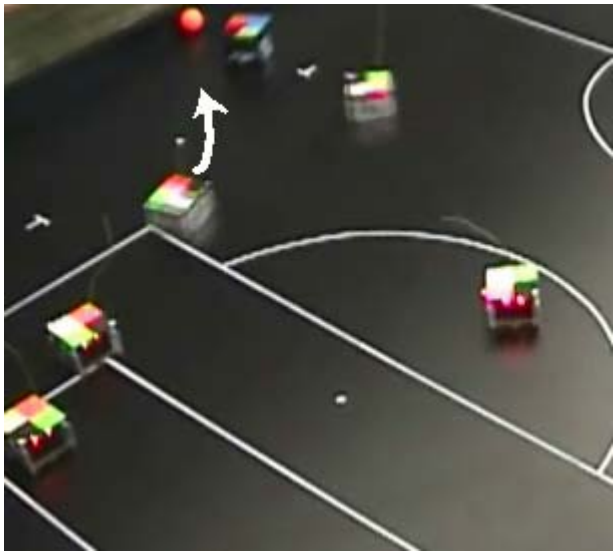

Figure 21 Defender taking offensive action

When the defenders do not deliberate upon an on-ball plan, they intend to plan the PEN_DEFENDER, which drives the robot in straight lines around the goal area at some distance, whilst keeping space for the goalkeeper to clear the ball as can be seen in Figure 17. The calculation of the desired position for the robot is shown in section A.3.1.

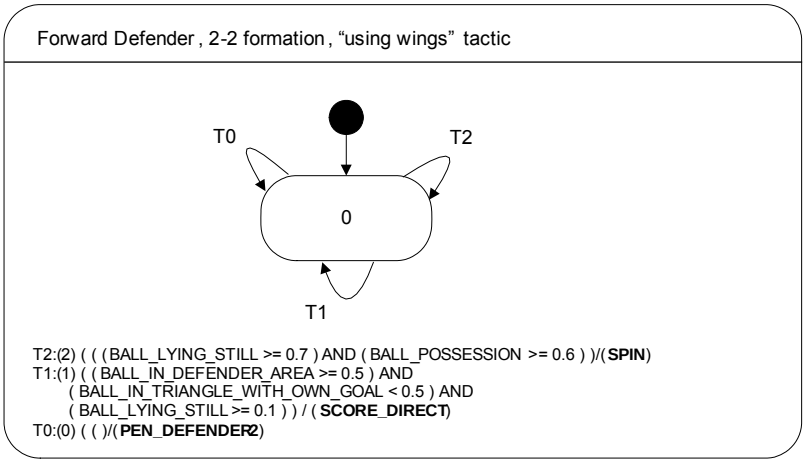

Figure 22 State diagram for the forward defender

In Table 12 the transitions in Figure 22 are listed with their rationale. The specific values in the conditions have been tuned during experiments and soccer play.

Transition name	Rationale
T2	If the ball lies still against the robot, which is often a stuck ball situation, the robot should spin.
T1	If the ball is in the defense and not moving too fast and not between the robot and its goal, the robot will shoot.
T0	Default, the robot positions as the defender, which, is most far positioned from the goal, but still within the penalty area.

Table 12 Transitions with their rationale

3.3.3 The Attackers

The attacker's behavior is based upon the ball position. The left forward player, which is shown as robot number 5 in Figure 24 and Figure 25, is considered first. The behavior of the right forward, shown as robot number 4, is the same but mirrored along the length of the field. For the left attacker the ball's position is classified to be in one of 4 areas, as shown in Figure 23.

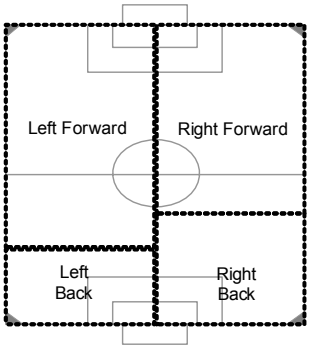

Figure 23 Quadrants for the left forward attacker

One area is left back, when the ball is behind 25% of the field length and in the left part of the field, shown in Figure 24. The second area is left forward, when the ball is in front of 25% of the field length and in the left part, shown in the left situation of Figure 25. The third area is right back when the ball is behind 37,5% of the field length and in the right part of the field. The fourth area is the when the ball is in front 37,5% of the field length and in the right part of the field, the right forward situation is shown in the right situation in Figure 25.

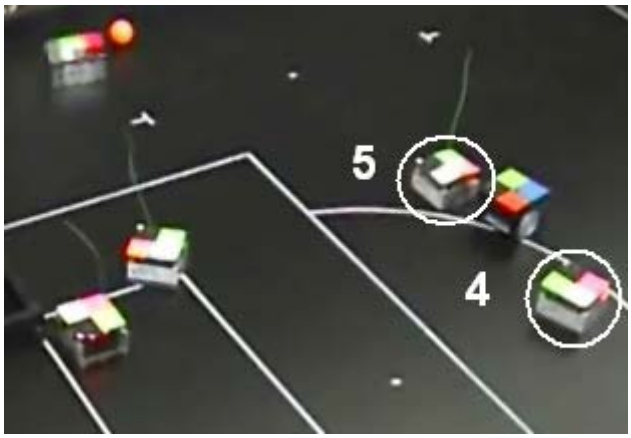

Figure 24 Attackers in passive stance

In Figure 24, the passive stance of the attackers is shown, the attackers are passive when the ball is in the defense (left below is our goal). The attackers are positioned some distance before the penalty area and are waiting for the defense to get rid of the ball.

In Figure 24, the left forward attacker (robot 5) positions itself on 25% of the field length and 65% of the field width. If the ball is in the right back, the left forward attacker positions itself at 25% of the field length and 50% of the field width.

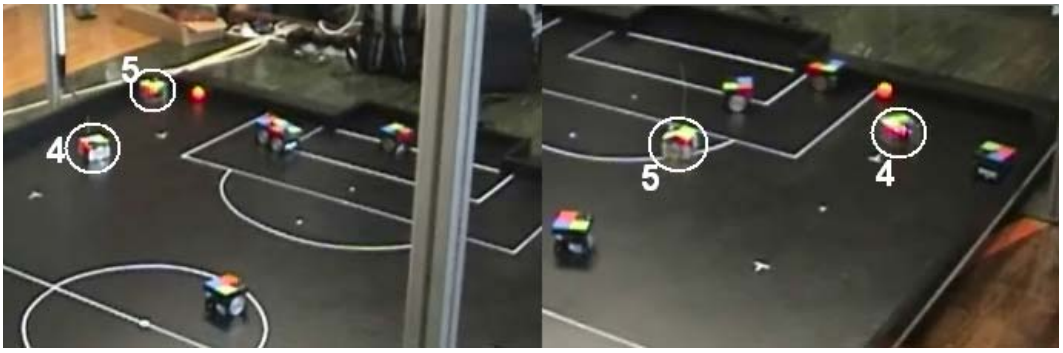

Figure 25 Attackers in the active stance

In Figure 25, two situations are shown. In the left situation, when the ball is in the left forward, the left forward attacker tries to shoot the ball by means of the plan SCORE_DIRECT. The right attacker, positions between the own goal and the ball at a distance of 600 mm behind the ball. This is done by means of the plan BACKUP_ATTACKER.
In the right situation the roles are switched, the right forward attacker plans the SCORE_DIRECT and the left forward the BACKUP_ATTACKER.

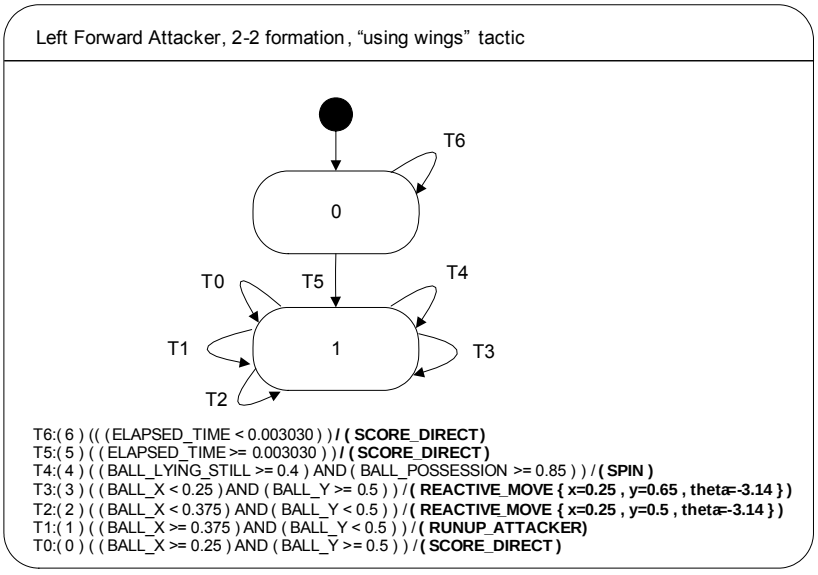

Figure 26 State diagram for the left forward attacker

In Figure 26, the state diagram of the left wing attacker is shown. The structure of the state machine of the right wing attacker is similar to the left wing attacker.

In Table 13, the transitions in Figure 26 are listed with their rationale. The specific values in the conditions are tuned with experiments and soccer play.

Transition name	Rationale
T6	When the game is started, the attacker always starts with an attempt to score the ball. This is because an attacker is often positioned close to the ball during special game situations, for example a kick off.
T5	This transition is only to change the state of the attacker to normal play.
T4	If the ball lies still against the robot, the robot will spin. Such a situation is often a stuck ball situation.
T3	If the ball is left back, the defense has to get the ball to the offense and this robot positions itself in a position in front of the penalty area.
T2	If the ball is right back, the defense has to get the ball to the offense and this robot positions at a position in front of the penalty area. Notice the difference of conditions between T3 and T2 with regard to the BALL_X observation. This difference exists because if the ball is right forward, this player adopts the BACKUP_ATTACKER plan, which positions the robot behind the ball and the planning is careful with regard to getting too many robots in the penalty area.
T1	If the ball is right forward, the attacker takes a supportive role in the offense.
T0	If the ball is left forward, the attacker takes an active role in the offense and tries to shoot the ball into the goal.

Table 13 Transitions with their rationale

3.4 Evaluation

In the soccer game, offense and defense are separated by distinguishing two goals. The offense has as primary goal to move the ball into the goal of the opponent. The defense has as primary goal to prevent the ball entering their goal. When the ball is being moved from the own goal straight to the opponents goal it can be considered both offensively and as defensively beneficial.

On the other hand, there are also plans available with low offensive and defensive value: an example of such action could be standing still at the border in the middle of the field. A trivial example in a 5vs5 game is that of a team with all players staying still at the side borders. This team should already in offensive and defensive aspect be beaten by a team of two players with one keeper and an attacker which only pursuits the ball and tries to shoot directly at the opponents goal.

Next a look on how the 1-3 formation with the rotational tactic (§ 3.2) can be more offensive and more defensive than the 2-2 formation tactic (§ 3.3) is given.

Two players form the offense of the 2-2 formation, the first player controls the left half in front of the opponent's goal and the second player controls the right half in front of the opponent's goal. The offensive operates in the following manner: first, when the ball is on the left part, the first player tries to shoot the ball toward the opponent's goal. The second player is moving at a point behind the ball. Furthermore, the second player starts immediately starts a shoot attempt when the ball moves to the right half. Second, when the ball is on the right part, the second player tries to shoot the ball toward the opponent's goal and the first player positions itself behind the ball.

With this strategy, always a player tries to shoot the ball when the ball is in front of the opponent's goal.

Unfortunately, shoot attempts can result in situations in which the player completely misses the ball. This is where the 1-3 formation with the rotational tactic (§ 3.2) becomes more offensive. The 1-3 formation is able to generate more shoot attempts per second than the 2-2 formation (§ 3.3). Furthermore, this strategy also returns robots that attempted to shoot, to positions behind the ball, which are supposed to be more favorable when shooting, toward the opponent's goal, starts. For shooting in the defense the same argument holds, furthermore most of the time 3 robots are positioned in the penalty area while in the 2-2 formation most of the time 2 robots are positioned there because the other defender robot attempts to shoot the ball, misses and drives somewhere else. This is illustrated in Figure 27; robot 1 is the keeper, robots 2 and 3 are defenders and robot 4 and 5 are the attackers.

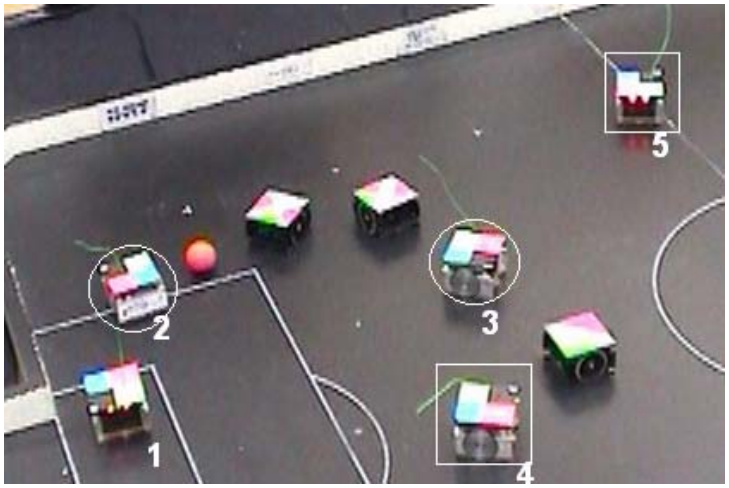

Figure 27 Defenders 2 and 3 missed the ball

Both strategies have been used during the EC 2007. In matches played against VSB-TUO from the Czech Republic, the 1-3 Formation was indeed more effective than the 2-2 formation. In matches against TUKE robotics with the 1-3 formation more pressure to the ball could be kept, but our transition from offense too defense was to slow which caused a higher change of success for long dribbles (for the TUKE) because in our defense only one defender was present. This could be solved by adjusting the Back Up position more to the back. However, in matches against TUKE robotics the difference of effectiveness between the two strategies was not easily observable. Mainly, because the ball interception of TUKE robotics has a higher success rate and is faster. In Table 14, the main advantages and disadvantages for the discussed strategies are listed.

	2-2 formation *zone based offense*	**1-3 formation** *rotational tactic*
Advantages	Efficient against long dribbles.	Able to recover from failed shoot attempts.
Disadvantages	Low pressure on the ball.	Vulnerable for long dribbles from the opponent.

Table 14 (Dis) advantages of the different strategies

From the results of played matches, future work for a 2-2 formation with rotational attack is recommended. In addition, increasing the interception speed as interception accuracy is recommended. Nevertheless, matches against a weaker opponent also leaves room for the development for a 0-4 formation.

4 Underlying systems

This chapter describes additional modification to the MI20 system, used to provide the Strategy module with consistent information and adequate control signal calculation.

4.1 Robot identification

Robot identification was improved by altering the Vision module. By using different color patches, as seen in Figure 28, direct identification of the robots on the field is possible.

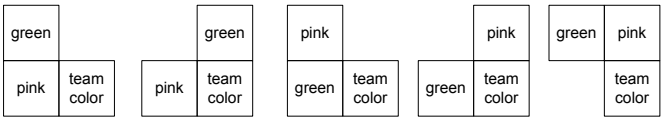

Figure 28 Different color area arrangements for the color patches

Before robot identification is explained, some terms are explained:

Blob
A blob is a data structure that contains the position information of the barycenter of a recognized color area on the soccer field. The color segmentation [20] in the vision module determinates the blobs that consist of clusters of adjacent pixels with similar color.

Blob combination
A blob combination is a tuple, with a green blob, pink blob and a team color blob.

Match function
The match function calculates a value that represents the correspondence of a given blob combination with a given color patch.

To recognize a certain color patch two steps are taken:

Step 1:
For each blob combination, the matching, in terms of an error value, is calculated. The blob combination with the minimal error is selected.

Step 2:
From the selected blob combination, the centre and the orientation is determined.

This approach has as disadvantage that wrong blob combinations can be associated with a certain color patch. Different patches positioned against each other can cause such a wrong association.

Another method for patch recognition could use least squares fitting.

4.2 Motion

This section describes two additions to the motion module to improve robot control. For the new version of the system only a proactive motion planning was developed by Buth [9]. There was a problem with the proactive planning in combination with, for example, the collision avoidance or actions used to follow the ball. Collision avoidance

caused frequent request of re-planning for the motion module. The proactive motion module could not adequately handle re-planning and the robots showed slow speeds and response.

4.2.1 Proportional Cosine control

With proportional cosine control [17] the linear velocity is proportional to the distance error d. Also the angular velocity is proportional to the angular error θ_e. See Figure 29.

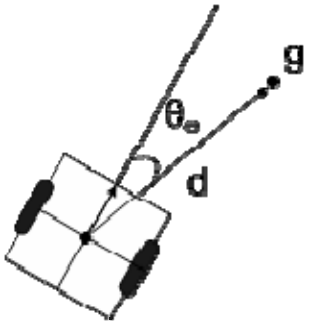

Figure 29 Cosine control

The following equations show the determination of the linear and angular velocities.

$$V_{lin} = K_p \cdot d \cdot \cos(\theta_e) + v_c$$

$$V_{ang} = K_\theta \cdot \theta_e$$

K_p and K_θ are called the proportional and angular gain, and are constant. v_c is a small constant velocity which can be used when we want to move the robot through some point instead of stopping at the goal position. This can be the case when a robot is moving towards an intermediate point in a planned path.

To help understand the working of these control law equations, consider the robot being oriented away from the target point g, with $\theta_e \pm \frac{1}{2}\pi$. In this situation, the linear velocity is limited by the cosine and a large angular velocity is calculated. In contrast, when the target point is in front of the robot and the angular error θ_e is small, the linear speed can reach its maximum height. The values for the gain are set to K_p=0.3 and K_θ=8.0. Determination of proper values and also the influence of dead time to the use of a proportion cosine controller should be investigated further.

The problem encountered with the proportional controller is that one has 2 options:

- High gains:
  Results in fast speeds for short targets, but in case of for greater distances unrealistic control signals.
- Low gains:
  Results is realistic control speeds for greater travel distances, but too slow speeds for short distances.

One would have a controller which maximizes the acceleration of the robots and has fast speeds for short distances in combination of realistic speeds for great distances.

4.2.2 The pursuit move function

In the pursuit approach [18] the main idea is to travel circle wise through the goal point *gv*, as seen in Figure 30. To accomplish this, the curvature of the circle is calculated and from the curvature the control signals (ω and *v*) are obtained. The curvature C of a circle is defined as the inverse of the radius *r*.

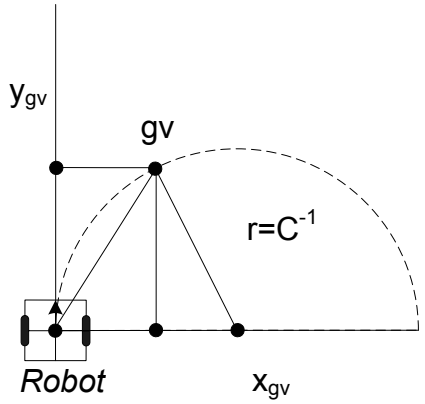

Figure 30 Pursuit approach

The calculation of the control speeds involves different steps:

Step 1:
Transform the goal point to robot coordinates. The transformed target is called *gv* with coordinates (y_{gv}, x_{gv})

Step 2:
An appropriate circle arc, with a certain radius *r*, is determined for the robot to travel. The aerial distance *D* from the robot to *gv* is given by:

$$D = \sqrt{y_{gv}^2 + x_{gv}^2}$$

The curvature C, which determines the ratio between ω and *v. C* defines the desired circle arc, and is given by:

$$C = \frac{\omega}{v} = \frac{2 \cdot x_{gv}}{D^2}$$

Step 3:
An adequate linear speed *v* is calculated. Default the maximal linear speed, geared by the system user, is used. However, when the robot has to stop at the given goal point, the linear speed is reduced as the robot comes closer to the goal point. This speed is limited by a factor *s,* which is made dependant on the squared distance as follows:

$$s = \frac{D^2}{c_a^2 + D^2}$$

With c_a=129 reasonable results have been obtained. When D^2 is zero *s* becomes zero, and the robots stops. On the other hand, when D^2 is great s approaches the value 1 and the linear speed is almost not limited.

Step 4:

An adequate angular speed ω is calculated. As long as the robot has not reached the goal point, the angular speed is calculated according to the linear speed and the radius of the circle. Otherwise, the angular speed is calculated to pose the robot to a given orientation.

Step 5:

The last step involves slippery prevention; this has also been investigated by Brandenburg [23]. In addition, the linear and angular speed is limited to prevent slippery of the robot. This means that for high curvature the linear and angular velocity has also to be reduced. Slippery for a robot, with mass *m,* can be caused by the centrifugal force. The centrifugal force can be calculated with:

$$F_c = \frac{m \cdot v^2}{r} = m \cdot \omega \cdot v \Rightarrow a_c = \frac{F_c}{m} = \omega \cdot v$$

The speeds have to be limited in such way that the centrifugal acceleration a_c is not too great. A maximal allowed centrifugal acceleration a_{max} should be determined experimentally. Furthermore $a_c < a_{max}$ should apply to the control speeds. If a_c is greater than a_{max} a limitation factor k can be calculated, in such way that the next equation holds:

$$a_{\max} = |k| \cdot a_c = |k| \cdot \omega \cdot v = \left(\sqrt{|k|} \cdot \omega\right) \cdot \left(\sqrt{|k|} \cdot v\right)$$

The limited speeds control speeds are calculated with:

$$\omega_l = \left(\sqrt{|k|} \cdot \omega\right), v_l = \left(\sqrt{|k|} \cdot v\right)$$

The problem encountered with the proportional controller arises mainly at strategy level. Sometimes, this pursuit method results in driving the robot along too small circle arcs, causing the robot in turning behavior. When this happens for the robot with the keeper role, the keeper loses its ability to react fast enough on a change of ball direction. The noise in the Vision system probably causes inconsistent radius calculation at close target distances.

4.2.3 Evaluation

This section compares the control methods. In Table 15, the main advantages and disadvantages of the methods are listed.

	Proportianal Cosine Control	**Pursuit**
Advantages	Can be used to reduce angular error and distance errors quickly.	- Constant curvature - Exact calculation - Slip prevention - Stop profile
Disadvantages	The use of constant gains results in bad control signals if used for both far or close targets.	Caused unwanted circular motion for the goal keeper

Table 15 Comparision between proportional cosine and pursuit control

In the Mi20 system, for reactive motion mainly the pursuit controller is used. For defensive plans, which tries the keep a robot moving on a single line, a proportional controller is uses. Therefore, the proportional controller is mostly used for close targets, and then it works fine.

4.3 State estimation

This section describes two alterations to the state estimation process of the MI20 system. First, the change of robot association in the state estimator module is discussed. Second, the use of state prediction is discussed.

Robot association
Robot association is the association of a robot id with a robot pattern observed in the image having some location and orientation. Kooij [20] created the vision part of the MI20 system that uses identical patches for each robot. To know where which robot is, Kooij created in the State Estimator a nearest neighbor algorithm for robot association.

With the introduction of robot identification by means of the use of distinct individual patches as described in section 4.1, the nearest neighbor algorithm is not needed. When using the new patches, the Vision module incorporates the id of the detected patch into the Snapshot. The State Estimator maintains the association of position and orientation with the provided id when it creates the WorldData object.

State prediction
The smith predictor is mainly used to overcome the problems that arise with dead time. Dead time is in our system the time between the moments a control signal is send, and when the control effect is observed. Smith predication applied to robot soccer has been investigated by Koay and Bugmann [19].

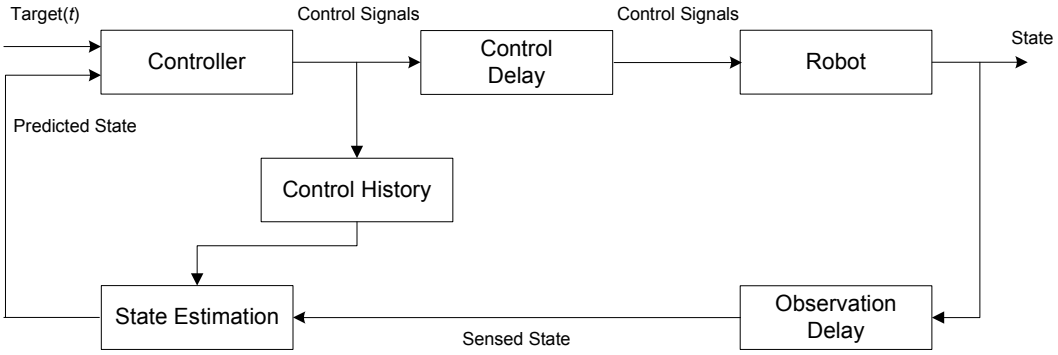

Figure 31 Smith prediction in MI20

In Figure 31 smith prediction is shown for a single robot. To encounter for the delayed observation (of the send control signal), the control signals are stored in a Control History. The state estimation takes the sensed state of the robotsoccer field, and predicts kinematics behavior using the control signals from the Control History. This predicted state provides in a more accurate state, and is more useful for the motion controllers.

Determination of the dead time is not done automatically yet. It is determined by the user with the aid of a real time monitoring diagram in which the predicted and the observed control signals are shown. When the two graphs coincide more or less, the delay is set appropriately.

Kalman filtering
The currently available Kalman filter in the system does not show any additional positive effect. It is likely that Kalman filtering improves when it is implemented using the control history. For the time being, the Kalman filter should be disabled during a soccer match.

4.4 Collision avoidance

An important reason for a robot not reaching his target is that another robot obstructs the route to the target. Another reason is that a collision causes the robot to move at some unintended manner. For testing of strategies or for a free flowing game the recovery from collisions and collision avoidance are very important. A mechanism for collision recovery is described in appendix A.4.

Also for automatic positioning, collision avoidance is needed. Obstacle avoidance is the task to reach a control objective without colliding. This is, most of the time, distinct from path planning, which involves control laws for driving. When a collision free path has been computed, but a path planner is not able to let for example drive a robot through that path, collisions may still occur. Nevertheless, this is considered as a path planner fault.

Obstacle avoidance has to pre-compute a collision free path. When one has in the dynamic situation an idea of where obstacles are moving to, their trajectories can be taken into consideration (dynamic avoidance). When one has no idea of what an obstacle is going to do, only the location of the obstacle is being considered (static obstacle avoidance). Van der Linden [21] has performed research with regard to dynamic avoidance in the MI20 project. Next different methods are discussed and finally these methods will be evaluated.

4.4.1 Role switching

When starting with building a strategy and playing or positioning without an opponent team already collisions appear. So collisions between team robots happen.

These collisions may even cause deadlock situations. Collisions sometimes occur when 2 robots want to go to a position where the other robot is closer to. An easy way to prevent this collision is to switch the targets of the robots. Because the targets depend on the role of the robot (e.g. keeper) the same effect can be reached by switching the roles of the robots.

An overall method for role switching was developed that minimizes the traveling distance between the different targets of the robots and the locations of all the robots.

This method performs three steps.

Step 1
A distance table, from the robots to the different targets, is created.

Step 2
All different permutations of mappings from robots to targets are evaluated on the summated traveling distance.

Step 3
The third step involves selecting the mapping in which the summated distance is minimal.

4.4.2 Local vector field avoidance

During the process of strategy engineering, matches with opponent robots caused also collisions. For these collisions, some form of resolution had to be created. First, risky situations had to be detected, and second the robot should avoid this risky situation.

A robot *g* has a target to which it has to move and has to consider other robots to which it can collide. The avoidance technique of this method uses the following steps.

Step 1
Robots that are in some radius of robot *g* are considered candidates with which robot *g* could collide. For these robots, repelling vectors are calculated.

Step 2
For the target of the robot *g,* an appealing vector is calculated.

Step 3
The repelling vectors and the appealing vector is combined, and the combined vector is used to create the direction in which the robot has to move.

4.4.3 Global vectorfield avoidance

Another method to resolve risky situations, with regard of collisions, considers all robots and walls. A vector field can be created to calculate for each position on the field the direction of the repelling force of the robots and the attracting force of the target. In Figure 32 a grid of positions is shown as dots with their corresponding vector represented as a line.

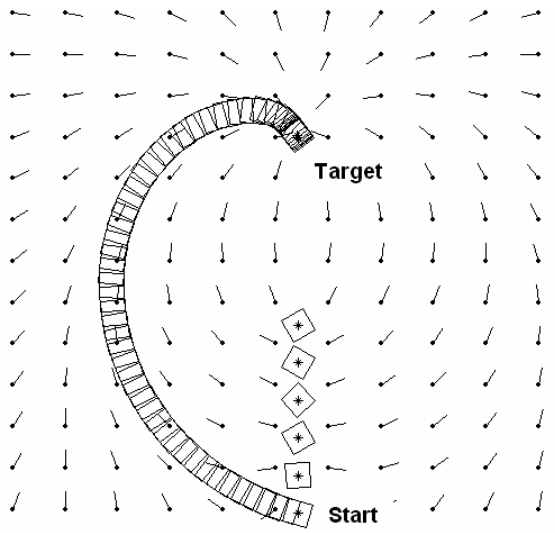

Figure 32 Robot driving through a vector field

Figure 32 shows a robot *g* that has to drive from its start position towards its target position.

Step 1
For all other robots, besides robot *g*, are repelling vectors calculated. To prevent the robot to be directed through walls also repelling vectors for the walls are calculated. The repelling vectors have a weight w_n that is proportional with the inverse squared distance d_n of the other robot or wall to robot *g*. For w holds: $w_n \sim d_n^{-2}$.

Step 2
For the target of the robot *g,* an appealing vector is calculated.
The appealing vectors has a weight *w*. For *w* holds: $w \sim d^{-1.5}$.

Step 3
The repelling vectors and the appealing vector are combined, and the combined vector is used to create the direction to which the robot has to move.

4.4.4 Single obstacle avoidance

This method is straight forwarded. When the straight line from the robot to a target position is crossing obstacles, the closest obstacle is being avoided by calculating an intermediate target next to this obstacle, as shown in Figure 33.

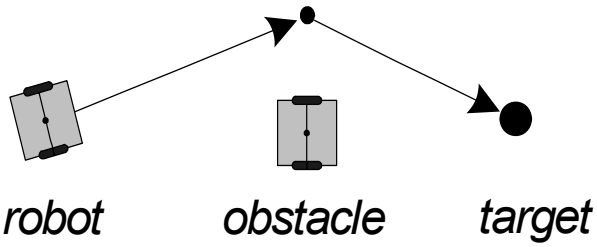

Figure 33 Robot driving around the obstacle to its target

The intermediate target is positioned perpendicularly to the straight line from the robot to its target. If the obstacle, from the robot point of view, is on the left of the straight to its target, the intermediate point will be positioned right. If the obstacle, from the robot point of view, is on the right of the straight to its target, the intermediate point will be positioned left. The distance from the intermediate point to the obstacle has been chosen 120 mm (about one and a half robot's width).

For this method the opponent robots are considered as static obstacles. Nevertheless, for team robots their (intended) movement is taking into account. When the team robot that is crosses desired path of the robot is about to leave before a collision happen, there is no need for the robot to avoid the moving obstacle. Figure 34 shows this situation.

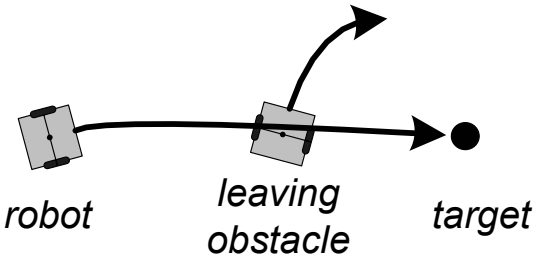

Figure 34 Robot driving straight to its target

4.4.5 A* avoidance

This method has been adopted by the current world champion Socrates from Singapore [17]. The robot is modeled as a hexagon. A collision free path is calculated with an A-star algorithm. This is done within each control cycle.

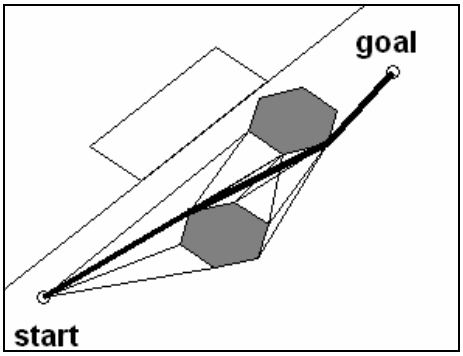

Figure 35 A* path search trough hexagons

The A*-search is based a visibility graph [22], which finds an shortest path (in the graph) if it exists. The basic principle is to expand nodes successively beginning at the start node until the goal is reached. Nodes are expanded by means of edges which connects the nodes. In this way additional reachable nodes are found. The expansion of the nodes can be directed by an evaluation function *f(n)*. The node with the lowest value for the evaluation function is expanded first. Next, the key elements of this algorithm are discussed with regard of the application in the robot soccer domain.

Nodes

Nodes are created in several steps. First around each robot, 6 points are calculated considering a bounding circle for the robot. Second, we have to validate that the calculated points are located in the field and that the calculated points are not in the bounding circle of other robots, then the points can be considered as suitable nodes.

Edges

Straight lines between a reachable, so called opened node, and all the other nodes are created and when a line is not in the bounding circle of a robot, the line is considered as an edge. The other node connected, by this edge, is supposed to be a *successor* (or *child*) of the opened node, the latter being the *parent* node.

Evaluation function *f(n)*

When a successor node *s* is determined, the cost value *f(s)* is estimated. The total cost estimate from the *start* through parent node *n* and *s* to the goal point is calculated by

f(s)=g(s)+h(s). The cost estimation *g(s)*, which estimates the cost from the *start* to *s*, is computed by *g(s)=g(n)+distance(n,s)*. For calculating *distance(s,n)* we use the Euclidean distance between *n* and *s*. The cost estimation h*(s),* from *s* to the *goal,* is the aerial distance between *s* and *goal*.

The search terminates if there are no nodes to be expanded anymore or when a path to the *goal* has been found.

To find the nodes of the path, the chain of parent nodes of the *goal* node is traversed backwards. When a found parent is the *start* node, the current node being investigated is the first node where the robot has to travel to.

4.4.6 Evaluation

In this section an overview of the main advantages and disadvantages of the methods is given.

Role switching works fine to avoid collisions between team robots. Nevertheless, in the current implementation role switching can cause risky strategic situations (e.g. empty goal). It provides an outstanding way to prevent collisions between team robots but has undesired effects at strategy level. Further research of the benefits of role switching could be focused on as a mechanism for collision recovery. It could make sense to apply role switching only when a collision will happen in the near future. Role switching could also be used between the transition between offensive team play and defensive team play or to minimized time for ball interception.

Radius based, local vector field avoidance works fine to avoid near robots, but progress towards its targets is often too slow. Some important flaws were visible such as, oscillating behavior. This behavior became visible when a robot, which was avoiding obstacles, stopped considering them when they became out of range. Then this robot drives directly again into the critical area. Another flaw was that robots could be 'scared away' from their target position.

Global vector field avoidance works fine to avoid single obstacles in a fluent way and keeping progressing towards its target. Nevertheless, avoiding groups of obstacles can cause a circuitous route for the robot. When robots are closest to their target it rarely happens that a new direction is calculated that moves away from their target. This method worked very well with auto-positioning. A positive point of this method is that the robot avoids also groups of obstacles. Nevertheless, there are still field configurations possible, which tend to drive the robot away from his target. This problem arises due to a stronger effect of the repelling forces of the obstructing robots, not that it was impossibility to reach the target. This latter problem became important when our robot discarded some scoring situations where the ball is surrounded by enemy defenders.

Single obstacle avoidance leads to an optimal path when only one obstacle has to be avoided. Nevertheless, in the robot soccer game sometimes multiple obstacles have to be taken into consideration to find a reachable path towards its target. The main disadvantage of this method is that the robot can get funneled by obstacles and will be unable to reach his target.

A*-avoidance obtains an optimal path considering multiple obstacles. There is one disadvantage; this method does not take into account the movement of the obstacles so there are situations in which avoidance could have been more efficient.

Table 16 shows the main advantages and disadvantages for the described methods.

	Advantage	Disadvantage
Role switching	Minimized traveling-time for all the robots	- Risky situations in the robot soccer game - Works only for team robots
Local vector field	Intuitive concept for obstacles avoidance	Repetitive behavior due to 'forgotten' obstacles
Global vector field	Fluent movement	Repelling 'force' can become to great for reaching the robot target
Single obstacle avoidance	Can recognize irrelevant moving obstacles	Robot can be 'trapped' in repetitive calculation of two paths
A*-search	Finds optimal path	Does not take moving obstacles into account

Table 16 (Dis) advantages of the collision avoidance methods

We can conclude that the best approach would be to use the A*-search which always gives a path if it exists. In addition, to tackle the disadvantage of the A*, when the A*-search returns a path that only avoids a single obstacle, one could use the single obstacle avoidance technique to be able to neglect this obstacle if when the obstacle is leaving its obstructing position.

A combination of methods has been applied by combining the vector field with the single obstacle avoidance (Appendix A.5). The A*-search with a moving obstacle exception should be researched further.

5 Conclusions and recommendations

*T*his chapter will conclude the work described in this thesis, which is part of the work done on the MI20 project since 2002. Furthermore, recommendations are given to further improve the robot soccer system.

5.1 Conclusions

A new design for the Strategy module has been made. This design is the so-called FSM approach because the used plan selection mechanism is based upon FSM's.

For each robot, a role based Player in Strategy exists. Each Player uses a FSM for plan selection. Goalie, Defender and Forwarder can be considered as roles. FSM's for different roles are given. Multiple robots can act as a coordinated robot soccer team when Strategy assigns different Roles to the robots. Such combinations of Roles are the fundamentals for the use of formations and tactics. Switching of strategy is easier because it is possible to play with different formations and different tactics.

The FSM approach is considered better than two previously developed strategies, the Single Neuron approach and the Potential Field approach, because it supports time based behavior, performs better at reductive explainablitity and does not have great disadvantages with regard to adaptability, extensibility, understandability and low computational complexity.

For the robot soccer system different strategies with the FSM Approach have been developed. In this thesis, two strategies, based upon a 1-3 formation and 2-2 formation, are presented. The designed 1-3 formation has one keeper, one defender and three attackers that alternating, time based, adopt shoot attempts. The designed 2-2 formation has one keeper, two defenders behind each other and two attackers next to each other.

The 1-3 formation and 2-2 formation based strategies have been tested in international tournaments. These strategies work well and are able to generate consisted planning while for each robot it is clear if it operates defending or attacking. If the two strategies are compared with each other, some differences are visible during game play. The 1-3 formation is more vulnerable for long dribbles from opponent robots than the 2-2 formation. This vulnerability becomes apparent when opponent robots are faster moving. On the other hand, the designed 1-3 formation is able to recover from failed shoot attempts while the 2-2 formation only assigns one particular robot to shoot when the ball is in some region, even when this robot stopped (undetected) his shoot attempt.

With the 2nd place at the European Championship 2007 reached by using the 1-3 formation and 2-2 formation, it can be stated that the FSM Approach can be used well to create strategic plans for robots.

To provide the Strategy with consistent information a new robot identification algorithm has been introduced. This method uses different color patches to identify the robots. In the new method, wrong association is still possible, but appears considerable less then before.

For adequate execution of the strategic plans also two motion functions are developed. The two motion functions, the Proportional Cosine Controller and the Pursuit, are used both in the system to use the advantages of each of them. The main advantages of the pursuit are constant curvature, exact calculation, slip prevention and the opportunity to use a stop profile. Unfortunately, the use of pursuit also caused unwanted circular motion

for robots that have as main objective to stay moving as fast as possible at a single line. The main advantage of the Proportional cosine control is that it can be used to reduce distance and angular error fast as long as slip and (de)acceleration of the robot is neglected, and is being used for defensive plans with close targets. These functions made execution of Plans possible.

Smith prediction has been introduced to compensate dead time. With smith prediction, the performance of motion control algorithms and state estimation process has been improved.

Collision avoidance techniques have been investigated because frozen situations in the robot soccer game have to be resolved. Playing without opponent already caused stuck situations between team players. The observation was made that this occurred in situations in which two robots tried to advance to a point where each of those two robots robot was closer to than the other robot. Such a situation could arise due lack of coordination or due to change of ball direction or collisions with opponent robots. With a role switching technique the distances between targets and robot positions were minimized and therefore these stuck situations disappeared. However, this method did not prevented situations in which opponent robots were involved. Therefore, a more generalized approach had to be taken. A local vector field method to avoid collisions was developed, which worked with an appealing and repelling principle to generate a new sub-goal for the robot. In the local vector field method only objects near the robots were taking into account, this resulted in oscillating behavior. In the global vector field method, oscillating behavior faded away, because the appealing and repelling principle was used for all objects in the field. Nevertheless, another drawback became apparent. With some configurations, robots drove very inefficient paths, or the goal of the robot became unreachable. Therefore, a more straightforward collision avoidance method was added which only tried to pass the first obstruction, the single-obstacle avoidance method. The vector field method showed in overall more fluently movement and the pass-first method showed to keep pursuing its target and more collisions. Avoidance seemed not always to be needed, especially when the obstructing robot is moving away from its obstructing position. Sometimes a robot was trapped between other robots trying to avoid in turn these robots. Therefore an A*-search algorithm was developed to create a new sub goal for the robot. This A*-search gives always a path if such a path exists and is considered the most robust method to resolve stuck robot situations. Improvements for the A*-search does not involve in resolving exceptions in which the method does not work as solution for frozen situations, but in finding exceptions in which the method is considered to be unnecessary.

With the developed Strategy module, strategies based upon the 1-3 formation and 2-2 formation, new robot identification algorithm, motion functions and collision avoidance techniques the robot soccer system of MI20 has been improved as whole.

5.2 Recommendations

A lot of work has been performed, but still a number of aspects can be improved.

In the Strategy module, one could start with creating an xml-language to describe the FSM. In addition, the development of an FSM editor for the system would be useful. Another more Artificial Intelligence direction would be the development of a decentralized coordination algorithm. With regard of communication between Players, situations in which communication is profitable and representing the communication, a lot of work can be performed. To explain why a finite state machine produces some strategic action more research could be done in tracing. Furthermore, more probabilistic analysis should be performed during game play

If it is chosen to continue the development of a new strategy for team play, a new 2-2 formation could be developed that uses the same type of attackers of the developed 1-3 formation. It is expected that it has the same defensive robustness and furthermore a stronger offensive strength than the current 2-2 formation based strategy. A more offensive approach is also recommended in creating a 1-1-2 or 0-4 formation in which a player also covers the line between the ball and own half.

For robot identification, also different approaches are possible, like using least squares fitting for robot identification or background subtraction for robot detection. With regard of the color patch identification, an improvement could be added to check if an area that is supposed to be black is not some blob. In the current method there are situation possible in which multiple patches are composed together in such way that the system could not tell were some color patch is located. With such a black space check, multiple solutions could be distinguished. These situations did not occurred noticeably yet, likely because there are few situations in which such precise configurations are maintained for a long period of time.

The new developed motion function improved the motion of the robots, nevertheless it is expected that faster speeds can be obtained. So more research can be performed toward accurate reactive motion that can be used at faster speeds than 1.5 m/s.

When attention is given to state estimation, a real improvement to the system would be the development of a algorythm that can determine the dead time automatically. Furthermore adapting the kalman filter in such way that it uses the control signal history is expected to improve estimation.

With regard of collision avoidance, optimizing the A*-search is recommended. An A*-search which recognize obstacles which are not relevant should be researched further. With regard of role swithing only a global method has been developed, perhaps a local approach does not cause additional risky situations.

6 Bibiliography

[1] MI20 official site, www.robosoccer.nl

[2] FIRA official site, www.fira.net

[3] MiRoSot information, http://www.fira.net/soccer/mirosot/overview.html

[4] R. Seesink, "Artificial intelligence in a multi agent robot soccer domain", Master Thesis, University of Twente, 2003

[5] C. Petit, "Strategy for robot soccer systems – Implemented for the MI20 team", Master Thesis, University of Twente, 2006

[6] A.M. Poelman, "Decision making in robotsoccer", Project report, University of Twente, 2004

[7] R. van Amstel, "The MI20 score_array", Project report, University of Twente, 2003

[8] P. de Groot, "Enhancing the performance and testability of the MI20 robot soccer system.", Master Thesis, University of Twente, 2006

[9] M. Buth, **"**Ball-handling motion control for soccer playing mini-robots", Master Thesis, University of Twente, 2006

[10] M. Poel, A.L. Schoute, W. Dierssen, N. Kooij and R. Seesink, "Twente makes its d´ebut at the FIRA EC Robot Soccer 2003", 2003

[11] R. S. Pressman, "Software Engineering – A practioner's approach", McGraw-Hill, Berkshire, 2000

[12] W. Dierssen, "Motion planning in a robot soccer system", Master Thesis, University of Twente, 2003

[13] T. Verschoor, "Dynamic motion planning of soccer playing mini-robots", Master Thesis, University of Twente, 2004

[14] R. van Amstel, "The MI20 score_array", Project report, University of Twente, 2003

[15] A. E. F. Segrouchni and S. Haddad, "A Coordination Algorithm for Multi-Agent Planning" *in Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away: agents breaking away*,Eindhoven , Pages: 86 - 99, 1996

[16] G. Noubit, D. R. Stephens, "Specification of timed finite state machine in Z for distributed real-time systems" *in proceedings of the 4th workshop on future trends of distributed computing systems*, Page(s):319 – 325, Sep 1993

[17] G. P. Khiam, L.K. Cheong,L.B. Chen, "Socrates – team description paper" *in Proceedings* FIRA RoboWorld Congress 2006, 2006

[18] R.C. Coulter, "Implementation of the Pure Pursuit Path Tracking Algorithm", tech. report CMU-RI-TR-92-01, Robotics Institute, Carnegie Mellon University, January, 1992.

[19] K.L. Koay and G. Bugmann, "Compensating Intermittent Delayed Visual Feedback in Robot Navigation" *in Proceeding of Taros'04*, 2004.

[20] N. Kooij , "The development of a vision system for robotic soccer", Master Thesis, University of Twente, 2003

[21] J. van der Linden, "Dynamic avoidance control of soccer playing mini-robots", Master Thesis, University of Twente, 2005

[22] G. Dudek and M. Jenkin, "Computational principles of mobile robotics", Cambridge University Press, New York, 2000

[23] R. Brandenburg, J Kalverboer, F. Mak, A Veenhuizen, "Robotsoccer - Improving robot soccer", Project report, University of Twente, 2006

A Appendix

A.1 Motivation for distributed plan generation

State machine(s) can be used in the strategy module in different ways. First one could create one state machine for plan generating for all robots. A second method is that for each robot a state machine can be created and the third method is to create a state machine for each role. This can be seen as a means to perform a role as known in field soccer. Such roles have to be designed. A way to design strategies is to look at plays performed during tournaments of the FIRA.

The first option is being disregarded because it was decided to create strategies as simple as possible. With creating a state machine for each role it is also possible to assign the these roles at a static way to the robots and that would exactly lead to the same result as the second option, because each role would then always have one state machine. Furthermore, when choosing for the third option one also has freedom to dynamic assign roles to robots, which can become useful when resolving problems with undetected or stuck team robots.

Therefore, it is chosen to create state machines for the different roles we want for the team robots such as goalkeeper. Synchronization between the finite state machines is the problem for the FMS's to operate in unison. In multi agent system, this problem is seen as a coordination problem. Synchronization in the FSM approach can be established with a shared clock and events. Moreover, communication between the behaviors can establish synchronization.

One could think of some hybrid structure in which a state machine is being used for plan generation for more robots with state machines generating plans for single robots. This is option is kept open for further research because simplicity is considered important.

A.2 Coordination between Players

The benefits of coordinated planning are according to Seghrouchni [15]:

- Harmful Plans are cancelled.
- Advantage is taken of helpful interactions.

First distributed coordination shall be discussed, second a previous developed centralized method shall be discussed.

Distributed coordination

If coordination between the Players is created with a distributed algorithm, communication channels between the Players are necessary.

The main requirements for distributed coordination are:

- Communication between Players.
- Recognition of potential interactions between Plans.
- Negotiation between Players in the case of conflictual situations.

Seghrouchni [15] also proposed a coordination algorithm consisting of two phases. In analogy of this algorithm, a coordination algorithm for use in our system is described. The players are initialized with already coordinated plans. Furthermore, players can only adopt another plan when this is coordinated with the other players. Therefore, the plans of the players stay coordinated.

Phase 1:

1. Player g_n produces a new plan π_n.
2. g_n sends his plan π_n to the other Players.
3. Each player g_k ($k \neq n$) receives π_n and tries to coordinate it with plan π_k.
4. The other players return to g_n the coordination results in a data structure which represents the possible synchronization between the different plans.

Phase 2:

1. Player $g_n$produces a coordinated plan Π_n by taking into account the possible synchronizations with the other players and its own desired plan π_n.
2. g_n send synchronization messages to all other players
3. Player g_n starts performing his plan Π_n.

Each Player should use this algorithm when it deliberates upon a new plan and these phases should be mutual exclusive with the plan generation process of the other players.

Centralized coordination

Above algorithm is a pure distributed coordination algorithm. The coordination algorithm used by Seesink [4] used a coach agent to solve the coordination problem centralized. It uses the principle of resource claiming [4]; a plan is seen as a resource that can only be used by a single player. In Seesinks approach, the coordination has two steps:

Step 1:

1. A Player calculates for each action a desirability score.
2. The Player sends its desirability scores to the coach agent.

Step 2:

1. The coach agent compares the different scores of the player and decides which resource is most suitable for the player to claim. If another player already owns that resource, the player is not suitable to claim it.
2. The coach agent assigns and sends the most claimable plan to the player.
3. The Player starts performing the assigned plan.

Each Player should use this algorithm when it deliberates upon a new plan and these phases should be mutual exclusive with the plan generation process of the other players.

A.3 Plan's position calculation

Some of the plans being used in the described strategy of this chapter are explained more in detail. In 3.1 an overview of plans in the system is given. In this section a more detailed view of how the plans PEN_DEFENDER and BACKUP_ATTACKER calculate positions is given. The plan REACTIVE_MOVE does not perform calculations but only passes on its (given) attributed. The plan SPIN only notifies motion to let the robot spin. The conversion of positions to control signals, for the robots, is done in the motion module. For the positioning and creation of the control signals for the plans SCORE_DIRECT and PANIC_SHOOT, one should read the work done by Buth[9]. The conversion of the plans calculated positions to control signals for the plans REACTIVE_MOVE, PEN_DEFENDER and BACKUP_ATTACKER is considered in chapter 4.2.

A.3.1 PEN_DEFENDER[id]

In this section,it is explain how the position for a robot with the PEN_DEFENDER plan is calculated. This plan is parameterized with a parameter id. How higher the id, how greater the distance is from the goal for this defensive positioning. An informal description for this plan was already given in 3.1, and it being recalled that the PEN_DEFENDER calculates positions for the robot, which should result in driving the robot in straight lines around the goal area at some distance, whilst keeping space for the goalkeeper to clear the ball.

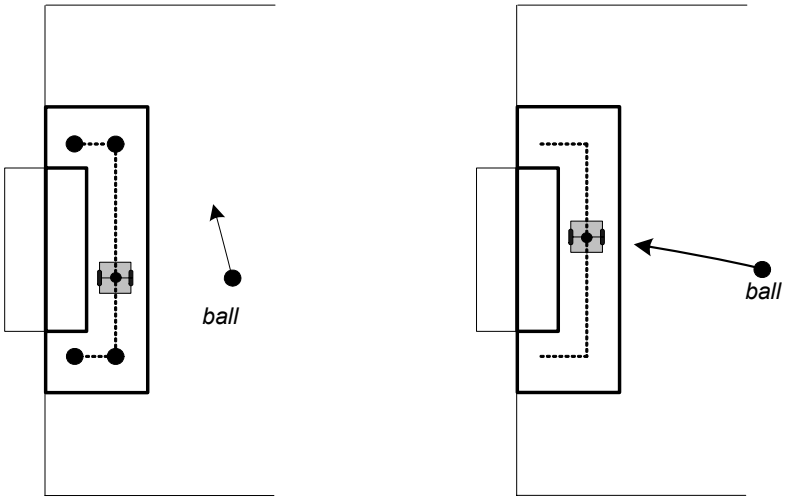

Figure 36 Positioning of PEN_DEFENDER

In Figure 36, the dotted lines between the dark points represent the straight lines where the robot has to travel over. If the direction of the ball crosses one of the dotted lines, the robot has to position there (left situation), otherwise the robot positions at the same height or width as the ball (right situation).

A.3.2 BACKUP_ATTACKER

In this section, it is explained how the exact position for a robot with the BACKUP_ATTACKER plan is calculated.

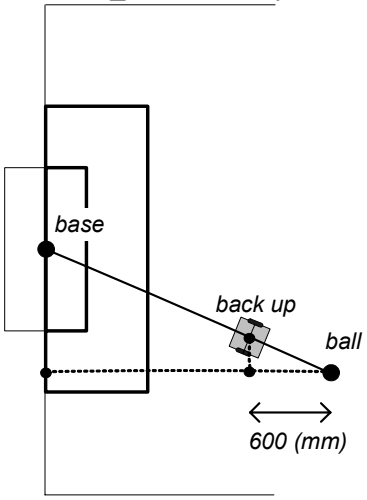

Figure 37 Positioning of BACKUP_ATTACKER

The calculated position for the BACKUP_ATTACKER is between the own goal and the ball at a distance of 600 mm behind the ball, as shown in Figure 37. The robot should block the ball when it suddenly starts moving towards the goal while in the meantime the distance to the ball is small enough to take fast offensive action.

A.4 Collision recovery

Because avoiding collisions is very difficult, one should also implement a method to recover from collisions. A simple method to detect a collision is to take into account the lack of progress given a history of send control signals. Then applying a turn back and then moving to the original target provides already a deadlock free game. One could think of situations in which more efficient methods could be applied but it was chosen to tackle the problem with an overall solution that can be used to recover from frozen situations.

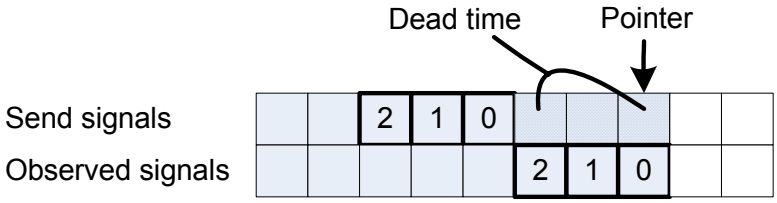

Figure 38 Circular buffer for linear velocities

First thing to do is to keep track of sent control signals and observed speed using a circular buffer as shown in Figure 38. Each time that new world data is being received the linear speed of the robot is stored at the pointers index in the buffer and the pointer is incremented. Each time a control signal is received this is stored at the pointers index. Second, it has to be determined if a robot is responsive. Dead time is also taken into consideration; dead time is the delay between signal send and being observed, in consideration. For the dead time 3 world data updates (99 ms) are assumed in Figure 38. Furthermore the last 3 signals observed are compared with there equivalent in the send signal history. When all these 3 observed speeds are less than 70% of the send signals, it is assumed that the robot is obstructed or stuck. To a stuck robot, for some time (13 World Data updates) a turn back (when the robot was moving forward) or turn forward

(when the robot was moving backward) command with v_l=±800 mm/s and v_{ang}=5.5 rad/s is send.

A.5 Hybrid method for collision avoidance

During the World cup 2006 a hybrid method was adopted for team play.
A robot first evaluated the vector field method, but if from robots point of view the angle between the original goal and the sub-goal became greater than 0.4π the single obstacle avoidance was used. During the matches played in the world cup 2006, some goals scored by the opponents could be imputed to a robot which got trapped between other robots trying to avoid in turn different other robots, and hindering the other robots. Therefore an A^*-search algorithm was developed to create a new sub goal for the robot.

Avoidance seemed not always to be needed, especially when a team robot, which has to be passed, has a goal further away than the avoiding robot. This exception was also integrated in the hybrid method after the world cup 2006.

The visibility graph method worked fine in the European cup 2007. The moving obstacle exception did also work fine in the European cup 2007 and improved the hybrid method.